毕业设计（论文）

GRADUATION DESIGN (THESIS)

论文题目　基于 CLIP 模型的视频文本检索设计与实现

学院名称　计算机科学与工程学院

专业名称　人工智能

学生姓名　黄元通

指导教师　栗伟

二〇二三 年 六 月

学号___20195063_____ 密级_____

东北大学本科毕业论文

基于 CLIP 模型的视频文本检索设计与实现

学 院 名 称 ：计算机科学与工程学院

专 业 名 称 ：人工智能

学 生 姓 名 ：黄元通

指 导 教 师 ：栗伟　　副教授

二〇二三年六月

郑 重 声 明

本人呈交的学位论文，是在导师的指导下，独立进行研究工作所取得的成果，所有数据、图片资料真实可靠。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确的方式标明。本学位论文的知识产权归属于培养单位。

本人签名：  日期：2023.6.7

摘 要

CLIP 预训练模型提出后，视频文本检索领域有了很大发展，短短一两年内基于CLIP 的新模型层出不穷，不断刷新 SOTA，但模型规模也是不断增大。而同时，模型规模的增大也促进了低参数量微调领域工作的发展，试图实现只调整极少的参数量，就能达到甚至是超越预训练模型全微调时的性能。

针对现有基于 CLIP 预训练模型方法存在的训练时间长、模型规模大等问题，本文在 CLIP4Clip 模型的基础上，采用关键帧保存方案和 Adapter Tuning 低参数量微调，提高了计算资源有限情况下的训练效率和模型性能。

训练速度方面，本文首先采用帧保存方案，将视频库中的关键帧提取并保存为图片，从而加快数据加载速度，将训练速度提高了 14.6 倍；然后采用 AIM 模型的 Adapter 设计方案，在 CLIP4Clip 模型中插入可训练的 Adapter 层，只训练少量参数实现快速收敛，最终实现训练速度提高 34 倍。

模型性能方面，论文证明了平均选取视频关键帧，比按照最大帧间差选取视频关键帧的效果更优；然后证明了 AIM 在视频检索领域的有效性；并可以使用 AIM 来提高模型性能，将模型在 MSR-VTT 数据集上的 R@1 从 42.2%提升至 43.4%，R@5 从 70.2%提升为 71.1%。

本文设计并实现了一个视频文本检索系统，为了保障系统的检索速度，搭建了向量数据库，并测试了不同数据保存方式的数据存取速度。系统使用 Django 搭建了 Web 端应用，实现了多种功能需求，展示了本文方法在视频文本检索系统中的应用效果和潜力。

关键词：视频文本检索；CLIP 模型；CLIP4Clip 模型；低参数量微调；训练加速

ABSTRACT

After the proposal of CLIP pre-training model, the field of video text retrieval has made great development. Numerous new models based on CLIP have emerged in just a year or two, constantly refreshing the state-of-the-art, but the size of models has also been increasing. Meanwhile, the increase in model size has also promoted the development of low-parameter fine-tuning techniques, attempting to achieve or even surpass the performance of full fine-tuned models with minimal parameter adjustments.

To address the problem of long training time and large model size existing in methods based on CLIP, this thesis adopts a keyframe-saving scheme and Adapter Tuning on the basis of CLIP4Clip model, which improves the training efficiency and model performance under the condition of limited computing resources.

In terms of training speed, this thesis first adopts a frame-saving scheme to extract and save keyframes from the video library as images, which speeds up data loading and increases training speed by 14.6 times. Then, the adapter design scheme of AIM is used to insert trainable Adapter layers into the CLIP4Clip model, trains only a small number of parameters to achieve fast convergence, ultimately increasing the training speed by 34 times.

In terms of model performance, this thesis proves that average selection is better than selecting video key frames according to the maximum frame difference. Then, it demonstrates the effectiveness of AIM in video text retrieval and shows that AIM can be used to improve model performance. The R@1 on the MSR-VTT dataset was increased from 42.2% to 43.4%, and R@5 was increased from 70.2% to 71.1%.

A video-text retrieval system is designed and implemented. To ensure its retrieval speed, a tensor database is built, and the data access speed of different data storage methods is tested. The system uses Django to develop a Web application that meets various functional requirements, demonstrating the application effects and potential of the proposed method in the video-text retrieval system.

Key words: video text retrieval; CLIP; CLIP4Clip; parameter-efficient fine-tuning; training acceleration

目 录

1 绪论

1.1 选题背景

随着网络上视频数量的不断增长，视频检索成为高效查找视频的一项新需求。视频检索任务就是输入一段文本，检索出最符合文本描述的视频。传统的视频检索通常要求视频带有额外的文字标签，通过匹配查询语句的关键词与视频标签实现检索。这一方案高度依赖关键词和视频标签，与真正的内容匹配存在差距。

随着深度学习在多模态领域的高速发展，视频文本跨模态检索能够理解文字和视频的内容，从而实现视频与文本之间的匹配。相比传统方法，基于内容理解的视频检索也更加接近人类的思考逻辑。目前主流的实践方法是将视频和文本编码成特征向量，由于含义相近的向量在空间中的位置也是相近的，可以通过计算向量之间的相似度实现文本视频跨模态检索任务。

