

论文汇报

Learning Video Representations from Large Language Models

(从大型语言模型中学习视频表示)

▶ 汇报时间：2024.04.27

▶ 汇报人：梁琨

目 录

CONTENT

壹	研究背景	03
貳	论文概述	12
叁	概念剖析	16
肆	实验设计	20

1 Part One

研究背景

介绍本篇论文的研究背景，重点介绍两个概念并回顾过去的研究成果——

**Vision-language representation learning
Generative Visual Language Models (VLM)**

概念：

这是一种机器学习方法，旨在让计算机理解并关联视觉内容（如图像和视频）与自然语言描述。通过使用度量学习技术将这两者映射到同一空间，可以更有效地进行检索、理解和生成图像或视频的描述。

图像-文本领域

扩大模型和预训练数据。CLIP在400M图像-文本对上进行了预训练，使用了对比损失(InfoNCE)，而CoCa通过单一基础模型统一了对比和生成方法

提出了不同的预训练任务来学习视觉和文本模态之间更细粒度的关联。例如掩码语言建模(MLM)和字幕生成

学习方法

视频-文本领域

通过对比学习或无监督对齐从未经整理的视频和机器生成的音频字幕中学习

将性能良好的图像-文本模型适应到视频，或者从单个帧到多个帧的渐进式学习。

相比之下，我们的方法利用语言模型在长格式视频上生成时间密集的文本监督。

生成式视觉语言模型（VLM）最初被用于基于循环网络和基于Transformer的架构的**图像/视频自动生成描述**。近期，生成式VLM通过训练多模态Transformer模型在视觉-文本对上的表现，实现了多种视觉任务的统一。同时，生成式VLM还通过利用在大量文本语料库上预训练的数十亿参数规模的语言模型，在多模态任务中实现了零样本或少样本提示的卓越表现。在我们的工作中，我们展示了生成式VLM能够为**长视频生成叙述，由此产生的视频-文本数据对视频-语言表征学习有积极作用。**

大规模多模态视频数据集对视频理解任务至关重要，但收集起来却非常困难。传统的视频-文本数据集要么场景有限，例如烹饪，要么规模不够大，无法学习通用的视频表示。Miech等人通过自动音频转录从长篇指导视频中抓取了超过1亿个视频-文本对。然而，自动语音识别（ASR）引入了明显的文本噪声和视觉-文本不对齐。WebVid包含了1000万个带有文本描述的短视频。但它比图像对应的小几个数量级，而且由于来源于库存视频网站，因此更难以扩大规模。最近发布的Ego4D数据集提供了3600小时的以自我为中心的视频，其中每几秒钟就有人工标注的书面句子叙述，但这需要大量的手工努力。**相比之下，我们的方法通过使用来自大型语言模型（LLM）的监督自动叙述视频，展现了一个有前途的替代方案。**

2 *Part Two*

论文概述

概括地叙述论文的主要工作内容，并提出论文的创新点，着重提出涉及到的模型、概念和原理。

视觉（视频）语言表征学习存在的问题：

论文中讨论如何利用大规模的图像-文本数据来训练计算机视觉模型，以改进其在各种任务上的表现，如分类、检测和生成图像。

尽管这种方法在图像方面取得了显著的进展，**但在视频方面却受到了限制**，因为可用的配对视频-文本数据集相对较小。尽管过去十年视频数据的获取量大幅增加，但配对的视频-文本数据集仍然**远远小于**图像-文本数据集。

生成式视觉语言模型

视觉语言表示学习

论文介绍了LAVILA (Language-model augmented Video-Language pre-training)，这是一种通过利用大型语言模型 (LLMs) 来学习视频语言表示的新方法。

生成式视觉语言模型：

我们将预训练的LLMs重新定位为对视觉输入进行条件化，并对它们进行微调以创建自动视频叙述者。

视觉语言表示学习：

通过这些额外的自动生成叙述，以**对比方式学习**的视频文本嵌入在多个第一人称和第三人称视频任务上优于以前的最先进水平，无论是在**零样本**还是**微调**设置中。

对比学习目标

CLIP

对称交叉熵损失函数InfoNCE

Narrator

Rephraser

优点：

包括对长视频的密集覆盖，更好地同步视觉信息和文本的时间，以及更高的文本多样性。

先利用大型语言模型 (LLMs) 来自动为视频生成文本描述，再利用大规模的视频数据来训练视频-语言模型。

双向编码器结构

视觉编码器

TimesFormer

文本编码器

Transformer

3 *Part Three*

概念剖析

介绍论文中的模型、原理和相关的概念，重点介绍论文的创新点，以及训练和推理方法。

A video V is a stream of moving images I . The number of frames $|V|$ can be arbitrarily long while video models typically operate on shorter clips, which are often in the range of a few seconds. Therefore, we skim through a long-form video and represent it by a set of N short clips, *i.e.* \mathcal{X} . Each clip x_i is defined by a specific start and end frame $x_i = \{I_{t_i}, \dots, I_{e_i}\}$, where $0 < t_i < e_i \leq |V|$, and is typically associated with some annotation y_i . This annotation could be a class label or free-form textual description of the clip. We denote a video by the set of annotated clips with their corresponding annotations, *i.e.* $(\mathcal{X}, \mathcal{Y}) = \{(x_1, y_1), \dots, (x_N, y_N)\}$. Note that the annotated clips often cannot densely cover the entire video due to the annotation cost and visual redundancy, *i.e.* $\bigcup_i [t_i, e_i] \subsetneq [0, |V|]$.

对于视频 V , 帧数为 $|V|$, 将其拆分为多个clips

由于视频标注成本和视觉冗余, 不是所有的clips都能够拥有其对应的annotations。

即生成的视频文本对不能密集地覆盖整个视频

这是视觉语言表示学习中数据集存在的一大问题, 本文通过提出Narrator和Rephraser来解决这个问题

Narrator的介绍

NARRATOR is a *visually-conditioned LLM* that pseudo-labels existing and new clips with narrations, generating new annotations $(\mathcal{X}', \mathcal{Y}')$



NARRATOR生成新的动作描述，可能会关注其他正在与之互动的对象。

Traditional LLMs, such as GPT-2 [50], are trained to generate a sequence of text tokens $(s_1 \cdots s_L)$ from scratch by modeling the probability of the next token given all tokens seen so far: $p(s_l | s_{<l})$. NARRATOR repurposes existing LLMs to be conditioned on the visual input and is trained on the original annotations $(\mathcal{X}, \mathcal{Y})$. The resulting model produces dense new annotations $(\mathcal{X}', \mathcal{Y}')$ on the full video. Following the formulation of factorized probabilities in language models [5], we model the visually conditioned text likelihood as follows:

$$p_{\text{NARRATOR}}(y' | x') = \prod_{\ell=1}^L p(s'_\ell | s'_{<\ell}, x'). \quad (2)$$

原始LLM的任务：

根据到目前为止看到的所有的tokens，预测下一个token： $p(s_l | s_{<l})$

Narrator任务：

在视觉输入条件下，使用现有的LLMs模型，在原始的数据集 $(\mathcal{X}, \mathcal{Y})$ 上训练模型。

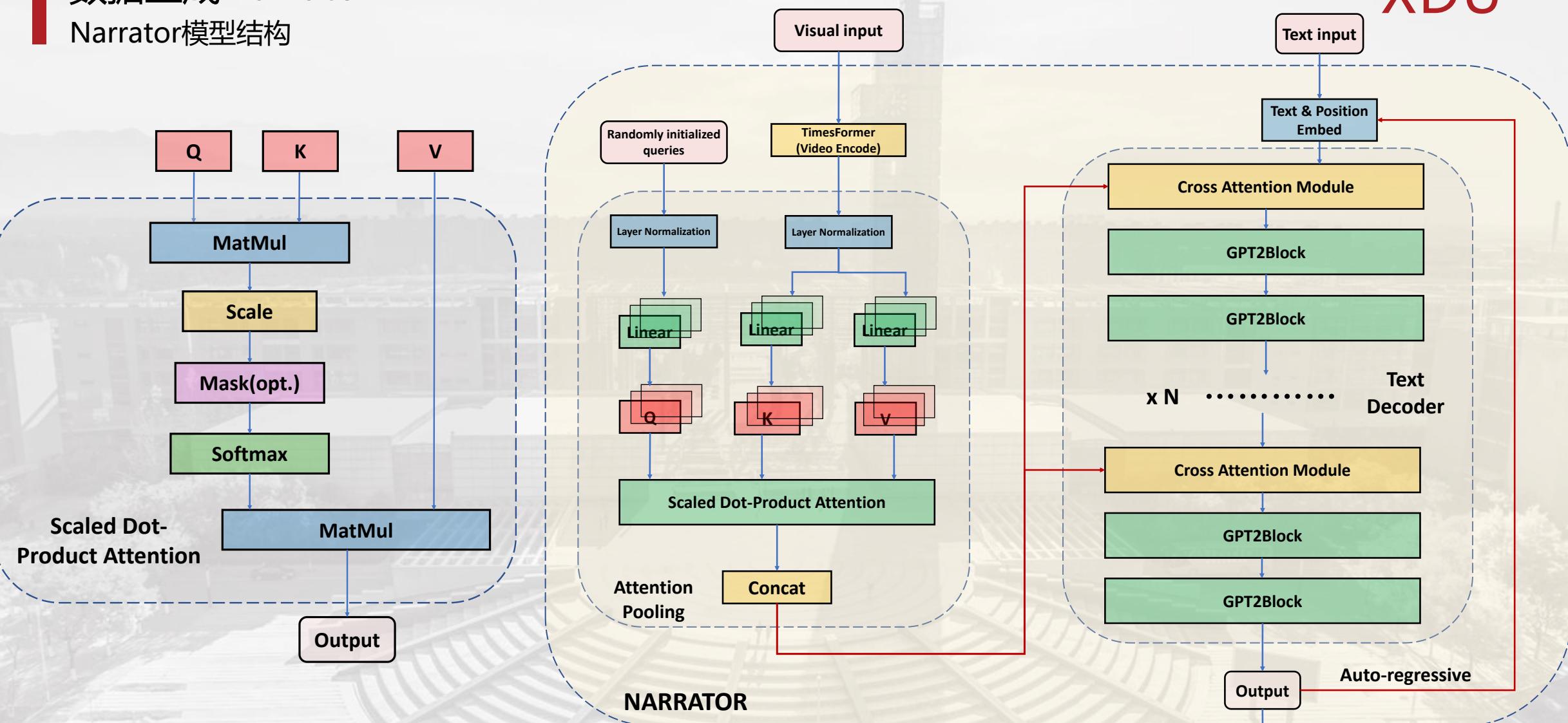
最终的模型在整个视频数据集上生成稠密的标注信息，即 $(\mathcal{X}', \mathcal{Y}')$

NARRATOR's architecture is a visually conditioned auto-regressive Language Model. The visual encoder is by default TimeSformer-L while the text decoder is a GPT-2 XL. During inference, we use nucleus sampling [29] with $p = 0.95$ and return $K = 10$ candidate outputs.

数据生成Narrator

XDU

Narrator模型结构



数据生成Narrator

XDU

Narrator模型结构

Attention Pooling:

随机初始化固定长度的向量: $\mathbf{q} \in \mathbb{R}^{N_q \times D_t}$

视觉特征 (由TSF提取) : $\mathbf{v} \in \mathbb{R}^{(T \times H' \times W') \times D_v}$

$\mathbf{q}', \mathbf{v}' = \text{LayerNorm}(\mathbf{q}), \text{LayerNorm}(\mathbf{v})$

$$\text{head}_i = \text{softmax}\left(\frac{(\mathbf{q}'\mathbf{W}_Q^{(i)}) (\mathbf{v}'\mathbf{W}_K)^{\top}}{\sqrt{d_0}}\right) \cdot (\mathbf{v}'\mathbf{W}_V)$$

AttentionPool = Concat (head₁, ..., head_h) · \mathbf{W}_O

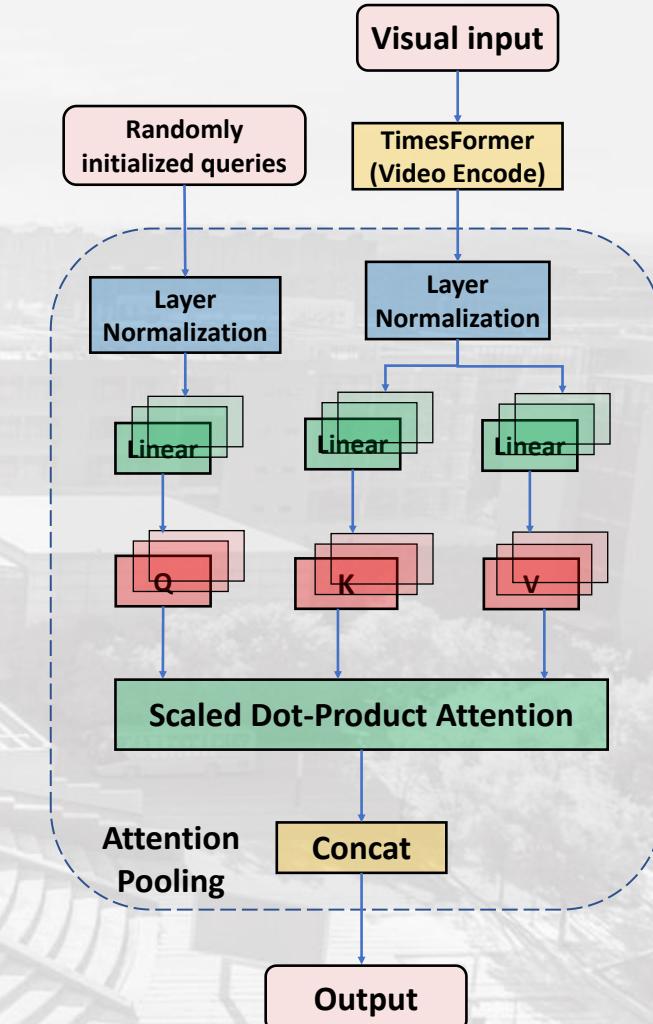
$$\mathbf{W}_Q \in \mathbb{R}^{D_t \times d_0}$$

$$\mathbf{W}_{K/V} \in \mathbb{R}^{D_v \times d_0}$$

$$\mathbf{W}_O \in \mathbb{R}^{(h \cdot d_0) \times D_t}$$

AttentionPool(\mathbf{q}, \mathbf{v}) $\in \mathbb{R}^{N_q \times D_t}$

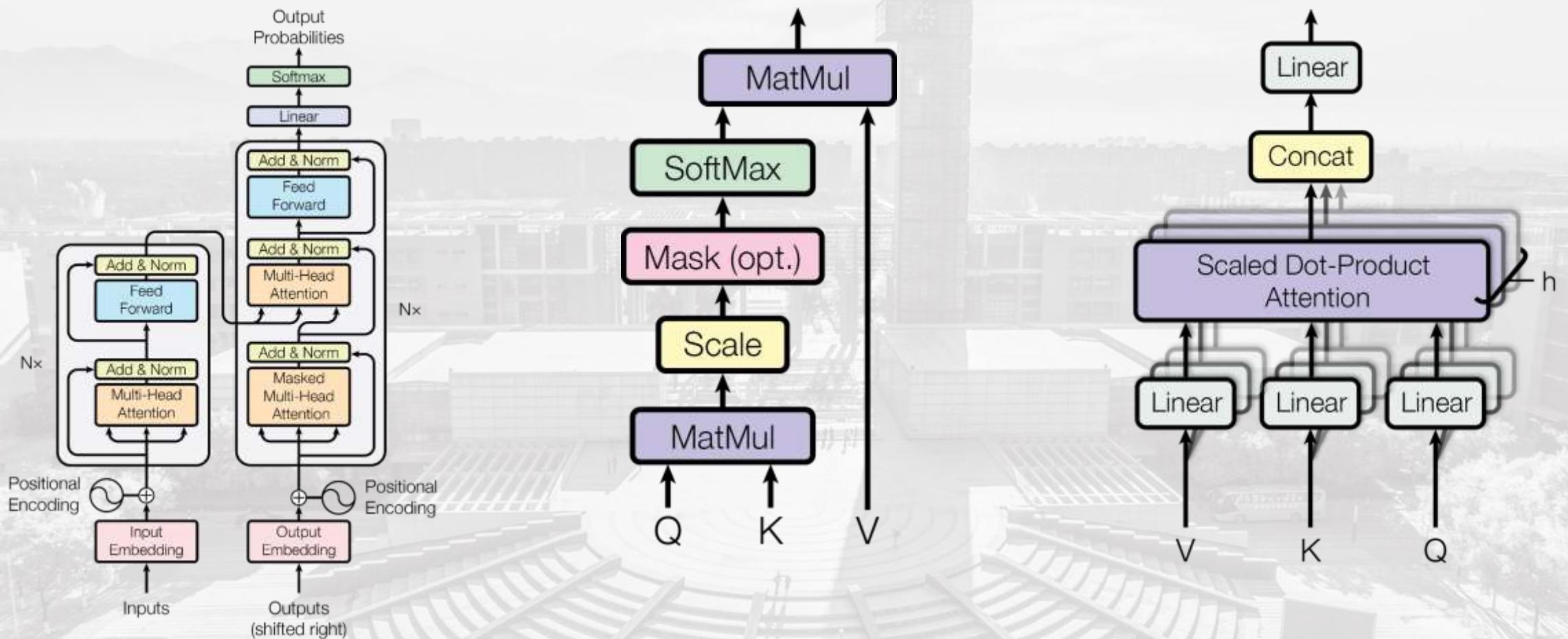
经过注意力池化后，生成固定长度的隐藏状态。它稍后将被输入到文本解码器的交叉注意模块中。这确保了文本解码器关注相同数量的视觉特征，而不考虑输入的视觉分辨率，例如 224×224 或 336×336 。