具体到端到端模型（直接以原始视频作为输入来训练模型）这一分支上，得益于数据集规模、模型规模和算力的增大，预训练模型在视频文本检索方面表现出显著的性能提升。尤其是 CLIP 预训练模型（contrastive Language-Image Pre-training）的提出，使得图像检索、视频理解、图像生成、图像描述、视觉问答等众多领域的相关模型有了显著的性能提升。

为了将预训练模型运用到不同下游任务中，往往需要对模型进行微调（fine-tuning），通常是全微调所有参数（full fine-tuning）。但随着预训练模型规模的不断增大，还有受实验条件的限制，全微调可行性变得越来越低，每增加一个新下游任务就需要全微调一个新模型，代价和成本很高。为了解决这个问题也出现了很多低参数量微调方案（parameter-efficient fine-tuning，简称 PEFT），如只调节部分参数、添加新的模块等，试图通过最小化微调的参数数量和计算复杂度，从而减小预训练模型的微调成本，同时达到甚至是超越全微调的效果。

1.2 国内外研究现状

1.2.1 基于大规模预训练模型

视频文本检索领域目前主流方法为使用大规模预训练模型并且在下游任务上进行微调，效果明显优于基于专家模型的方法。更细地，目前方法又可大致分为两类：

学习全局视频表示和句子嵌入的全局特征匹配方法，和使用片段-短语、帧-单词等细粒度信息的细粒度匹配方法。

1.2.1.1 基于全局特征匹配的方法

基于全局特征的方法检索过程相对简洁高效，该方法采用视频和文本编码器提取特征，将全局的视频和文本特征映射到一个公共的度量空间，然后根据特征表示之间的点积或余弦相似性进行检索任务，可以保证检索的效率。

针对视频文本检索任务，很多工作以端到端的形式训练模型。ClipBERT[1]模型是视频端到端的开山之作，模型使用 BERT[2]和 ResNet50 分别提取文本和视频特征，证明了稀疏采样效果明显优于密集采样的效果，且效率更高。Frozen[3]模型提出一种用于端到端检索的视频图像联合编码器，模型是对最近的 ViT[4]和 Timesformer[5]架构的扩展，用以编码空间和时间信息，相比于 ClipBERT 模型，结构复杂性更低，具有更好的效率。

CLIP[6]直接在 4 亿个图片文本对上进行对比学习，将自然语言作为图片的监督信号，预训练后的模型可以直接在诸多下游任务上进行零样本迁移，其强大的迁移能力，能够有效提高视频文本检索的性能。CLIP4Clip[7]（CLIP For video Clip retrieval）以端到端的方式将 CLIP 模型的知识转移到视频语言检索中，文章验证了三种相似度计算方式（无参数方法、序列型方法、密集型方法）的效果，验证了基于 CLIP 模型的视频文本检索工作的可行性。

CLIP2TV[8]模型基于 CLIP 的框架，由视频文本对齐模块和视频文本匹配模块组成，这两个模块以协调的方式进行端到端的训练，并相互提高性能。CLIP2Video[9]模型将视频和语言理解分为两个独立的问题，多模态图像文本训练的空间表示、和视频帧与视频语言的时间关系,设计了时间差分块和时间对齐块来捕获视频帧的时间关系和视频语言关系。

1.2.1.2 基于细粒度匹配的方法

基于全局特征的方法可以保证检索的效率，但是将很长的序列信息压缩到单一向量，促使网络学习去学习一个很复杂的抽象，很容易忽略细粒度信息。细粒度信息是指文本视频中的帧、词、动作等信息，实现文本和视频之间细粒度信息的匹配，对模型的优化具有极大的帮助。

细粒度匹配近两年的研究同样基于 CLIP 模型来初始化文本和视频特征。Object-aware Transformers[10]，在关键帧中提取物体信息，文本中加入检测物体的标签信息，

用目标特征和标签来指导训练。DRL[11]将 token-wise 交互引入视频文本检索，并自适应的赋予权重，从序列维度和通道维度进行语义的分解，能够高效的捕获顺序和层次信息，实现细粒度检索。HunYuan_tvr[12]模型探索了分层次的跨模态交互，同时在视频-句子、片段-短语和帧-词层面交互，结合了 CLIP4Clip 的粗粒度检索以及 DRL 的细粒度检索，是目前的视频文本检索 SOTA 模型。

1.2.2 低参数量微调

目前常用的低参数量微调方法主要可分为 Adapter Tuning、Prompt Tuning、LoRA（Low-Rank Adaptation of Large Language Models）等三类。