数据生成Narrator

XDU

Narrator模型结构——注意力机制



数据生成Narrator

Narrator模型结构——注意力机制

XDU

Self-attention

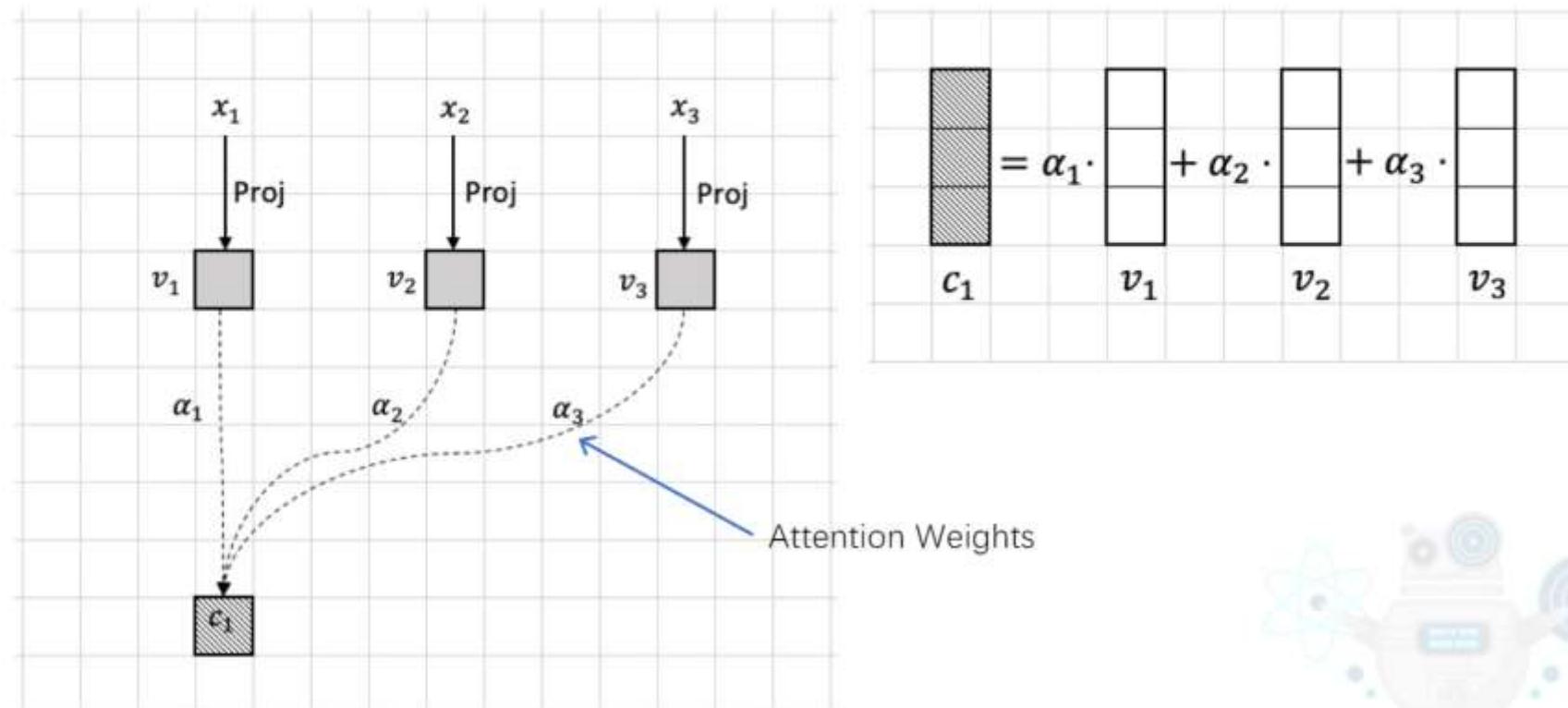
input #1
1 0 1 0

input #2
0 2 0 2

input #3
1 1 1 1

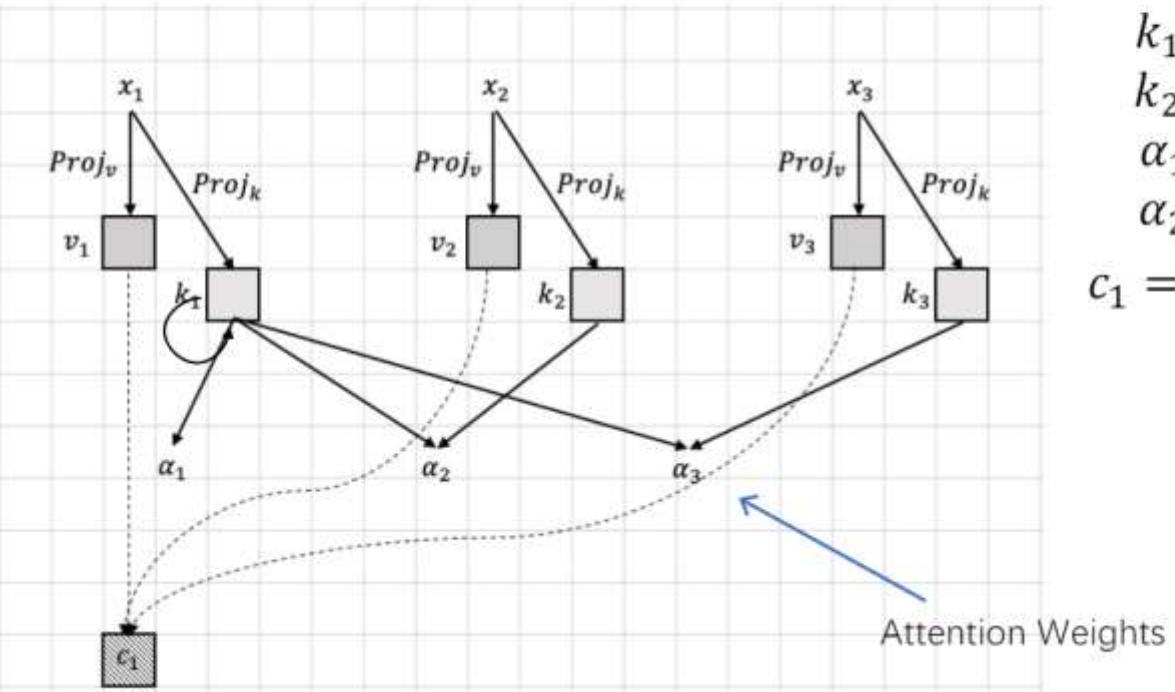
Self Attention : Transformer中的注意力

Attention in Transformer: Self Attention



Self Attention : Transformer中的注意力(2)

让我们把注意力计算变得“复杂”一点



$$k_1 = x_1 \cdot Proj_k$$

$$k_2 = x_2 \cdot Proj_k$$

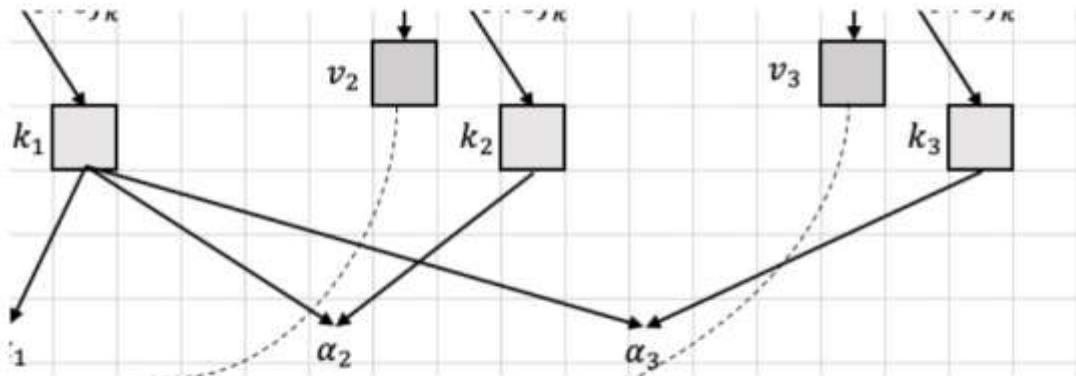
$$\alpha_1 = k_1 \odot k_1$$

$$\alpha_2 = k_1 \odot k_2$$

$$c_1 = \alpha_1 \cdot v_1 + \alpha_2 \cdot v_2 + \alpha_3 \cdot v_3$$

Self Attention : Transformer中的注意力(3)

为什么要用两个向量的点乘？



$$\text{点乘} : k_1 \odot k_2 = \|k_1\| \cdot \|k_2\| * \cos(\theta)$$

一定程度上表示相似程度

为什么不是余弦相似度？

$$\text{余弦相似度} : \text{similarity}(k_1, k_2) = \frac{k_1 \cdot k_2}{\|k_1\| \cdot \|k_2\|}$$

严格讲：没有定论

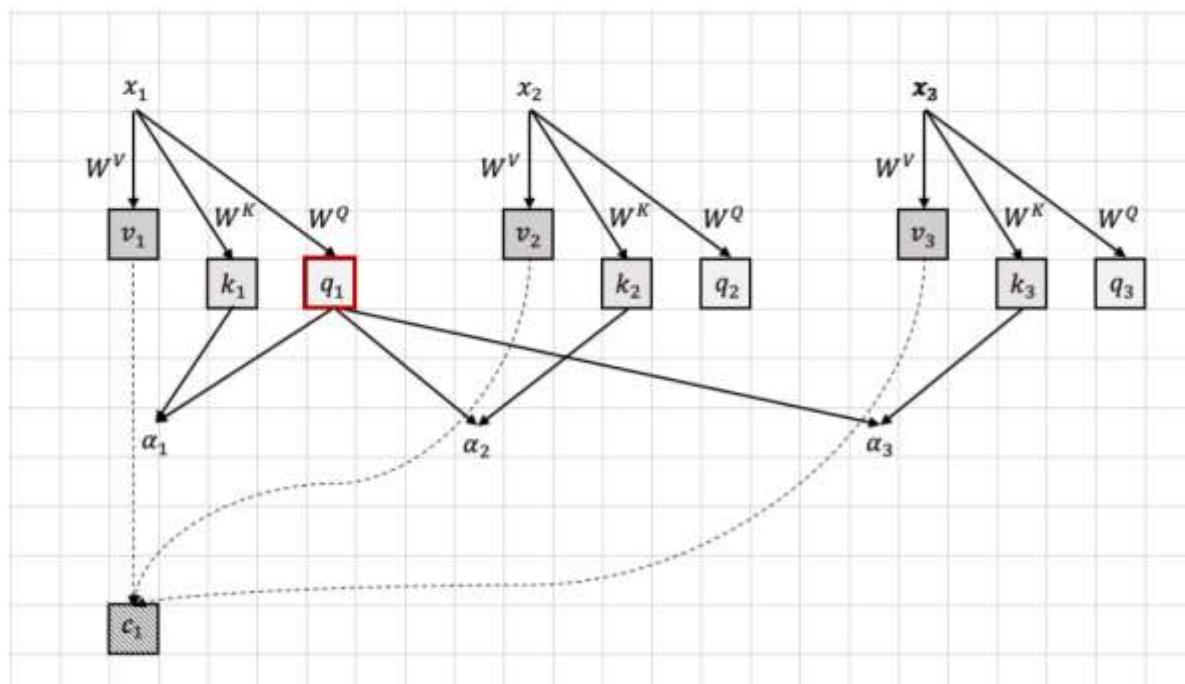
简单讲：1. 为了方便

2. 可能需要考虑长度

3. 总之这样work了

Self Attention : Transformer中的注意力(4)

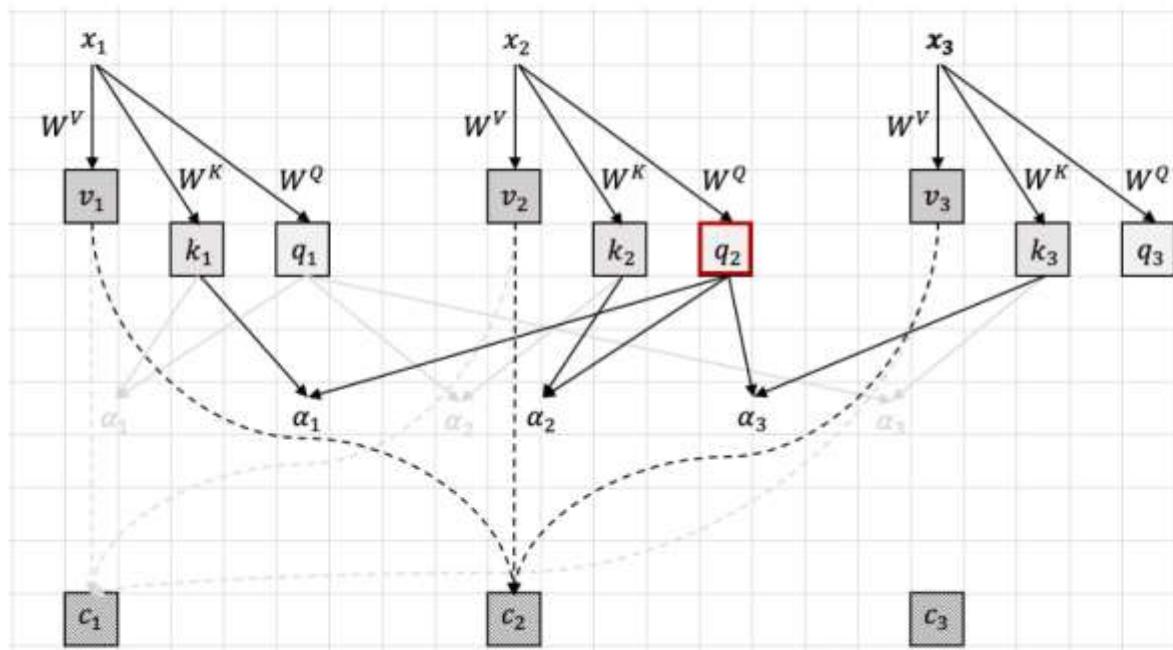
让我们把注意力计算变得“更复杂”一点



$$\begin{aligned} \alpha_1, \alpha_2, \alpha_3 &= \begin{matrix} \square & \square & \square \end{matrix} \cdot \begin{matrix} q_1 \\ k_1 & k_2 & k_3 \end{matrix} \\ c_1 &= \alpha_1 \cdot \begin{matrix} \square \\ v_1 \end{matrix} + \alpha_2 \cdot \begin{matrix} \square \\ v_2 \end{matrix} + \alpha_3 \cdot \begin{matrix} \square \\ v_3 \end{matrix} \end{aligned}$$

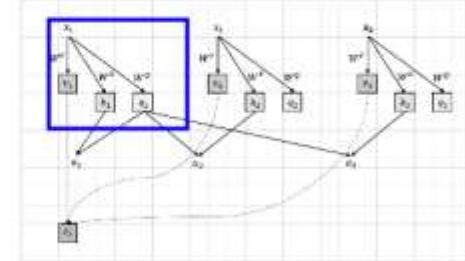
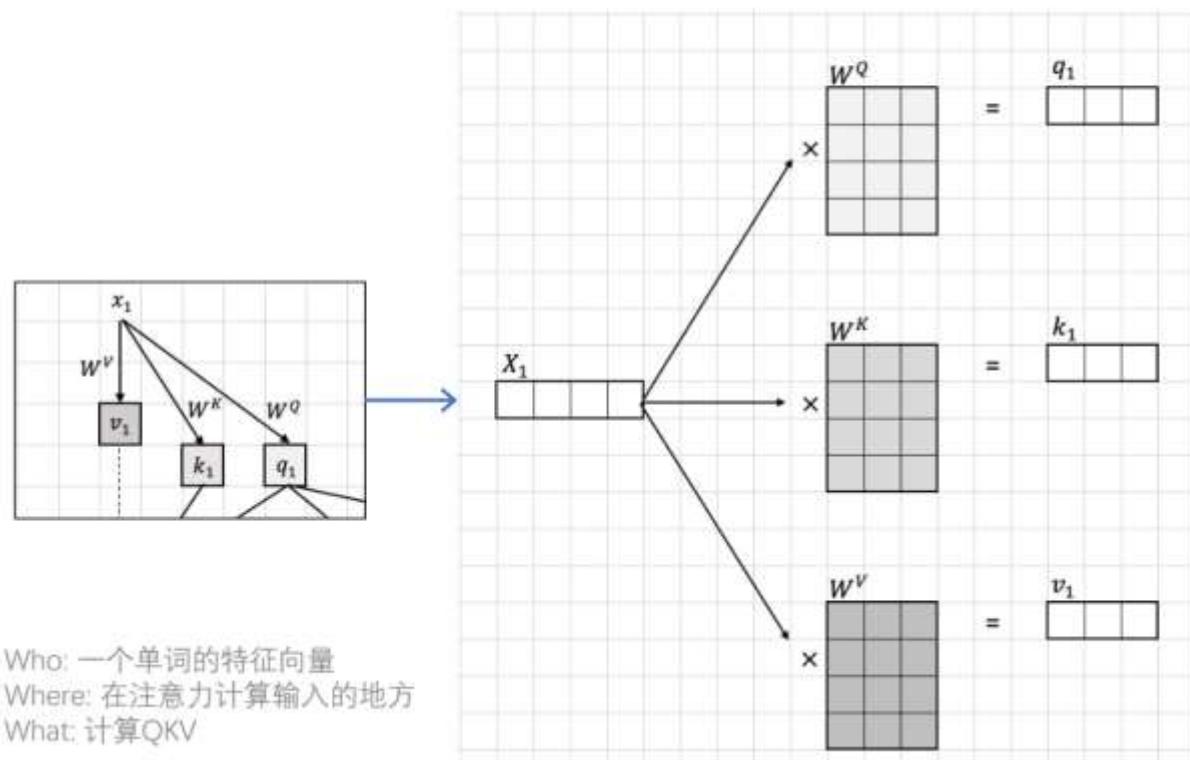
Self Attention : 对单个Token如何计算 ? (3)

让我们把注意力计算变得“更复杂”一点



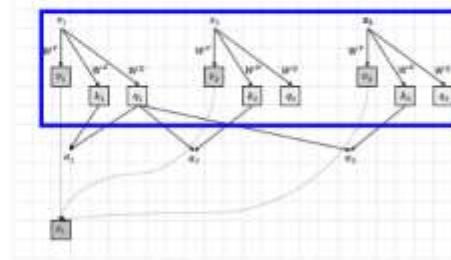
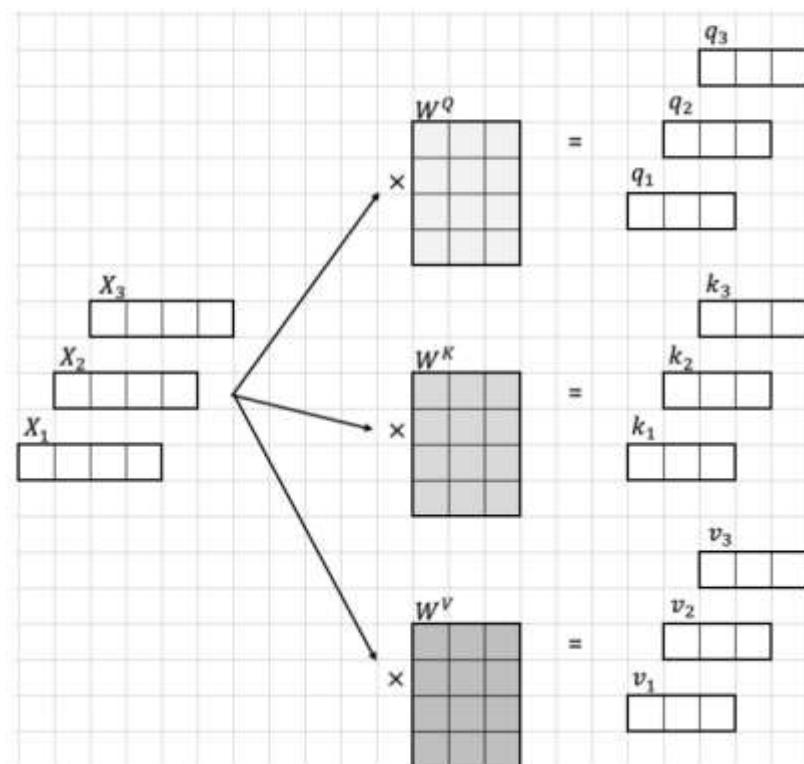
Self Attention : 对单个Token如何计算？

让我们把前面的图展示更清楚些



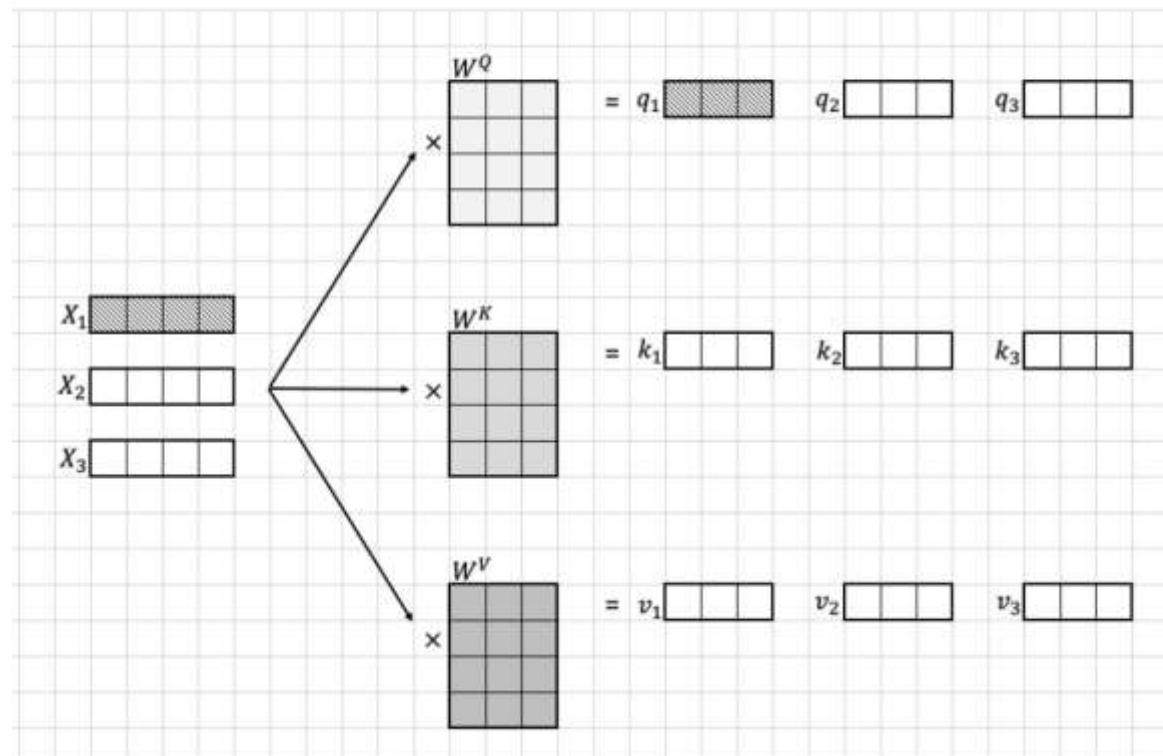
Self Attention : 对单个Token如何计算 ? (2)

多个Tokens作为输入，分别使用同样的QKV映射



Self Attention : 对单个Token如何计算 ? (3)

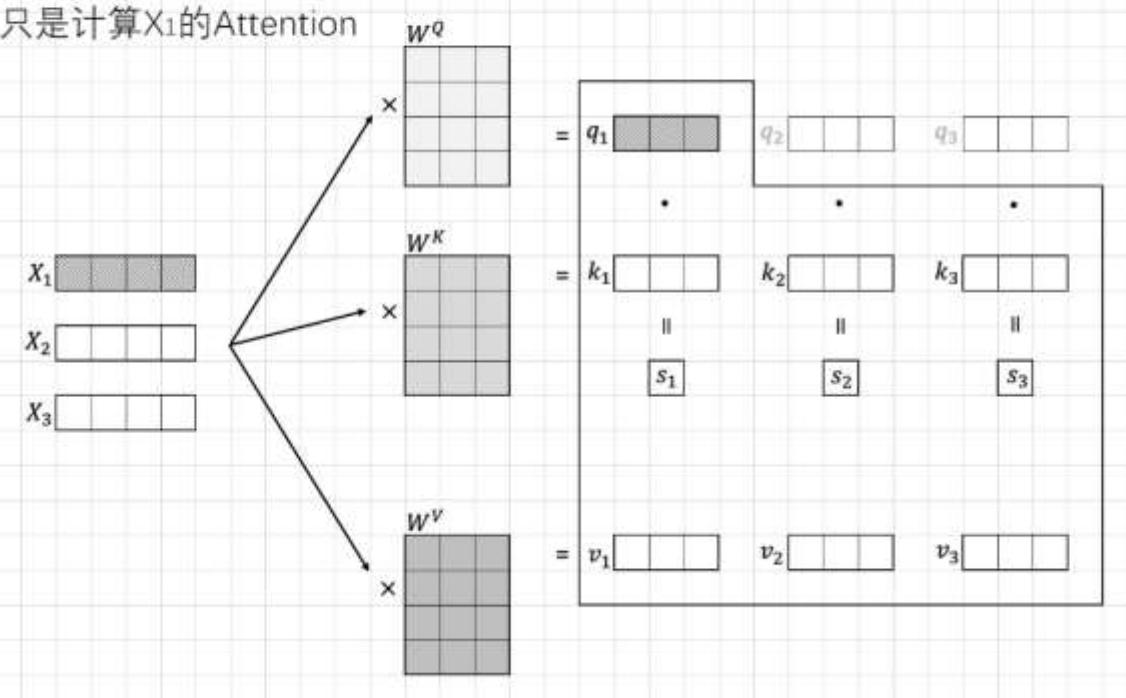
现在我们计算 x_1 在整个序列中的Attention



Self Attention : 对单个Token如何计算 ? (4)

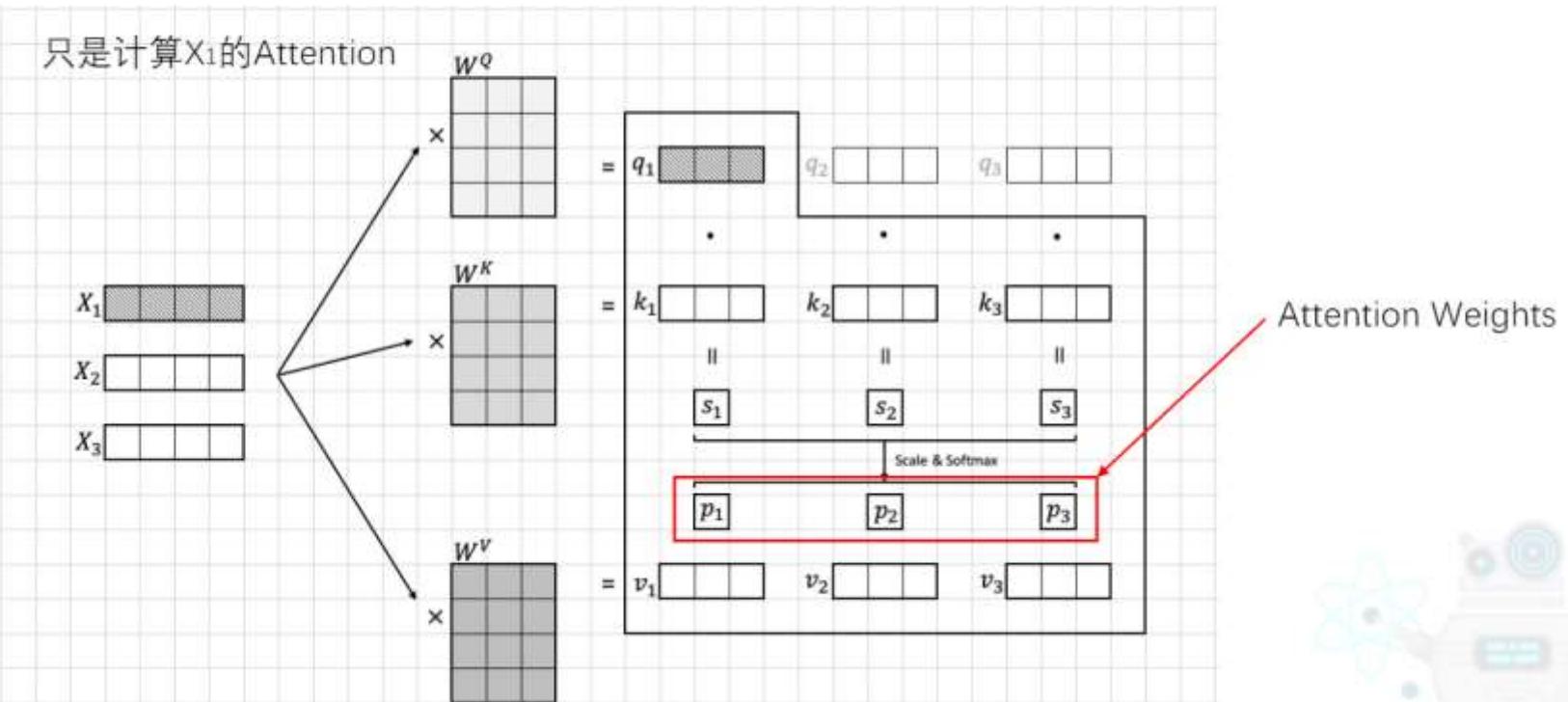
现在我们计算 X_1 在整个序列中的Attention

只是计算 X_1 的Attention



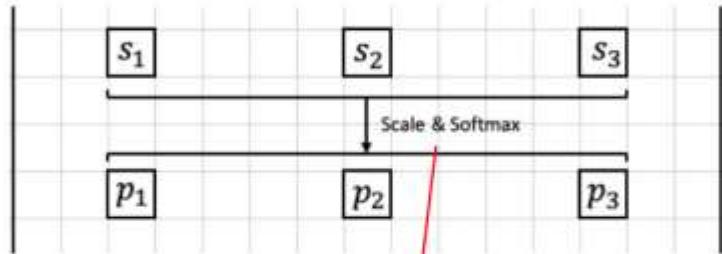
Self Attention : 对序列如何计算 ? (4)

现在我们计算 X_1 在整个序列中的Attention



Self Attention : Scale for Softmax

现在我们计算 x_1 在整个序列中的Attention



$$\text{Attention} = \text{Softmax}\left(\frac{(S)}{\sqrt{d_k}}\right)$$

Why $\sqrt{d_k}$?

- Variance(var)表示什么？序列的波动
- 序列var越大，那么经过softmax越容易偏向大值
- 假设序列(feature)Q和K每一位都是iid，并且是random variable(std=1, mean=0)
- 那么 $S(Q * K^T)$ 的var就是 d_k
- 所以需要把var稳定回1.0

Why Scale?

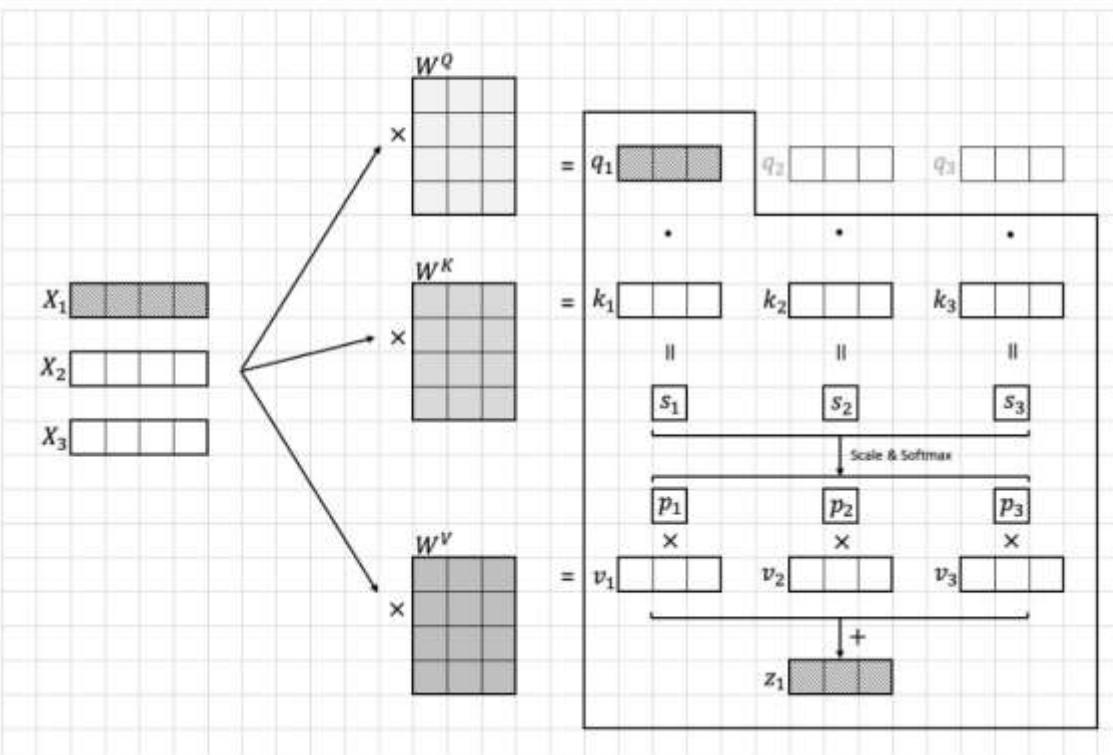
```
>>> paddle.nn.functional.softmax(paddle.to_tensor([1.0, 20.0, 400.0]))
Tensor(shape=[3], dtype=float32, place=CPUPlace, stop_gradient=True,
      [0.00000000, 0.00000000, 1.00000000])
>>> paddle.nn.functional.softmax(paddle.to_tensor([0.001, 0.02, 0.4]))
Tensor(shape=[3], dtype=float32, place=CPUPlace, stop_gradient=True,
      [0.28493962, 0.29040524, 0.42465511])
```

var=33773.56

var=0.03

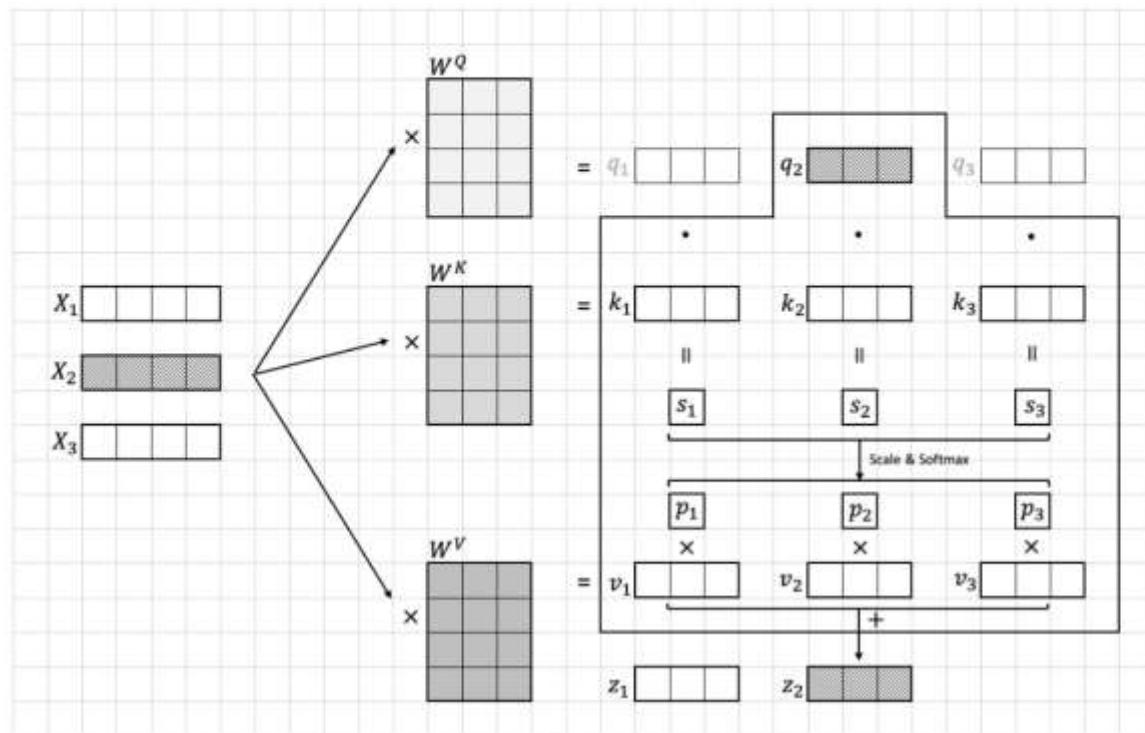
Self Attention : 对单个Token如何计算 ? (5)

分别计算每个token在整个序列中的Attention并生成特征



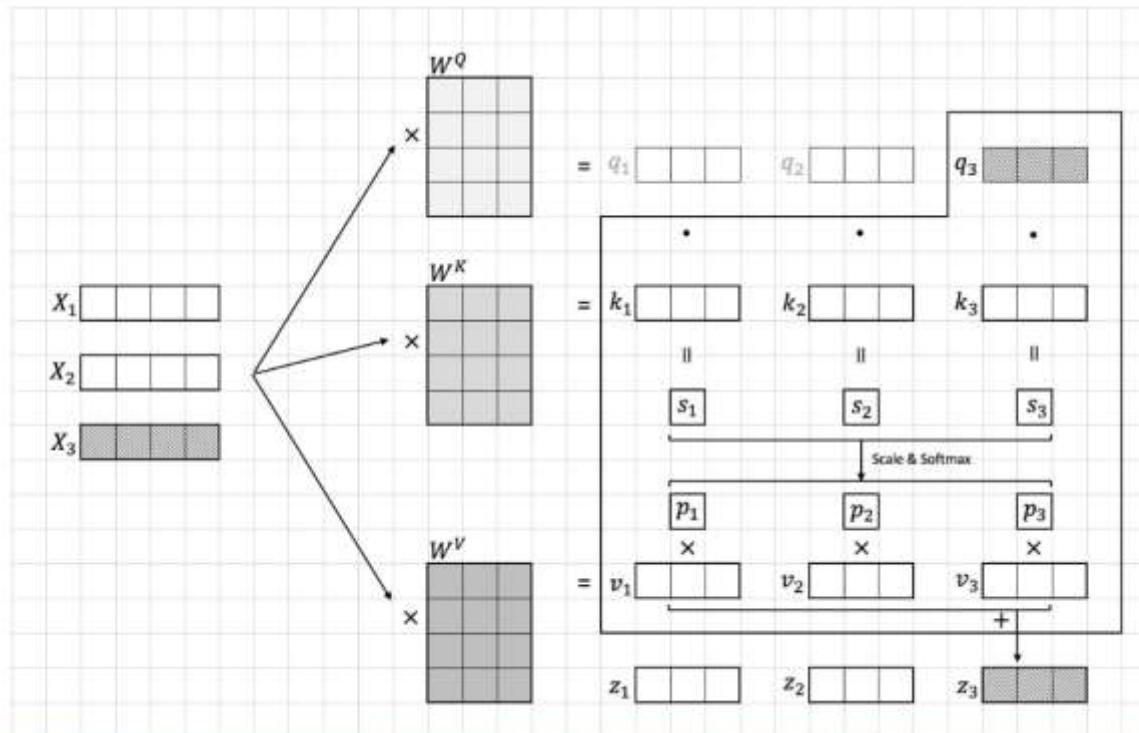
Self Attention : 对整个序列计算注意力

分别计算每个token在整个序列中的Attention并生成特征



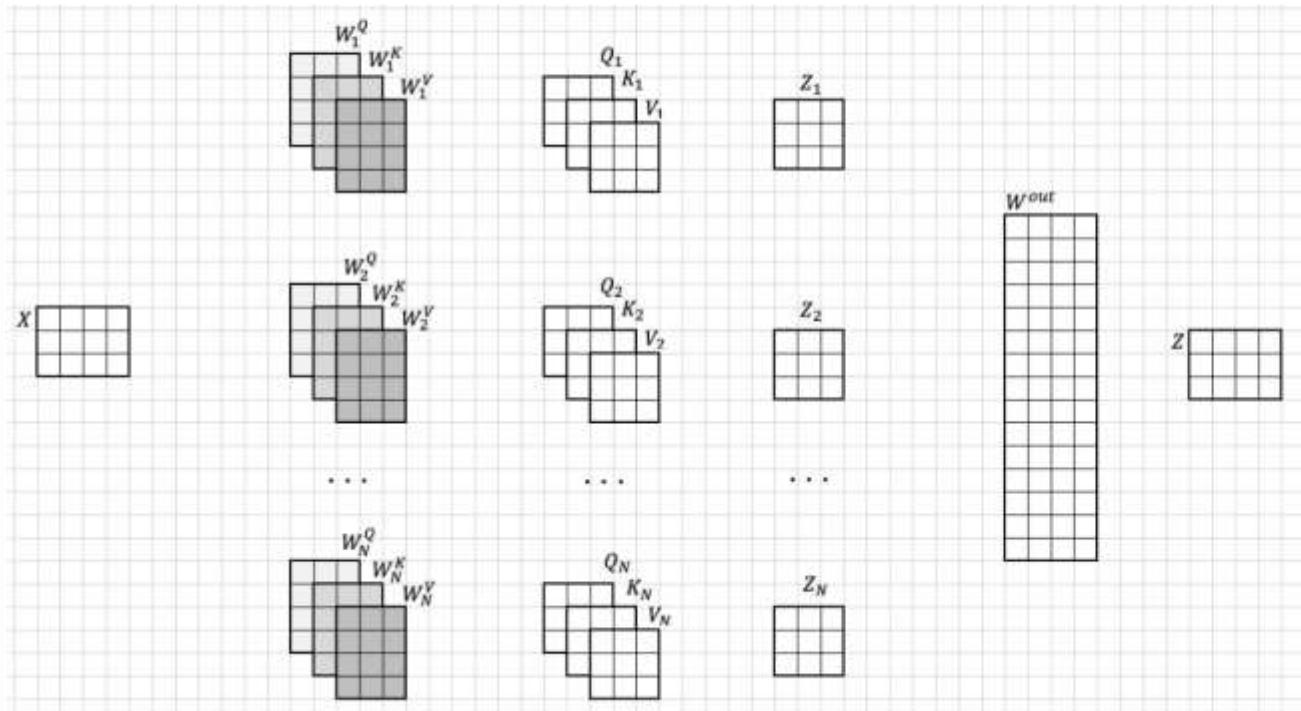
Self Attention : 对整个序列计算注意力(2)

分别计算每个token在整个序列中的Attention并生成特征



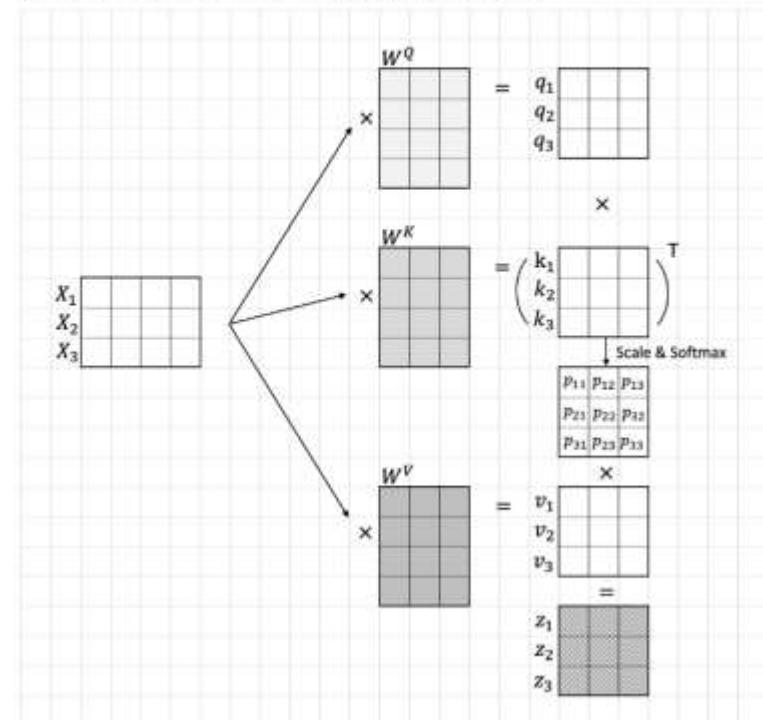
Self Attention : 对整个序列计算多头注意力

Multi-Head Self Attention



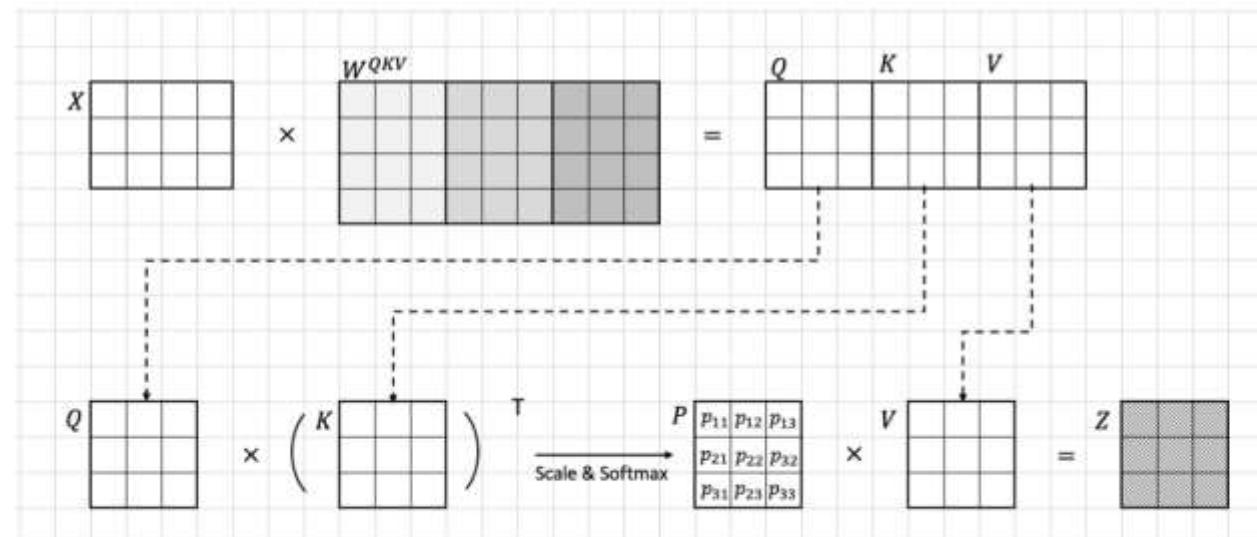
Self Attention : 注意力计算的矩阵表示

从单个Vector计算->使用矩阵计算



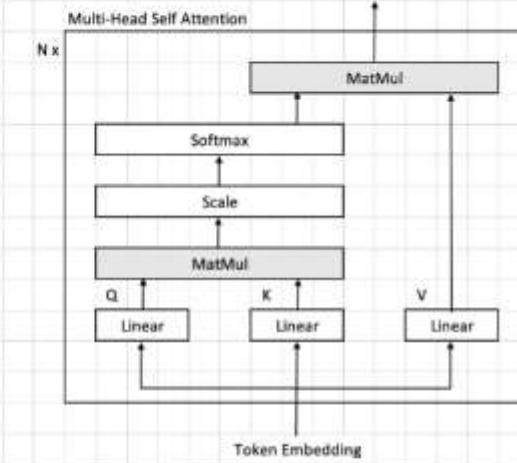
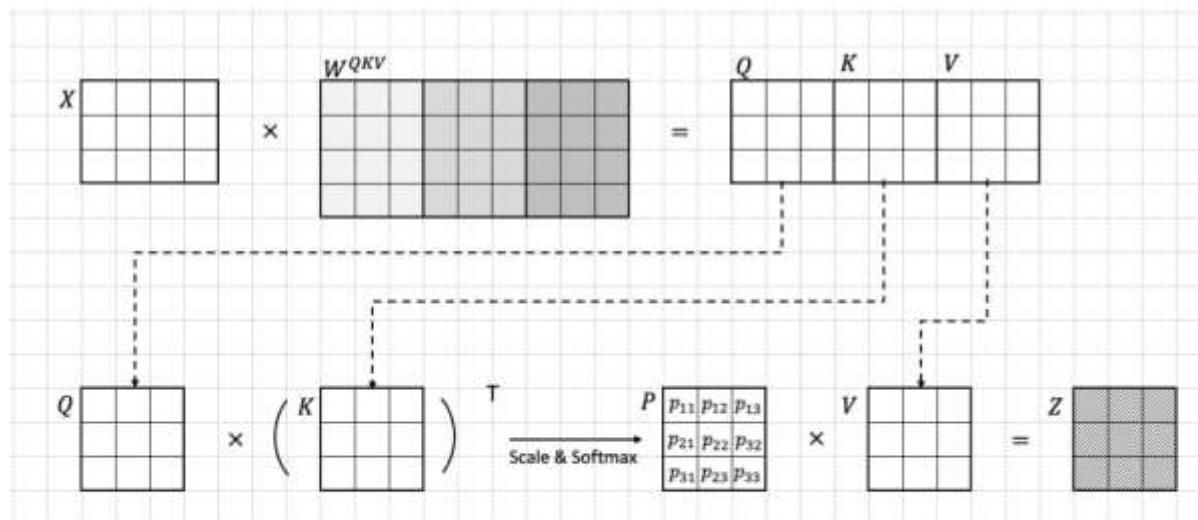
Self Attention : 注意力计算的矩阵表示(2)

从单个Vector计算->使用矩阵计算-> 合并QKV



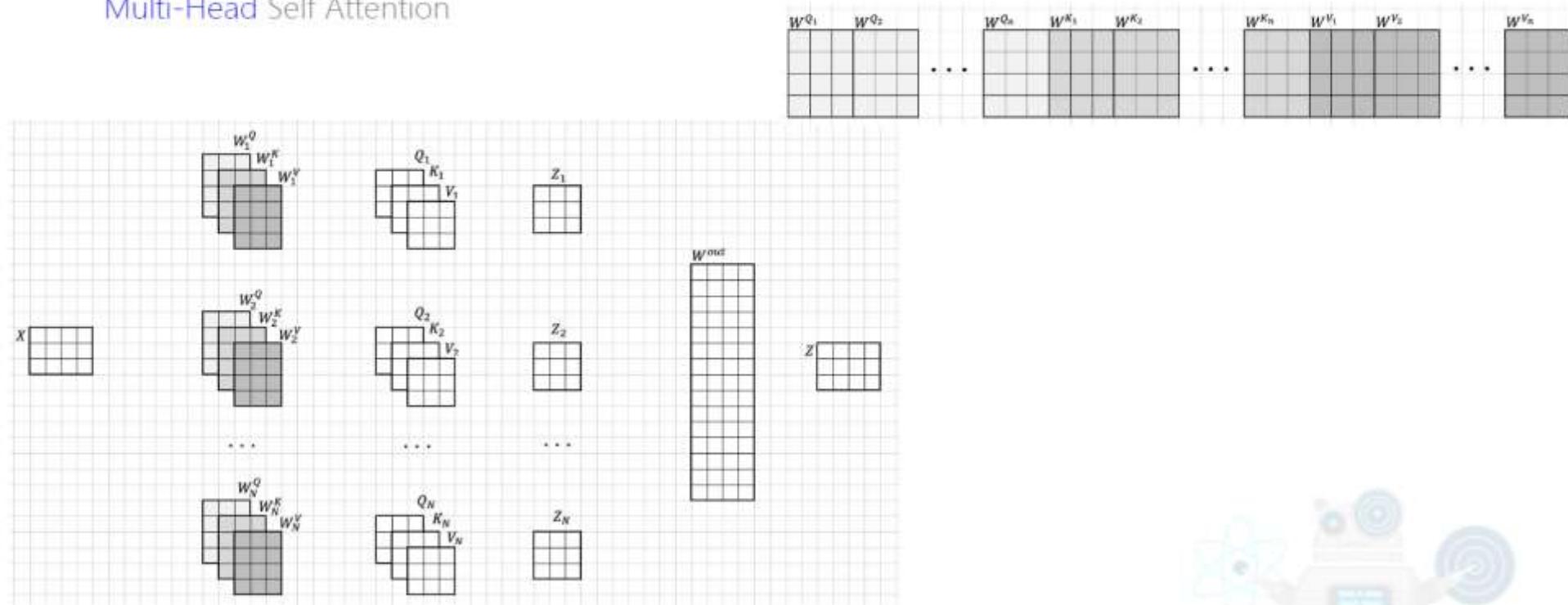
Attention实践：Multi-Head Self Attention

实现Multi-Head Self Attention



Attention实践：Multi-Head Self Attention

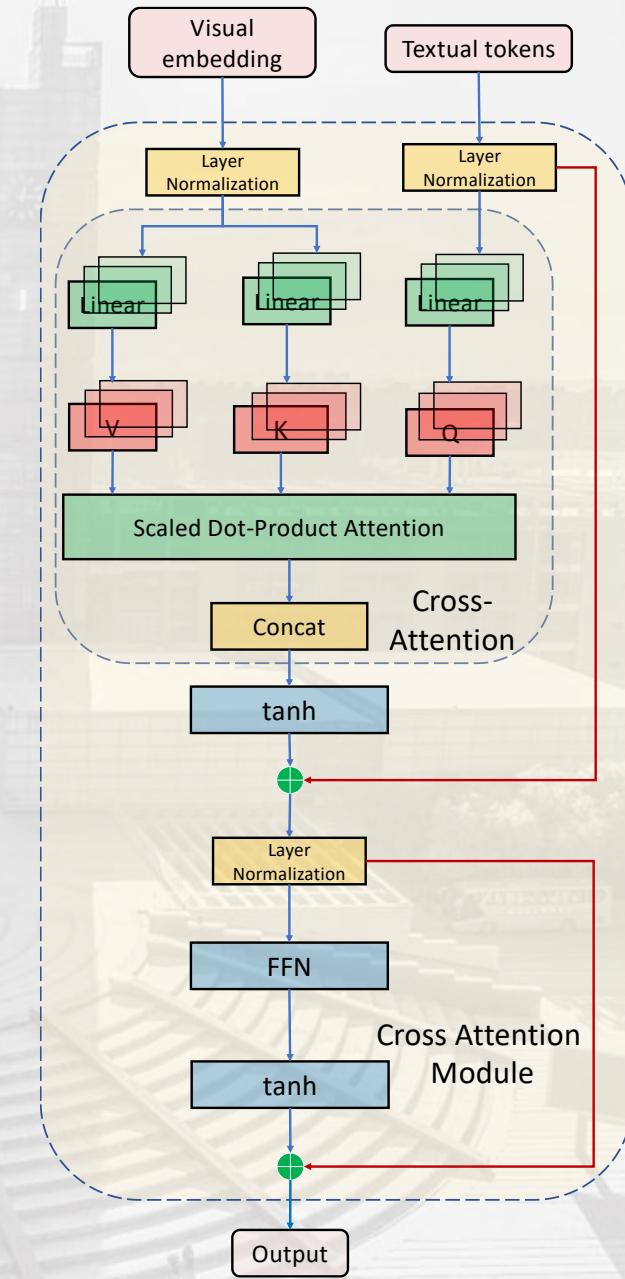
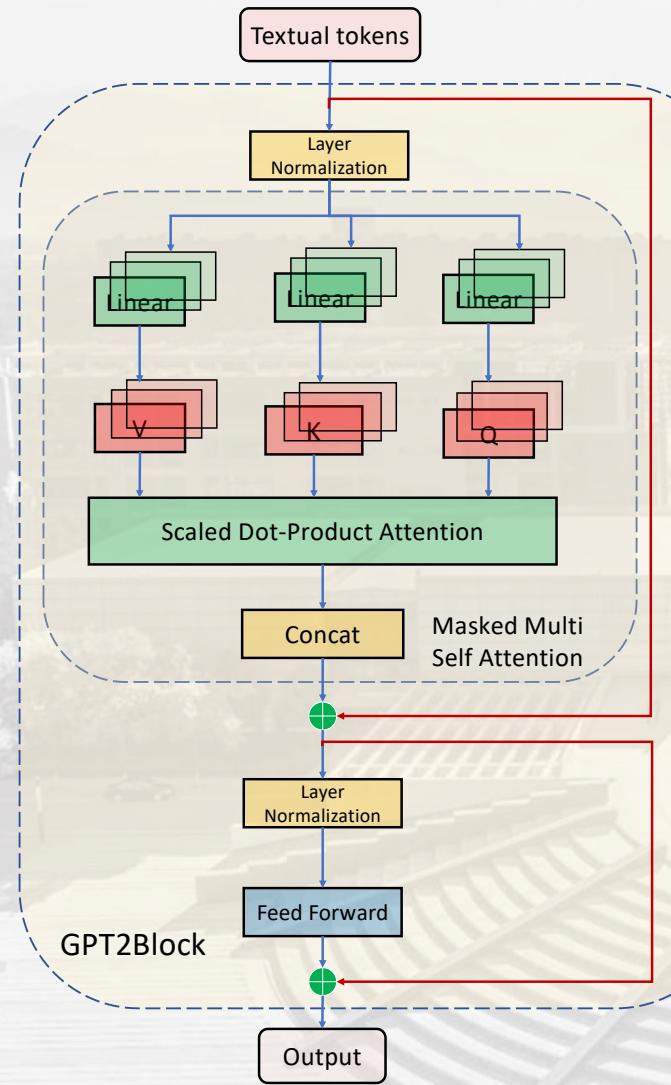
Multi-Head Self Attention



数据生成Narrator

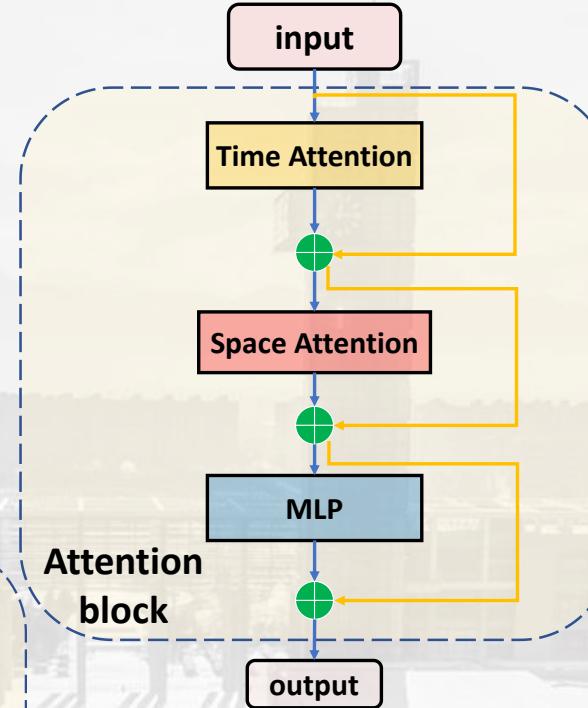
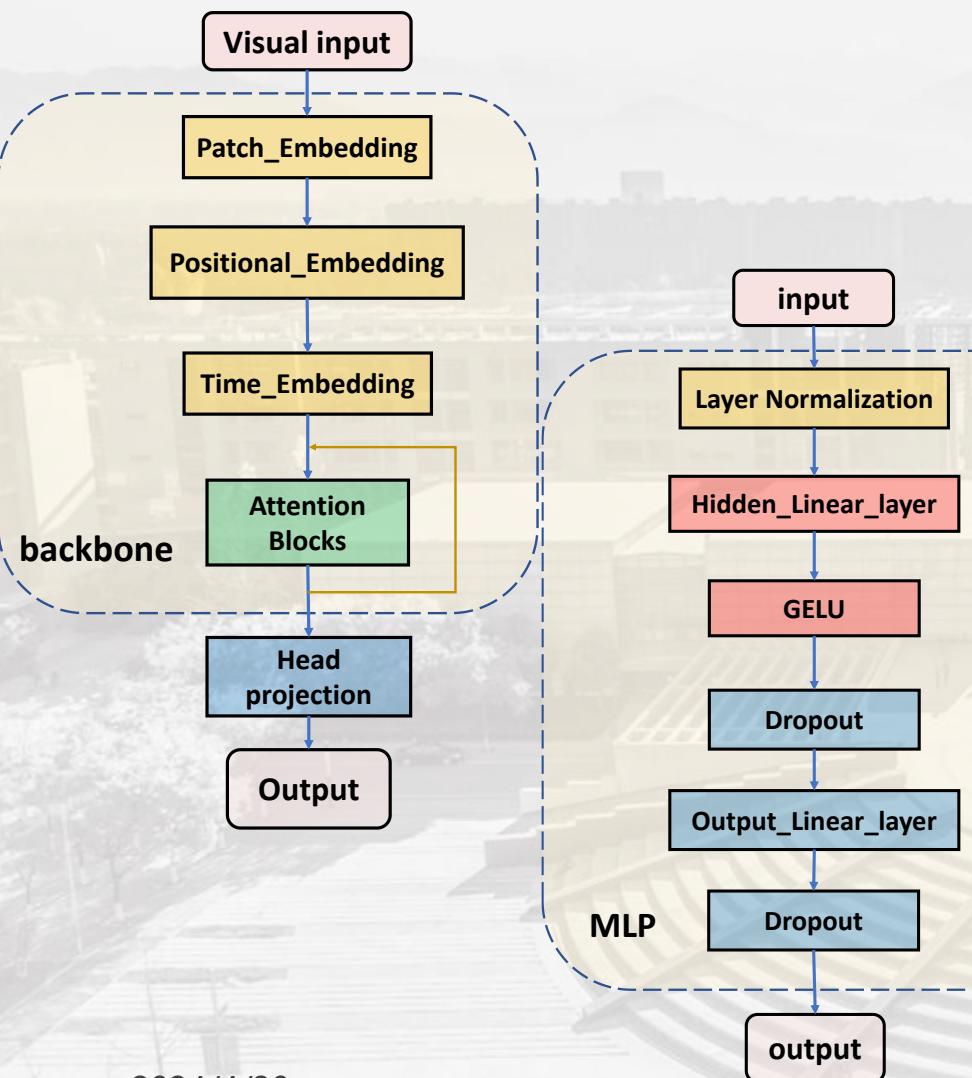
Narrator模型结构

XDU

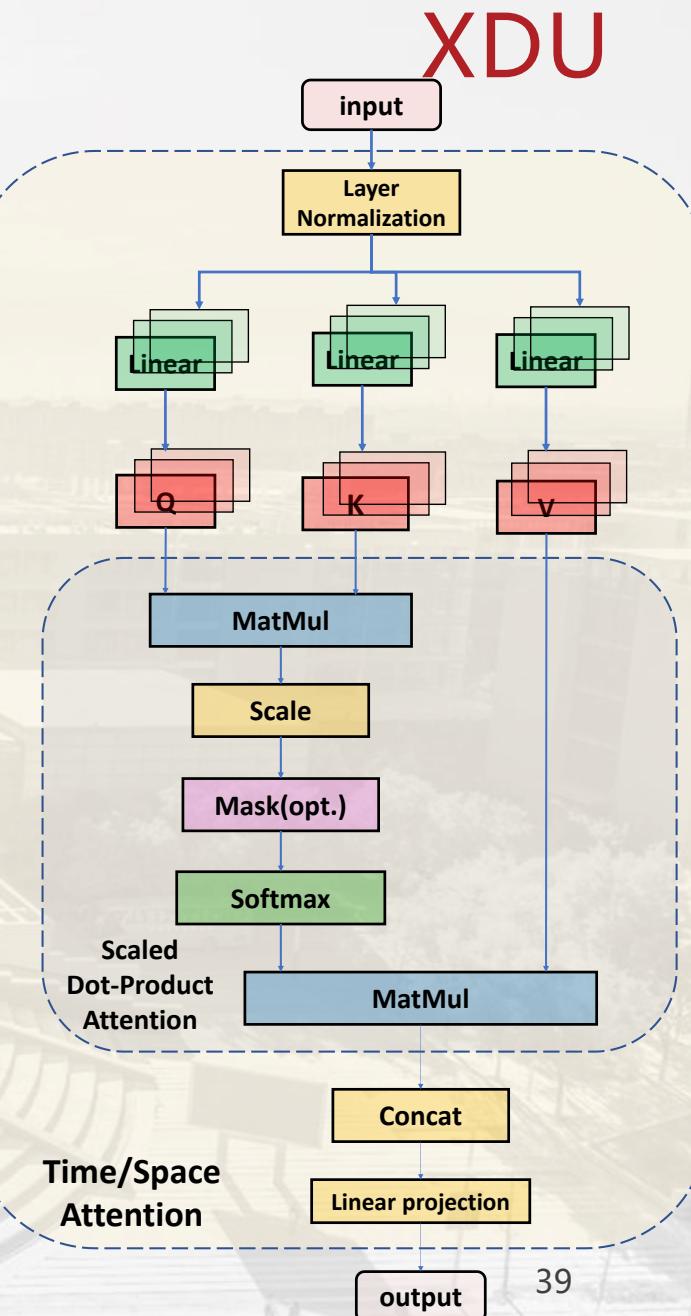


数据生成Narrator

Narrator模型结构



TimeSformer



Narrator密切遵循标准LLM的架构，只添加了一些额外的交叉注意模块来提供视觉调节。

叙述者能够从预先训练过的权重中初始化，这对我们的任务至关重要，因为我们用来训练叙述者（与视频剪辑相关的叙述）的数据比通常用于训练LLM的大型文本语料库在规模上要小得多。

具体来说，采用一个冻结的预先训练过的LLM，并在每个Decoder层之前添加一个交叉注意模块，以便文本输入能够关注视觉信息。交叉注意模块的输出然后通过残差连接与输入的文本特征求和，并进入下一个Decoder层。

每个交叉注意模块包括一个交叉注意层，它以文本标记作为查询Query，视觉嵌入作为键Key和值Value，然后是一个前馈网络（FFN）。

在交叉注意和FFN的开始阶段都应用了层归一化。我们添加了tanh-gating，初始值为零，这样新模型的输出与原始语言模型的输出相同。

我们在真实标记数据 $(\mathcal{X}, \mathcal{Y})$ 上训练Narrator。对于每一对 $(\mathcal{X}, \mathcal{Y})$, 损失是每一步中正确单词的负对数似然值的和

$$\mathcal{L}_{\text{NARRATOR}}(x, y) = - \sum_{\ell=1}^L \log p(s_\ell | s_{<\ell}, x)$$

Input: x (visual input) + 特定的句子开始的标记 $\langle s \rangle$

$$\tilde{s}_\ell \sim p(s | [\langle s \rangle, \dots, \tilde{s}_{\ell-1}], x)$$

我们递归地从上述分布中进行抽样，直到句子末尾的标记 $\langle /s \rangle$ 出现

在每一步中，我们从包含绝大多数概率质量的标记子集中采样，这被称为**核采样**。

1. 核采样 (nucleus sampling) 是一种文本生成方法，它在生成文本时不是选择概率最高的下一个词（如在束搜索中那样），而是根据累计概率分布选择下一个词。也就是说，它会选择累积概率超过某个阈值（例如0.95）的词作为下一个词。
2. 核采样与传统的最大似然方法（如束搜索）相比，能生成更多样化、开放式结尾且更类人的文本。这是因为核采样允许模型探索不同的词汇选择，而不是总是选择概率最高的词，从而生成更加丰富和自然的文本。
3. 然而，核采样也有其缺点，由于没有基于句子级似然进行后处理，生成的文本可能包含不相关或噪声信息。这是因为在选择下一个词时，核采样只考虑了词的概率分布，而没有考虑整个句子的连贯性和语义一致性。
4. 为了解决这个问题，作者对相同的视觉输入重复进行K次采样过程。也就是说，对于同一个输入图像，他们生成K个不同的描述，这样就可以从多个角度来描述图像，增加描述的多样性。
5. 通过使用核采样生成更多样化的叙述，可以提高模型的性能。这是因为多样化的叙述可以提供更多的信息，帮助模型更好地理解输入图像的内容，从而生成更准确和丰富的描述。

数据生成Narrator

XDU

Narrator模型推理——数据生成

- 对数据集 X 中已经标注的视频片段进行重新标注
- 对视频中未标注的部分进行伪标注

均匀采样未标注区间的片段：

在假设视频是一个平稳过程的基础上，我们从未标注的时间区间均匀地采样视频片段。

计算片段持续时间和采样步长：

采样的视频片段持续时间被设置为所有真实标注片段持续时间的平均值： $\Delta = \frac{1}{N} \sum_{i=1}^N (e_i - t_i)$

如果我们希望采样的片段之间没有重叠且紧密相连，那么采样步长应该等于片段的持续时间。

结合重新标注和伪标注的叙述，我们得到了NARRATOR生成的最终标注集合，记为 $(\mathcal{X}', \mathcal{Y}')$

长格式视频的稀疏叙述：长视频通常只有部分内容被标注，这意味着所有已标注片段的时间联合并不覆盖整个视频。

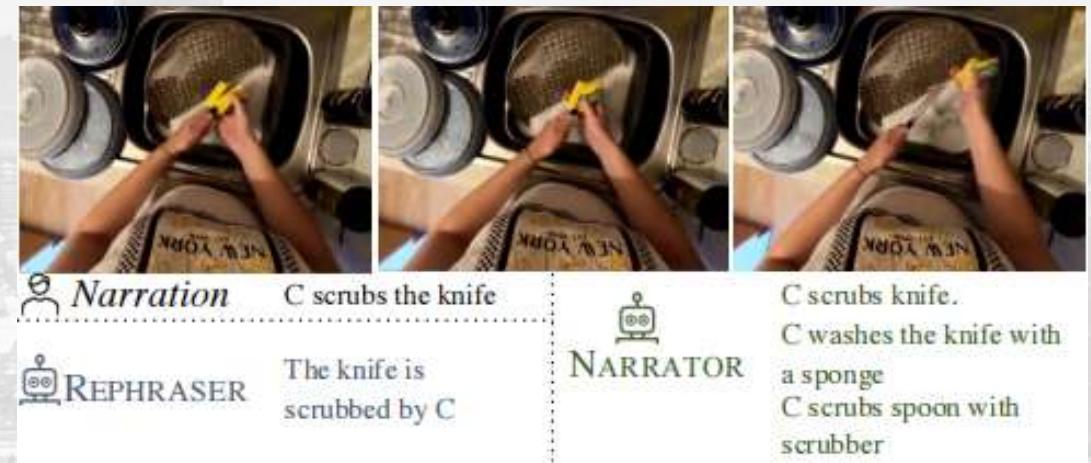
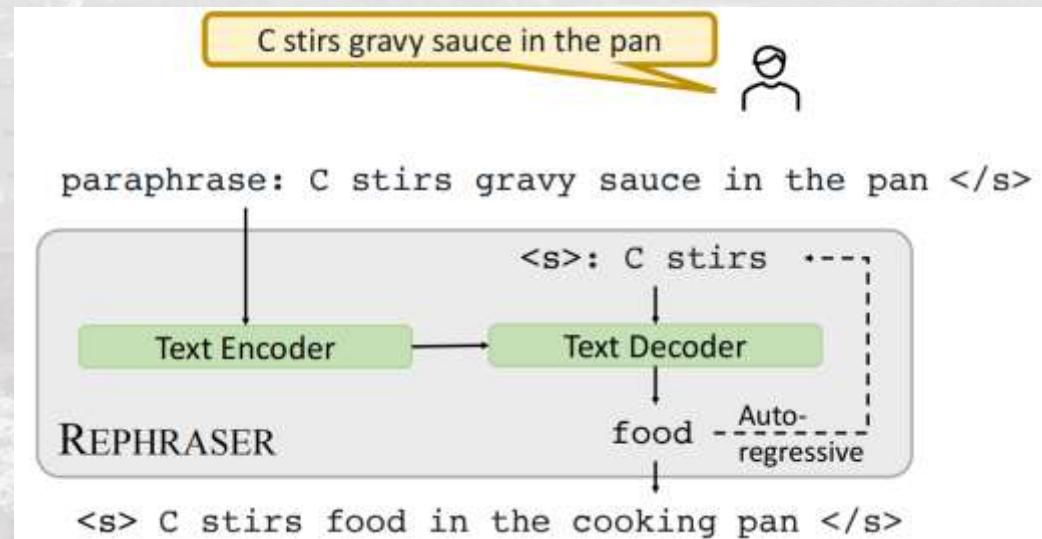
后处理。详尽的伪标注可能会包含一些不提供有用信息的视觉片段，并生成无用的文本。因此，我们增加了一个过滤过程来消除低质量的片段及其相关的描述。我们使用在真实配对片段上训练的基线双编码器模型 F ，来计算伪标签对的视觉和文本嵌入，并根据相似度分数进行过滤，即 $\text{Filter}(f_v(x'_j)^T \cdot f_t(y'_j))$ ，其中 $\text{Filter}(\cdot)$ 可以是所有生成文本的top-k或基于阈值的过滤。在实验中，我们使用0.5的阈值。

数据生成Rephraser

XDU

Rephraser的介绍

REPHRASER is a standard LLM that paraphrases narrations in existing clips, augmenting those annotations to $(\mathcal{X}, \mathcal{Y}')$



REPHRASER用于增强文本输入，例如，改变人类叙述的单词顺序，并额外替换常见的动词或名词，使注释更加多样化。

The data generated by NARRATOR is several times larger than the ground-truth pairs. To ensure that we do not overfit the pseudo-labeled data, we increase the number of ground-truth narrations by paraphrasing. In particular, we use a text-to-text LLM which models conditional text likelihood:

$$p_{\text{REPHRASER}}(y''|y) = \prod_{\ell=1}^L p(s''_\ell | s'_{<\ell}, y).$$

The text-to-text model is implemented by an encoder-decoder architecture, e.g. T5 [51], to auto-regressively generate a new sentence given the original one. We observe that REPHRASER is able to do basic manipulations such as replacing synonyms or changing word order, which serves as an efficient way of automatic data augmentation. The resulting annotations are referred to as $(\mathcal{X}, \mathcal{Y}'')$.

为了确保我们不会过度拟合伪标记数据，我们通过释义增加了真实叙述的数量。

Rephraser

文本到文本的模型：通过编码解码器架构实现

根据给定的原始tokens，自回归地生成一个新的句子

自动数据增强的有效方法，可以替换同义词或改变词序

生成的视频注释被称为 $(\mathcal{X}, \mathcal{Y}'')$

REPHRASER. We use an open-source paraphraser [23] based on T5-large [51]. It is pre-trained on C4 [51] and then finetuned on a cleaned subset of ParaNMT [74]. During inference, we use Diverse Beam Search [67] with group number the same as beam number ($G = B = 20$) and set the diversity penalty to be 0.7. We keep 3 candidates per sentence, remove punctuations, and do basic de-duplication.

对比预训练目标 (contrastive pre-training objective) 是一种在机器学习领域，特别是在自然语言处理 (NLP) 和计算机视觉 (CV) 中常用的训练策略。它的核心思想是通过学习数据中正例和负例之间的差异来提高模型的表示能力。在预训练阶段，对比目标被用来指导模型学习如何区分相似 (正例) 和不相似 (负例) 的数据样本。

在对比学习中，正例通常是指相似的样本对，例如，来自同一类别的两个样本，或者通过某种变换得到的同一图像的不同视图。而负例则是指不相似的样本对，例如，来自不同类别的样本。模型的训练目标是最小化正例之间的距离，同时最大化负例之间的距离。

在NLP中，对比预训练目标可以应用于各种任务，例如文本分类、机器翻译、文本生成等。例如，在文本分类任务中，可以将正例定义为来自同一类别的文本对，而负例定义为来自不同类别的文本对。通过这种方式，模型可以学习到如何区分不同类别的文本。

在CV中，对比预训练目标也经常用于图像识别、图像生成等任务。例如，在图像识别任务中，可以将正例定义为同一图像的不同增强视图，而负例定义为不同图像的视图。通过这种方式，模型可以学习到如何区分不同的图像。

总的来说，对比预训练目标是一种有效的训练策略，可以帮助模型学习到更鲁棒、更具有区分性的数据表示，从而提高模型的性能。

对比预训练是一种机器学习技术，旨在通过比较不同样本之间的差异来训练模型。这种方法的含义在于，它不是单独地处理每个样本，而是将样本成对地比较，从而让模型学会区分它们之间的差异。对比预训练的目标是让模型能够识别出相似的样本，并将它们与不相似的样本区分开来。

对比预训练的核心目标可以符号化表示为：给定一个样本集 $D = \{x_1, x_2, \dots, x_n\}$ ，我们希望模型学会一个函数 $f(x)$ ，这个函数能够将相似的样本映射到接近的表示，而不相似的样本映射到相隔较远的表示。

对比预训练的内容

对比预训练的内容通常包括以下步骤：

1. **样本选择**: 从数据集中选择一个正样本对 (x_i, x_j) ，它们应该是相似的，比如来自同一类别的不同样本。同时，选择一个负样本 x_k ，它应该与 x_i 和 x_j 都不相似。
2. **特征提取**: 使用一个编码器 E 来提取每个样本的特征表示。对于正样本对，我们希望它们的特征表示接近；对于负样本，我们希望它的特征表示与正样本对相隔较远。
 - $h_i = E(x_i)$
 - $h_j = E(x_j)$
 - $h_k = E(x_k)$
3. **对比损失函数**: 设计一个对比损失函数 L 来衡量模型的表现。常见的对比损失函数有InfoNCE损失和Triplet损失。
 - InfoNCE损失:
$$L = -\log \frac{\exp(h_i \cdot h_j / \tau)}{\exp(h_i \cdot h_j / \tau) + \sum_{x_l \in D_{neg}} \exp(h_i \cdot h_l / \tau)}$$
 - Triplet损失:
$$L = \max(|h_i - h_j|^2 - |h_i - h_k|^2 + \alpha, 0)$$
其中， τ 是温度参数，用于调节相似度的敏感度； α 是边际参数，用于控制正负样本之间的间隔； D_{neg} 是负样本集合。
4. **模型优化**: 通过最小化损失函数 (L) 来更新模型参数。

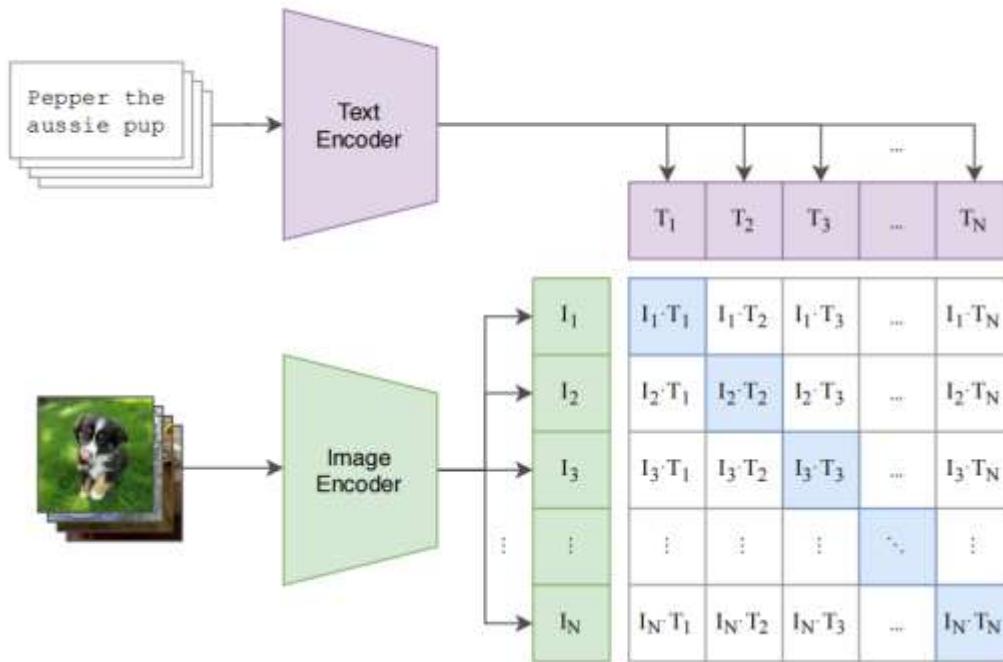
对比预训练的原理基于一个简单的概念：。这种方法的关键在于如果一个模型能够学会区分相似的样本和不相似的样本，那么它就能在更复杂的学习任务中表现良好，**它不依赖于标签信息（无监督学习），而是通过比较样本之间的相似性来学习。**

通过这种方式，模型可以学习到数据集中存在的一些基础结构和模式，这些结构和模式对于后续的下游任务（如分类、检测等）是非常有用的。对比预训练使得模型能够在**没有大量标注数据的情况下，也能学到有效的特征表示**，从而提高模型在各种任务上的表现。

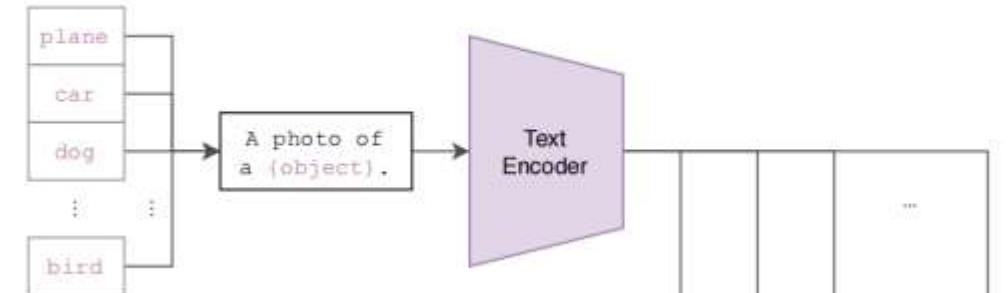
总结来说，对比预训练是一种强大的**无监督学习方法**，它通过比较样本之间的相似性来训练模型，使模型能够学会区分不同类型的样本。这种方法在许多机器学习任务中都显示出了优异的性能。

对比预训练过程——CLIP

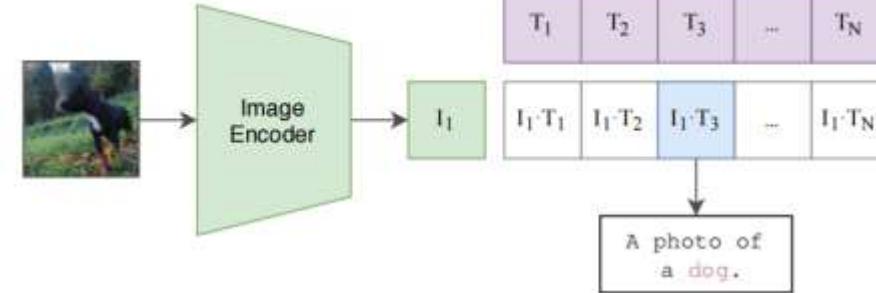
(1) Contrastive pre-training



(2) Create dataset classifier from label text



(3) Use for zero-shot prediction



CLIP (Contrastive Language-Image Pre-training) 是由OpenAI提出的一种多模态预训练模型，它通过大量的图像-文本对进行预训练，以学习图像和文本之间的关联。CLIP的核心思想是将图像编码器和文本编码器映射到同一嵌入空间中，使得图像和其对应的描述在该空间中具有相似的嵌入向量。

对比预训练过程——CLIP

在CLIP的预训练过程中，对比预训练目标是关键组成部分，其工作原理如下：

- 1. 数据准备：** CLIP的预训练数据集由大量的图像-文本对组成，每个图像都有一个或多个描述它的文本。这些图像和文本对可以是来自互联网的各种数据，例如社交媒体、网页等。
- 2. 编码器：** CLIP包含两个主要的组件，一个是图像编码器，用于将图像转换为特征向量；另一个是文本编码器，用于将文本描述转换为特征向量。这两个编码器可以是任意的神经网络模型，例如卷积神经网络（CNN）用于图像编码器和Transformer模型用于文本编码器。
- 3. 嵌入空间：** 图像编码器和文本编码器都设计为将输入数据映射到同一嵌入空间中。在这个空间中，图像和文本的嵌入向量具有相同的维度。
- 4. 对比目标：** CLIP的对比预训练目标是最大化图像和其对应文本嵌入向量之间的余弦相似度，同时最小化图像和非对应文本嵌入向量之间的余弦相似度。 具体来说，对于每个图像，它的正例是与之配对的文本描述，而负例是其他图像的文本描述。
- 5. 信息检索视角：** 可以将CLIP的预训练过程看作是一个多模态的信息检索任务，其中图像和文本嵌入向量被训练为在检索过程中能够相互匹配。
- 6. 损失函数：** CLIP使用了一种称为InfoNCE损失（或称对比损失）的函数来计算图像和文本嵌入向量之间的对比目标。InfoNCE损失是一种交叉熵损失，用于衡量模型在区分正例和负例方面的性能。

通过这种方式，CLIP在预训练过程中学习到了**图像和文本之间的关联**，使得模型能够**在没有大量标记数据的情况下，对新的图像进行zero-shot分类**。这意味着CLIP能够使用预训练的模型来识别和分类它在预训练阶段没有直接见过的对象类别，只需提供相关的文本描述即可。这种**zero-shot**学习能力是CLIP的一个重要特性，也是它在多种视觉任务中表现出色的原因之一。

实例：

假设我们有一个图像，它是一只猫的图片，以及与之配对的文本描述“一只可爱的猫”。在CLIP的训练过程中，图像编码器会生成猫图片的嵌入向量 z ，文本编码器会生成描述“一只可爱的猫”的嵌入向量 y 。

在计算损失时，我们会将 z 与 y 之间的相似度与其他描述（如“一只狗在玩耍”、“一个孩子在微笑”等）的嵌入向量之间的相似度进行比较。我们的目标是让 z 与 y 之间的相似度尽可能高，而与其他描述的嵌入向量之间的相似度尽可能低。

通过这种方式，CLIP学习到了如何将图像和文本嵌入到同一空间中，并使得匹配的图像和文本在该空间中靠近，而不匹配的图像和文本相隔较远。这种能力使得CLIP能够在zero-shot分类等任务中表现出色。

对比预训练损失函数——InfoNCE

介绍：

InfoNCE损失函数，也称为噪声对比估计（Noise Contrastive Estimation, NCE）损失，是一种用于对比学习的损失函数，它来源于噪声对比估计（Noise Contrastive Estimation, NCE）和互信息估计（Mutual Information Estimation）。

它的核心思想是通过比较正例和负例之间的相似度来训练模型，使得模型能够区分它们。InfoNCE损失函数在多模态学习、自监督学习、生成模型等领域都有广泛的应用。

在 CLIP模型中，InfoNCE 损失函数用于比较图像嵌入和文本嵌入之间的相似度，以实现图像和文本的联合嵌入。

CLIP 模型通常包含两个部分：图像编码器 (E_{img}) 和文本编码器 (E_{txt})

CLIP 模型通常包含两个部分：

图像编码器 (E_{img})

文本编码器 (E_{txt})

每一批数据都包含成对的图像和文本，其中每个图像都有一个对应的正确描述作为正样本，其他图像的描述作为负样本。模型的目标是学会区分正确的图像-文本对和错误的图像-文本对。

在 CLIP 中，InfoNCE 损失函数的表达式实际上是由两部分组成的，一部分是**图像到文本的损失**，另一部分是**文本到图像的损失**。这两部分损失是相互对称的，它们共同确保了图像和文本嵌入能够在同一空间中正确地对齐。

相似度计算：

对于每个图像嵌入 v 和文本嵌入 u ，它们之间的相似度通过以下方式计算：

$$s(v, u) = v \cdot u / \tau$$

图像到文本的 InfoNCE 损失：

$$L_{img2txt} = \frac{1}{N} \sum_i \left(-\log \frac{e^{s(v_i, u_i^+)/\tau}}{\sum_{j=0}^{N-1} e^{s(v_i, u_j)/\tau}} \right)$$

其中：

- v_i 是第 i 个图像的嵌入。
- u_i^+ 是与 v_i 对应的正确文本嵌入（正样本）。
- u_j 是其他文本嵌入（包括正样本和 $N - 1$ 个负样本）。
- $s(v_i, u_j)$ 是图像嵌入 v_i 和文本嵌入 u_j 之间的相似度，通常通过点积计算。
- τ 是温度参数，用于调节相似度的敏感度。
- N 是每批数据中图像的数量。

文本到图像的 InfoNCE 损失：

$$L_{txt2img} = \frac{1}{N} \sum_i \left(-\log \frac{e^{s(u_i^+, v_i)/\tau}}{\sum_{j=0}^{N-1} e^{s(u_i, v_j)/\tau}} \right)$$

其中：

- u_i^+ 是第 i 个文本的嵌入。
- v_i 是与 u_i^+ 对应的正确图像嵌入（正样本）。
- u_j 是其他图像嵌入（包括正样本和 $N - 1$ 个负样本）。
- $s(u_i, v_j)$ 是文本嵌入 u_i 和图像嵌入 v_j 之间的相似度，通常通过点积计算。
- τ 是温度参数，用于调节相似度的敏感度。
- N 是每批数据中图像的数量。

对比预训练损失函数——InfoNCE

总损失：

最终，CLIP 模型的总损失是图像到文本和文本到图像的 InfoNCE 损失的加权组合，即 InfoNCE 损失函数用于计算图像嵌入和文本嵌入之间的对比损失：

$$L_{CLIP} = L_{img2txt} + L_{txt2img}$$

通过最小化这个损失函数，CLIP 模型学会了将图像和文本嵌入到同一个连续的语义空间中，使得具有相同语义的图像和文本在该空间中靠近，而不相似的图像和文本相隔较远。这使得 CLIP 能够在 **没有直接监督的情况下，学会图像和文本之间的关联。**

我们会对所有图像嵌入的损失进行平均，对所有文本嵌入的损失进行平均。这样，每个样本的损失都平等地贡献到了总的损失中，确保了模型在训练过程中对所有图像和文本对都进行了平等的考虑。

对比预训练损失函数——InfoNCE

温度参数 τ :

温度参数 τ 在 InfoNCE 损失函数中起着关键作用。它是一个超参数，用于控制相似度的软最大化过程。具体来说， τ 影响着模型对正负样本之间相似度差异的敏感度：

- 当 τ 较大时，函数的梯度会**更加平滑（梯度较小）**，模型对噪声的鲁棒性更强，但可能导致模型对正负样本的**区分度降低**，这有助于模型学习更平滑的决策边界。
- 当 τ 较小时，函数的梯度会**更加尖锐（梯度较大）**，模型能够**更明确地区分正负样本**，但可能对噪声更敏感，这有助于模型学习更明确的决策边界。

温度参数 τ 的选择通常通过实验来确定，常见的选择范围在 $(0.01, 1)$ 之间。在实际应用中， τ 的最佳值可能因具体任务和数据集而异，需要通过验证集上的性能来调整。例如，在CLIP模型中，OpenAI的研究者发现使用温度参数 $\tau = 0.07$ 能够取得较好的性能。

总之，温度参数 τ 在InfoNCE损失函数中起着调节模型学习过程的作用，它通过调整模型对正例和负例之间相似度的敏感性来影响模型的学习效果。

We train the dual-encoders as described in Algorithm 1 in Appendix E. In each iteration, we first sample a batch \mathcal{B} of video clips. It comprises a subset of clips \mathcal{B}_l with labeled timestamps as well as narrations, and a subset \mathcal{B}_u whose clips are randomly sampled from videos without narrations. For clip $x_i \in \mathcal{B}_u$, we obtain the pseudo-caption y'_i by querying the NARRATOR $y'_i \sim p_{\text{NARRATOR}}(y'|x)$, resulting in a set of clips with LLM-generated narrations $\tilde{\mathcal{B}}_u$. For clip $(x_i, y_i) \in \mathcal{B}_l$, the text supervision is obtained from either the REPHRASER or the NARRATOR, with a probability of 0.5. We denote the resulting set of pairs to be \mathcal{B}_l similarly. Following CLIP [49], we use the symmetric cross-entropy loss over the similarity scores of samples in the batch $\tilde{\mathcal{B}}_l \cup \tilde{\mathcal{B}}_u$.

In practice, we run REPHRASER and NARRATOR in advance and cache the resulting video-narration pairs so that there is no computational overhead during pre-training.

Algorithm 1 One step of training LAVILA

Require: A subset of narrated (unnarrated) clips \mathcal{B}_l (\mathcal{B}_u)
 clips with LM-generated narrations $\tilde{\mathcal{B}}_l = \{\}$, $\tilde{\mathcal{B}}_u = \{\}$

for $(x_i, y_i) \in \mathcal{B}_l$ **do**

- $u \sim U(0, 1)$ \triangleright Uniform sample between 0 and 1
- if** $u < 0.5$ **then** \triangleright Query REPHRASER

 - $y'_i \sim p_{\text{REPHRASER}}(y'|y_i)$, $\tau_i \leftarrow \tau_r$

- else** \triangleright Query NARRATOR

 - $y'_i \sim p_{\text{NARRATOR}}(y'|x_i)$, $\tau_i \leftarrow \tau_n$

end if

$\tilde{\mathcal{B}}_l \leftarrow \tilde{\mathcal{B}}_l \cup \{(x_i, y'_i, \tau_i)\}$

end for

for $x_i \in \mathcal{B}_u$ **do**

- $y'_i \sim p_{\text{NARRATOR}}(y'|x_i)$, $\tau_j \leftarrow \tau_n$
- $\tilde{\mathcal{B}}_u \leftarrow \tilde{\mathcal{B}}_u \cup \{(x_j, y'_j, \tau_j)\}$

end for

Train $\mathcal{F}_{\text{LAVILA}}(x, y)$ with the batch $\tilde{\mathcal{B}}_l \cup \tilde{\mathcal{B}}_u$ using Eq 4.

训练LAVILA的算法如右图所示。损失是在样本的相似度分数上，**基于CLIP的对称交叉熵损失**。其中，对于文本重述和图像伪标记两种数据，采用两个独立的温度参数 (τ_r, τ_n) ：

$$\mathcal{L} = -\frac{1}{2N} \sum_{i=1}^N \left(\log \frac{\exp\left(\frac{\mathbf{v}_i^\top \mathbf{u}_i}{\tau_i}\right)}{\sum_{j=1}^N \exp\left(\frac{\mathbf{v}_i^\top \mathbf{u}_j}{\sqrt{\tau_i \tau_j}}\right)} + \log \frac{\exp\left(\frac{\mathbf{u}_i^\top \mathbf{v}_i}{\tau_i}\right)}{\sum_{j=1}^N \exp\left(\frac{\mathbf{u}_i^\top \mathbf{v}_j}{\sqrt{\tau_i \tau_j}}\right)} \right)$$

Algorithm 1 One step of training LAVILA

Require: A subset of narrated (unnarrated) clips \mathcal{B}_l (\mathcal{B}_u)
clips with LM-generated narrations $\tilde{\mathcal{B}}_l = \{\}$, $\tilde{\mathcal{B}}_u = \{\}$

for $(x_i, y_i) \in \mathcal{B}_l$ **do**

- $u \sim U(0, 1)$ \triangleright Uniform sample between 0 and 1
- if** $u < 0.5$ **then** \triangleright Query REPHRASER
- $y'_i \sim p_{\text{REPHRASER}}(y' | y_i), \tau_i \leftarrow \tau_r$
- else** \triangleright Query NARRATOR
- $y'_i \sim p_{\text{NARRATOR}}(y' | x_i), \tau_i \leftarrow \tau_n$
- end if**
- $\tilde{\mathcal{B}}_l \leftarrow \tilde{\mathcal{B}}_l \cup \{(x_i, y'_i, \tau_i)\}$

end for

for $x_i \in \mathcal{B}_u$ **do**

- $y'_j \sim p_{\text{NARRATOR}}(y' | x_i), \tau_j \leftarrow \tau_n$
- $\tilde{\mathcal{B}}_u \leftarrow \tilde{\mathcal{B}}_u \cup \{(x_j, y'_j, \tau_j)\}$

end for

Train $\mathcal{F}_{\text{LAVILA}}(x, y)$ with the batch $\tilde{\mathcal{B}}_l \cup \tilde{\mathcal{B}}_u$ using Eq 4.

Step1: 数据准备

训练LAVILA的算法如右图所示。损失是在样本的相似度分数上，**基于CLIP的对称交叉熵损失**。其中，对于文本重述和图像伪标记两种数据，采用两个独立的温度参数 (τ_r, τ_n) ：

$$\mathcal{L} = -\frac{1}{2N} \sum_{i=1}^N \left(\log \frac{\exp\left(\frac{\mathbf{v}_i^\top \mathbf{u}_i}{\tau_i}\right)}{\sum_{j=1}^N \exp\left(\frac{\mathbf{v}_i^\top \mathbf{u}_j}{\sqrt{\tau_i \tau_j}}\right)} + \log \frac{\exp\left(\frac{\mathbf{u}_i^\top \mathbf{v}_i}{\tau_i}\right)}{\sum_{j=1}^N \exp\left(\frac{\mathbf{u}_i^\top \mathbf{v}_j}{\sqrt{\tau_i \tau_j}}\right)} \right)$$

涉及到编码器模型的初始化

在一批数据 $\tilde{\mathcal{B}}_l \cup \tilde{\mathcal{B}}_u$ 上，设模型为 $\mathcal{F}_{\text{LAVILA}}(x, y)$
其中，

x 为视觉输入，即视频片段

y 为文本输入，即视频的文本描述

视觉编码器 (E_{img}) : TimeSformer(TSF)

文本编码器 (E_{txt}) : Transformer

$$v_i = E_{img}(x_i)$$

$$u_i = E_{txt}(y_i)$$

Step2: 使用视觉和文本编码器分别对视频和文本描述进行编码（即特征提取，得到特征向量）

Step3: 利用InfoNCE计算损失

Step4: 根据损失对两个编码器进行参数更新（微调）

Dual-Encoder Architecture. The video-language model follows a dual-encoder architecture as CLIP [49]. The Visual encoder is a TimeSformer (TSF) [6], whose spatial attention modules are initialized from a ViT [18] which is contrastively pre-trained on large-scale paired image-text data as in CLIP [49]. We sample 4 frames per clip during pre-training and 16 when finetuning on downstream tasks. The text encoder is a 12-layer Transformer [50, 66]. We use BPE tokenizer [55] to pre-process the full sentence corresponding to the video clip and keep at most 77 tokens.

视觉编码器 (E_{img}) : TimeSformer(TSF)

文本编码器 (E_{txt}) : Transformer

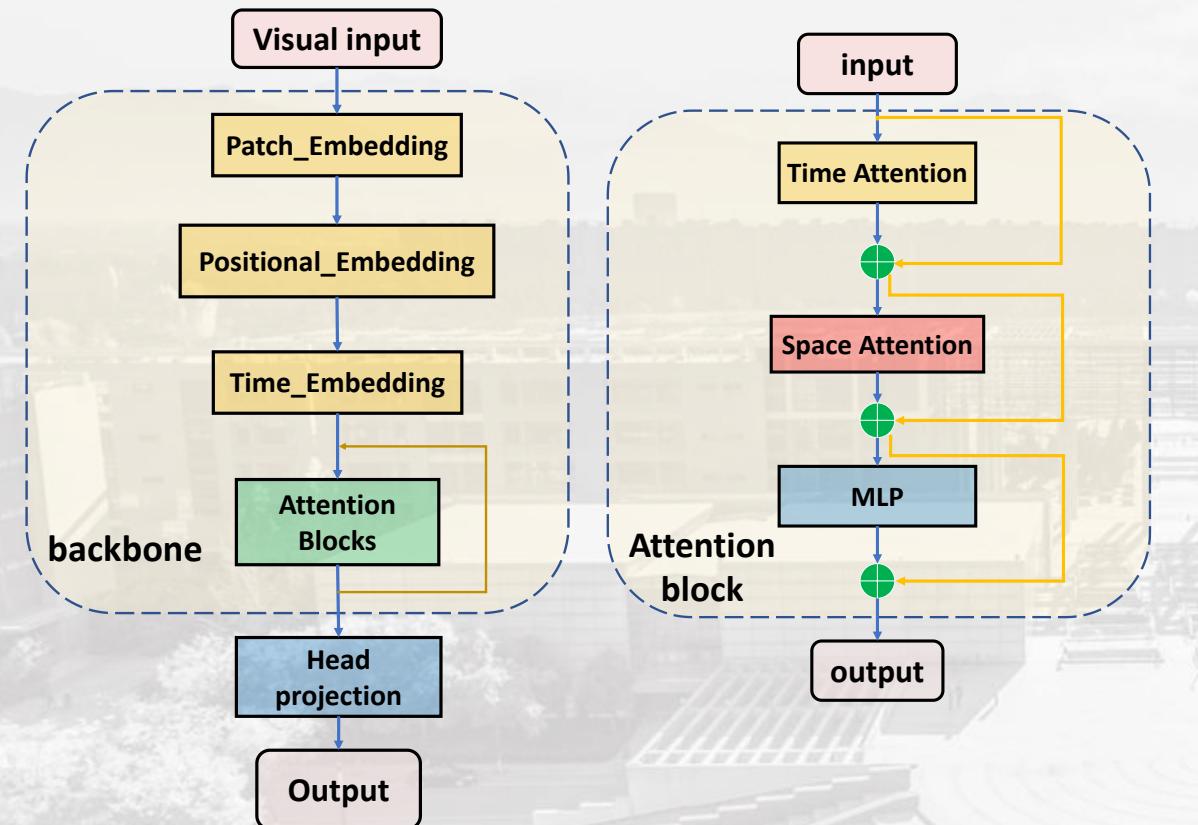
TimeSformer的**空间注意力模块**由ViT预训练的模型参数进行初始化，其中ViT是在大规模的**图像文本对**上进行对比预训练的模型。

文本编码器采用一个12层的Transformer模型。

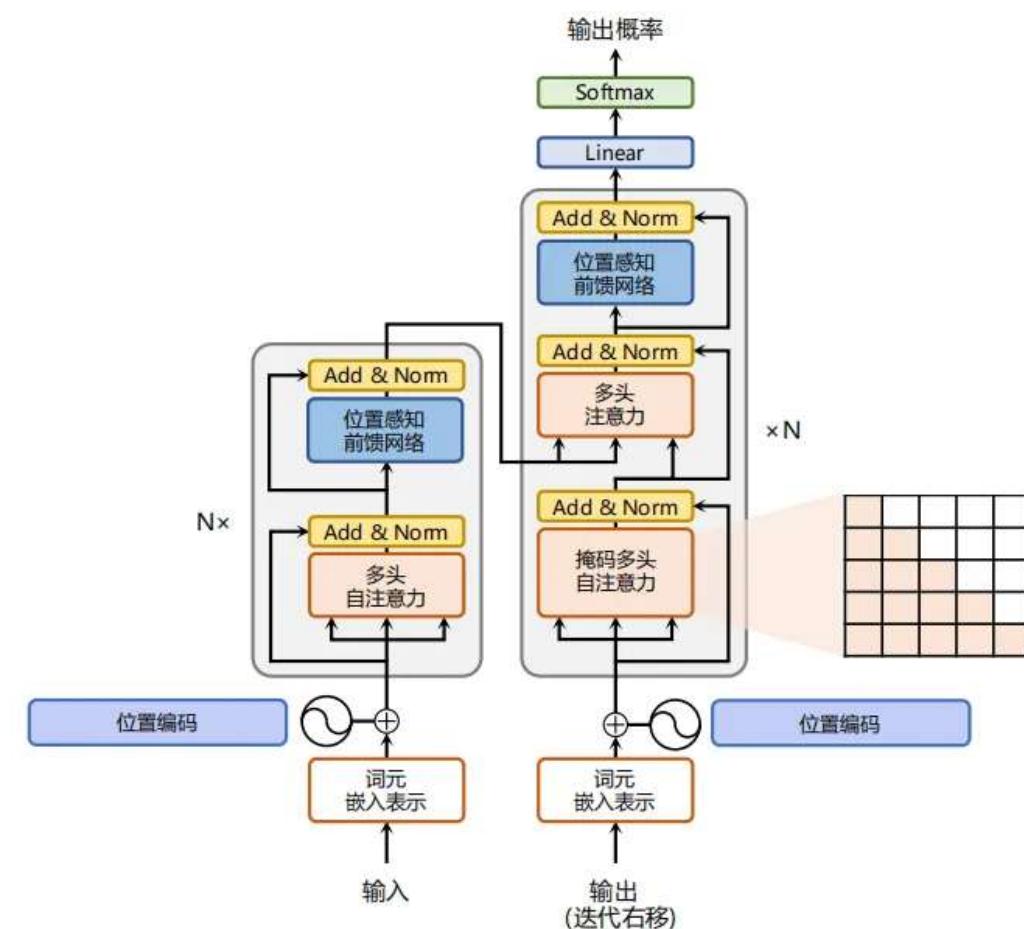
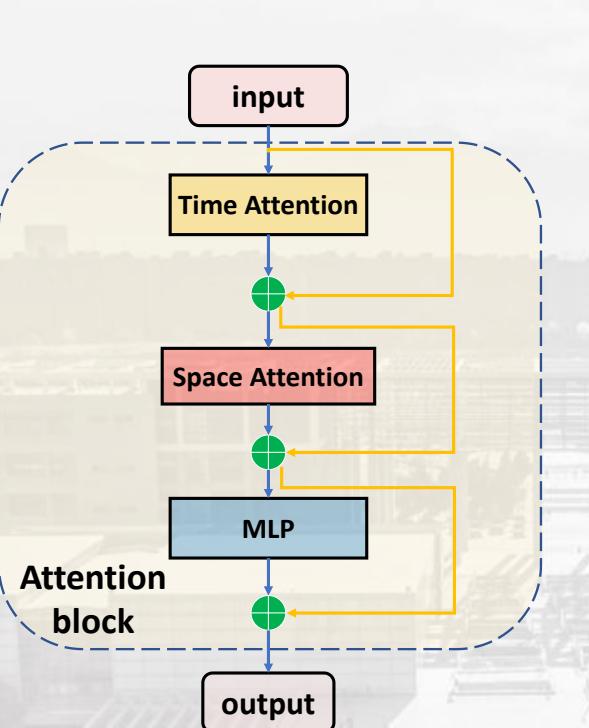
视觉语言模型

双向编码器的训练——训练全流程

XDU



视觉编码器



文本编码器

4Part Four

实验设计

介绍论文中实验的数据集、设计方案和实验结果，突出本篇论文的创新点

双向编码器的预训练：

首先，对Ego4D数据集的视频+叙述对进行预训练。在双向编码器后的线性映射是一个输出维度为256的线性层（以便后续视觉特征向量和文本特征向量进行相似度计算）

NARRATOR的训练：

对于视频编码器，**使用双向编码器预训练后的视频编码器模块**（即TimeSformer），并保持其冻结。丢弃TSF中的全局平均池化层，并附加了一个注意池化模块，该模块由一个标准的交叉注意和一个层归一化构成。然后，在Ego4D数据集上进行训练。

NARRATOR使用了双向编码器预训练模型中的TimeSformer模型参数，并使他们冻结保持不变。然而，作者丢弃了全局平均池化层，而是连接了一个注意力池化层，由一个标准的交叉注意力和一个层归一化构成。对于一个视频样本，在通过TimeSformer后，会在最终的线性映射之前，对每个视频的T帧的visual tokens（提取出的特征，每个视频的每帧都有N个Patches，对N个Patches在时间维度上）进行全局平均池化，然而，作者并没有这么做，而是使用注意力池化，保留了每帧上的特征。

我们使用三种评估协议来评估学到的视频-文本编码器。

1. **零样本 (Zero-Shot, ZS)** : 在这个评估协议中, 模型不需要看到任何下游任务的数据, 即不需要针对特定任务进行调整或训练。模型直接利用其在预训练阶段学到的知识来处理新的任务。例如, 如果模型已经学会了如何理解视频和文本之间的关联, 那么它可以被用来检索与给定文本描述最匹配的视频, 或者反之。在零样本分类中, 模型会为每个可能的类别生成一个文本描述, 然后计算视频与这些描述的相似度, 以此来判断视频属于哪个类别。
2. **微调 (Finetuned, FT)** : 在这个评估协议中, 模型首先在大量的数据上进行预训练, 以学习视频和文本之间的通用表示。然后, 模型在特定的下游任务上进行微调, 即使用该任务的数据进一步调整模型的参数, 以提高在该任务上的性能。这种方法可以使模型更好地适应特定的数据集和任务。
3. **线性探测 (Linear Probe, LP)** : 在这个评估协议中, 模型的参数在预训练之后被冻结, 即不再更新。然后, 在模型的输出上添加一个线性分类器 (如线性支持向量机), 这个分类器仅使用下游任务的数据进行训练。这个评估协议的目的是测试预训练模型学到的特征对于特定任务来说有多好的区分能力, 而不考虑模型在任务上的最终性能, 因为最终性能还会受到线性分类器的影响。

线性探测 (Linear Probe) 是一种评估预训练模型特征学习能力的方法。在这个方法中，预训练模型的参数被固定，即不进行更新。然后，在这些特征的顶部添加一个简单的线性分类器，通常是线性支持向量机 (SVM) 或线性回归模型，这个分类器会使用下游任务的训练数据进行训练。

线性探测的目的是为了评估预训练模型学到的特征对于特定任务来说有多好的区分能力。通过这种方式，我们可以单独评估特征表示的质量，而不受模型复杂性的影响。

符号化表示整个训练和推理过程如下：

1. **预训练模型**: 设 f_{pretrain} 为预训练的 video-text 编码器，它将视频 v 或文本 t 映射到一个高维特征空间，即 $f_{\text{pretrain}}(v) \rightarrow \mathbb{R}^d$ 或 $f_{\text{pretrain}}(t) \rightarrow \mathbb{R}^d$ 。
2. **特征提取**: 对于下游任务的数据集 $D = (v_i, y_i)$ ，其中 v_i 是视频， y_i 是对应的标签，我们使用预训练模型提取特征，得到 $\phi(v_i) = f_{\text{pretrain}}(v_i)$ 。
3. **线性分类器训练**: 设 g 为线性分类器，其参数为 w 和 b ，用于将特征 $\phi(v_i)$ 映射到标签 y_i 。分类器的训练目标是最小化损失函数 L ，例如对于线性 SVM，损失函数是合页损失 (hinge loss)。因此，训练过程可以表示为：

$$\min_{w,b} \sum_{(v_i,y_i) \in D} L(g(\phi(v_i); w, b), y_i)$$

4. **推理过程**: 在推理阶段，对于一个新的视频 v ，我们首先提取其特征 $\phi(v) = f_{\text{pretrain}}(v)$ ，然后使用训练好的线性分类器 g 进行预测：

$$\hat{y} = g(\phi(v); w, b)$$

线性探测方法的关键优势是它的简单性。由于分类器是线性的，因此它提供了一个关于预训练特征表示的直接性能指标。如果线性分类器能够在下游任务上表现良好，那么这表明预训练模型已经学到了有用的特征。相反，如果线性分类器的性能不佳，那么这可能意味着需要更复杂的微调策略或更高级的特征工程来提高性能。

1. Epic-Kitchens-100 (EK-100): 这是一个流行的、具有挑战性的第一人称视频识别基准。该段落提到了两个基于此数据集的任务：

- **多实例检索 (MIR):** 这个任务要求模型能够根据给定的视频找到对应的文本描述，以及根据给定的文本找到对应的视频片段。这需要模型理解和关联视觉和文本信息。
- **动作识别 (CLS):** 这个任务要求模型对视频中的动作进行分类，包括97个动词和300个名词，总共形成3,806个动作类别。这需要模型能够识别和理解复杂的动作和对象。

2. Ego4D: 这是另一个第一人称视角的数据集，该段落提到了两个基于此数据集的任务：

- **多选题 (EgoMCQ):** 这个任务要求模型在给定的视频片段中选择正确的文本描述。这需要模型能够理解视频内容并从多个选项中选择最匹配的描述。
- **自然语言查询 (EgoNLQ):** 这个任务要求模型根据给定的文本查询输出视频中的相关时间间隔。这需要模型能够理解文本查询并定位视频中的相关信息。

对于EgoNLQ任务，模型根据给定的自然语言文本查询，在视频中找到相关的时空区间（temporal intervals）。

自然语言查询任务要求模型具备以下能力：

- 1. 文本理解：**模型首先需要理解输入的文本查询，这通常是一个关于视频中可能发生的事件或活动的描述。例如，查询可能是“找到我切洋葱的时刻”。
- 2. 视频内容检索：**模型接着需要在视频数据中检索与文本查询内容相关的信息。这意味着模型需要识别并定位视频中与查询相关的视觉内容。
- 3. 时间定位：**由于视频是一个时间序列数据，模型不仅需要找到相关的视频帧，还需要准确地定位这些帧出现的时间区间。例如，如果查询是“找到我切洋葱的时刻”，模型需要输出视频中切洋葱开始和结束的确切时间点。
- 4. 时空关联：**模型需要将文本查询中的描述与视频中的时空事件关联起来。这要求模型能够理解视频中事件的时序关系，并能够根据文本描述提取出相应的时间段。

EgoNLQ任务的难点在于它要求模型在理解自然语言描述的同时，还能够处理视频数据的时间维度，并在视频中找到与描述匹配的特定时间段。这需要模型具备高级的视频内容理解和时间序列分析能力。

总的来说，EgoNLQ任务是评估模型在第一人称视角视频数据上的时空事件定位能力，这对于开发能够理解和响应用户查询的交互式视频系统非常重要。通过这个任务，研究者可以推动模型在视频内容理解和自然语言处理方面的边界，为实现更加智能和用户友好的视频分析工具奠定基础。

3. EGTEA: 这是一个第一人称视角的数据集，用于评估模型在动作识别方面的性能。**模型需要将视频分类为106类细粒度的烹饪活动。**
4. CharadesEgo: 这是另一个第一人称视角的数据集，用于评估模型在动作识别方面的性能。**模型需要将视频分类为157类日常室内活动。**与EK-100、Ego4D和EGTEA不同的是，CharadesEgo的数据是通过众包方式使用头戴式手机相机捕获的，这可能会引入不同的视角和场景变化。

实验设计

基准测试——EK-100

XDU

Method	Backbone	mAP			nDCG		
		V→T	T→V	Avg.	V→T	T→V	Avg.
(ZERO-SHOT)							
EgoVLP [39]	TSF-B	19.4	13.9	16.6	24.1	22.0	23.1
EgoVLP* [39]	TSF-B	26.0	20.6	23.3	28.8	27.0	27.9
LAVILA	TSF-B	35.1	26.6	30.9	33.7	30.4	32.0
LAVILA	TSF-L	40.0	32.2	36.1	36.1	33.2	34.6
(FINETUNED)							
MME [75]	TBN	43.0	34.0	38.5	50.1	46.9	48.5
JPoSE [75]	TBN	49.9	38.1	44.0	55.5	51.6	53.5
EgoVLP [39]	TSF-B	49.9	40.5	45.0	60.9	57.9	59.4
LAVILA	TSF-B	55.2	45.7	50.5	66.5	63.4	65.0
LAVILA	TSF-L	54.7	47.1	50.9	68.1	64.9	66.5

Table 2. **EK-100 MIR.** LAVILA outperforms prior work across all settings, metrics and directions of retrieval, with larger gains when switching to a larger model. Specifically, our best model achieves over 10% absolute gain in the zero-shot setting and 5.9 ~ 7.1% gain in the finetuned setting. EgoVLP* refers to our improved version of [39], details of which are given in Appendix F.

Method (Backbone)	Pretrain	Top-1 accuracy		
		Verb	Noun	Action
IPL (I3D) [71]	K400	68.6	51.2	41.0
ViViT-L [2]	IN-21k+K400	66.4	56.8	44.0
MoViNet [34]	N/A	72.2	57.3	47.7
MTV [79]	WTS-60M	69.9	63.9	<u>50.5</u>
MTCN (MFormer-HR) [33]	IN-21k+K400+VGG-Sound	70.7	62.1	49.6
Omnivore (Swin-B) [22]	IN21k+IN-1k+K400+SUN	69.5	61.7	49.9
MeMViT [76]	K600	71.4	60.3	48.4
LAVILA (TSF-L)	WIT+Ego4D	72.0	62.9	51.0

Table 9. **The performance of action recognition on EK-100.** We report top-1 accuracy on verb, noun, and action. LAVILA outperforms all prior works in terms of action-level top-1 accuracy.

Method	EgoMCQ		EgoNLQ			
	Accuracy (%)		mIOU@0.3		mIOU@0.5	
	Inter-video	Intra-video	R@1	R@5	R@1	R@5
SlowFast [24]	-	-	5.45	10.74	3.12	6.63
EgoVLP [39]	<u>90.6</u>	<u>57.2</u>	10.84	<u>18.84</u>	6.81	<u>13.45</u>
LAVILA (B)	93.8	59.9	<u>10.53</u>	19.13	<u>6.69</u>	13.68
LAVILA (L)	94.5	63.1	12.05	22.38	7.43	15.44

Table 3. **Ego4D EgoMCQ and EgoNLQ.** LAVILA outperforms prior work on both Multiple-Choice Questions and Natural Language Questions on Ego4D, with nearly 6% absolute gain on the challenging intra-video MCQ task that requires reasoning over multiple clips from the same video to answer a question.

Method	Backbone	Pretrain	Top-1 Acc.	Mean Acc.
Li <i>et al.</i> [37]	I3D	K400	-	53.30
LSTA [63]	ConvLSTM	IN-1k	61.86	53.00
IPL [71]	I3D	K400	-	60.15
MTCN [33]	SlowFast (V+A+T)	K400+VGG-Sound	<u>73.59</u>	<u>65.87</u>
Visual only	TSF-B	IN-21k+K400	65.58	59.32
LAVILA	TSF-B	WIT+Ego4D	77.45	70.12
LAVILA	TSF-L	WIT+Ego4D	81.75	76.00

Table 4. **EGTEA Classification.** LAVILA obtains significant gains on this task, outperforming prior work with over 10% mean accuracy. Since the backbones used are not all comparable, we also report a comparable baseline with TSF-B (“Visual only”).

基准测试——CharadesEgo

Method	Backbone	mAP (ZS)	mAP (FT)
ActorObserverNet [57]	ResNet-152	-	20.0
SSDA [12]	I3D	-	25.8
Ego-Exo [38]	SlowFast-R101	-	30.1
EgoVLP [39]	TSF-B	<u>25.0</u>	<u>32.1</u>
LAVILA	TSF-B	26.8	33.7
LAVILA	TSF-L	28.9	36.1

Table 5. **CharadesEgo Action Recognition.** LAVILA sets new state-of-the-art in both zero-shot (ZS) and finetuned (FT) settings. Note that CharadesEgo videos are visually different compared to Ego4D videos, on which LAVILA is pretrained.

基准测试——HowTo100M(Third-Person Video)

Method	Vis. Enc.	UCF-101	HMDB-51
MIL-NCE [44]	S3D	82.7	54.3
TAN [25]	S3D	83.2	56.7
Baseline (w/o LLM)	TSF-B	<u>86.5</u>	59.4
LAViLA	TSF-B	87.4	<u>57.2</u>
LAViLA	TSF-L	88.1	61.5

Table 6. **LAViLA on third-person videos.** We measure the linear-probing action classification performance of the video model after pre-training on HowTo100M [45].

基准测试——Semi-supervised Learning

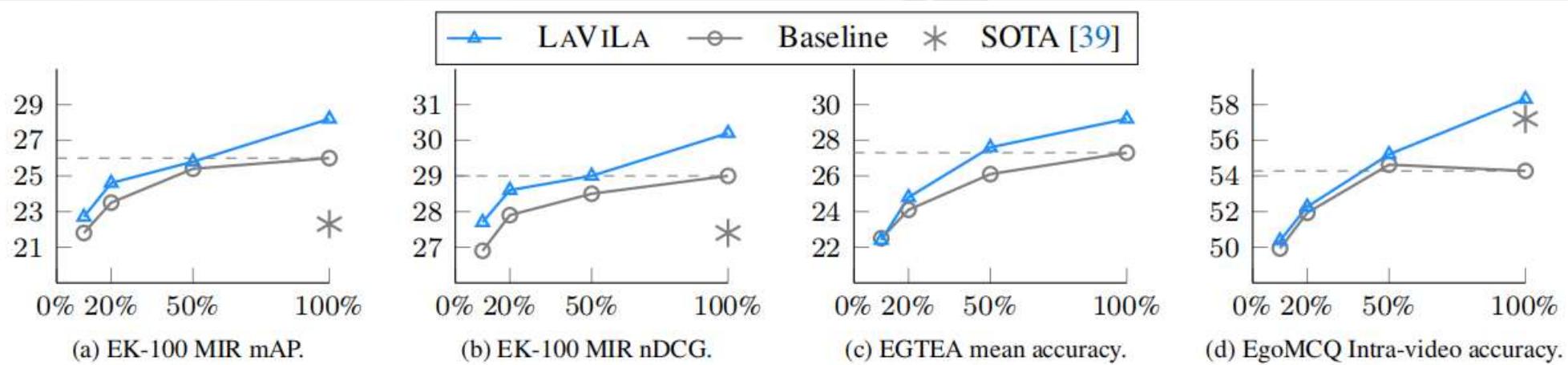


Figure 5. **LAVILA is effective in a semi-supervised setting where only a limited amount of narrations are given.** Comparing zero-shot performance of pre-training, LAVILA consistently outperforms the groundtruth-only baseline when 10, 20, 50, 100% data is used. We also achieve comparable result with state-of-the-art with only 50% of the annotated data.

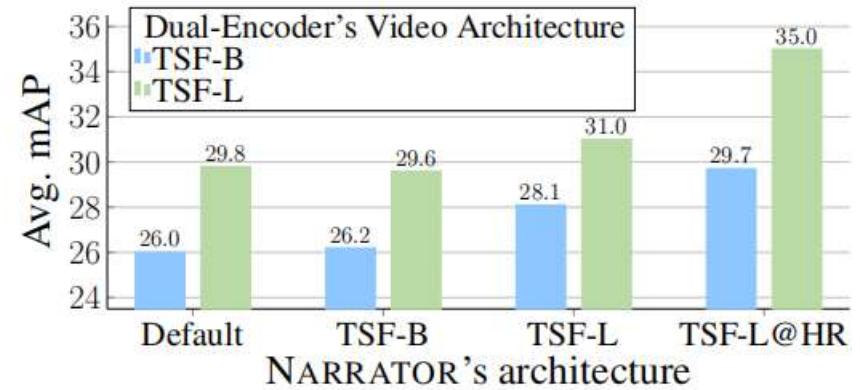
消融实验——不同语言监督的贡献

Rephr. Recap.	Pseudo cap.	EK-100 MIR		EgoMCQ		EGTEA	
		Avg. mAP	Avg. nDCG	inter-video	Intra-video	Mean	Top-1
✓		26.0	28.8	93.6	54.3	27.3	30.1
	✓	28.0	30.1	93.5	56.9	<u>29.8</u>	30.8
✓	✓	27.1	29.9	93.2	59.2	26.8	31.2
✓	✓	<u>29.7</u>	31.5	93.6	58.3	29.4	36.6
✓	✓	✓	29.9	<u>31.4</u>	93.6	<u>59.1</u>	31.1
							36.0

Table 8. **Contributions of different Language Supervision.** We can see that (1) using REPHRASER (“Rephr.”) and NARRATOR (“Recap.”) improve downstream zero-shot performance complementarily, (2) dense pseudo-captioning further improves performance on 3 out of 6 metrics.

消融实验——采样方法

Sampling method	# of sentences	Avg. mAP
N/A (baseline)	-	26.0
Beam search	1	27.9
Nucleus	1	29.6
Nucleus	10	29.7



(b) **Sampling.** LAVILA benefits more from narrations produced by nucleus sampling than beam search.

(c) **Scaling effect of LAVILA.** Gains increase on scaling the video encoder in NARRATOR. Default refers to only using the original narrations.

LAVILA是一种新的视频理解方法，它通过结合视频内容和大型语言模型来自动生成视频的叙述。这种方法允许系统在不需要大量人类生成的叙述数据的情况下学习视频与语言之间的关系。通过这种方式，LAVILA能够从长视频中提取有用的信息，并生成描述视频内容的自然语言叙述。

这项研究表明，LAVILA在多种视频理解任务上取得了显著的进步，这些任务包括理解第一人称（如使用头戴式摄像机拍摄的视频）和第三人称视角（如一般场景视频）的视频内容。**LAVILA的性能超过了使用同样数量人类叙述视频训练的基线模型，这意味着自动生成的叙述数据可以作为一种有效的训练资源。**

此外，研究还发现，**随着训练叙述的增加、视觉基础模型的扩大以及使用的语言模型变得更强大，LAVILA的性能也随之提高**。这表明LAVILA方法**具有良好的扩展性**，可以通过增加数据量和模型复杂性来进一步提升性能。因此，这些方向被视为未来工作中具有潜力的研究领域，可以通过进一步的技术创新来优化视频-语言表示学习。

XDU

THANK FOR ATTENTION

厚德 求真 励学 笃行