1.2.2.1 Adapter Tuning

谷歌在论文[13]中提出了针对 BERT[2]模型的 PEFT 微调方法，以尝试提高微调效率。在处理特定的下游任务时，全微调效率低下，而只微调接近下游任务的部分层又很难达到较好的效果。于是提出了 Adapter 模块，将其插入到 Transformer 中。在训练时，冻结原来预训练模型的所有参数使其不变，只对新增的 Adapter 结构进行微调。为了尽可能减小引入参数的数量保证训练的高效性， Adapter 结构是下采样层（全连接层）、非线性层（GeLU）、上采样层（全连接层）并使用 skip-connection 连接，将高维度特征映射到低维度空间，然后再通过非线性变换，将低维度特征重新映射回原来的高维度空间，skip-connection 保障网络性能不退化。Adapter 同样可以应用到其他大规模预训练语言模型上。

Adapter 的改进中，为了支持更大的模型和更复杂的自然语言处理任务，提出了 Multi-Adapter[14]、Dynamic Adapter[15]等，Multi-Adapter 在 Transformer 模型中引入多个 Adapter 结构，以提高模型的表示能力和泛化能力；Dynamic Adapter 通过对 Adapter 结构进行动态选择，根据下游任务的需求来选择合适的 Adapter 结构，从而增强模型的适应性和灵活性。

1.2.2.2 Prompt Tuning

Prompt Tuning 在 CLIP 中使用的是 hard prompt，即一组人为设计的句子。在分类任务中，将 prompt 添加到“类别”标签前，组成完整的句子以减小“类别”标签这样的单词与训练数据中句子的数据集差异。但很明显，对于不同的任务集，需要认真考虑所需的任务和数据集，并结合专业知识重新设计不同的 prompt。

为避免手动设计各数据集的 hard prompt，之后 prompt 普遍采用可学习的 soft prompt。斯坦福大学在提出了 Prefix Tuning[16]，通过在输入序列之前拼接 Prefix 序

列，微调时预训练模型所有参数冻结，仅更新 Prefix 部分的参数。论文作者认为，Prefix 通常与特定任务相关联，因此在许多情况下，这种微调方法比全微调更有效；另外，Prefix Tuning 的方法还具有良好的可解释性，因为它可以通过查看 Prefix 来理解模型在执行特定任务时的行为。

谷歌在 2021 论文[17]中进行了 Prompt Tuning 在不同规模预训练模型上的效果实验，发现只要模型规模足够大，简单地在输入序列之前加入 Prompt 进行微调就能取得全微调的效果。

1.2.2.3 LoRA

微软和 CMU 在 LoRA[18]的论文中提出，现有的 PEFT 的方法仍存在以下 3 种问题：Adapter Tuning 由于增加了模型的深度，从而导致额外的推理延迟；Prompt Tuning、Prefix Tuning 等 Prompt 方法较难训练，同时缩短了模型可用的序列长度；往往效率和质量不能兼得，效果不如 full fine-tuning。而有语言模型参数的研究表明，尽管语言模型的参数数量庞大，但其中低秩的本质维度才是关键。受到这个观点启发，提出了 Low-Rank Adaption (LoRA) 方法，使用低秩矩阵模块模拟 full fine-tuning 过程，只更新在语言模型中起关键作用的低秩本质维度，以解决上述三个问题。

相比 Adapter Tuning 方法，该方法无需增加网络深度，避免了额外的推理延迟；由于未使用 Prompt Tuning 方式，也不会带来 Prompt 方法带来的一系列问题；而且该方法相当于使用 LoRA 模拟 full fine-tuning 过程，几乎没有训练效果的损失，并且后续实验结果证明了该方法的有效性。在实验中，研究人员将 LoRA 模块与 Transformer 的 attention 模块相结合，在 RoBERTa[19]、DeBERTa[20]、GPT-2[21]和 GPT-3 175B[22]这几个大型模型上进行了实验，并证明了该方法的有效性。

1.3 研究内容

1.3.1 实验思路

首先，因为计算机视频文件本质上就是超多倍的图像，所以当任何实验涉及到以视频为训练数据时，避免不了地会需要比其他情况下更多的计算资源。本论文的模型有关实验均在 1 张 3090 显卡上进行，可以说是相对非常有限了，所以模型结构的可扩充范围、模型可获得的最高性能也比较小。所以就将实验的首要目标设为缩短训练时间。

为了在尽可能保持模型规模小的情况下，改进模型结构以提高性能和缩短训练耗时，首先考虑到采用低参数量微调方案。三种低参数量微调方案分类里，LoRA 从理论上证明了其可以几乎无损失地拟合全微调，但这也意味着 LoRA 使得性能只能达到、而无法超越全微调的模型。Prompt Tuning 的问题正如 LoRA[18]论文里提出的那样，太过于依赖人类经验且不稳定；而且依据谷歌对 Prompt Tuning 的研究论文[17]结论，只要模型规模大，那么很短、很简单的 Prompt 设计方案就能取得很好的效果；可是本实验的模型规模相对来说非常小，那么很明显就不适合采用 Prompt Tuning。

所以在阅读到动作识别领域采用 Adapter Tuning 的 AIM[23]（Adapt pre-trained Image Models）论文后，因为其 Adapter 插入方案在动作识别领域数据集上获得了超过全微调模型的性能，所以选择使用 Adapter Tuning，尝试将 AIM 的 Adapter 插入方案应用到视频文本检索领域中，并在后续实验中依次证明 AIM 在视频文本检索领域的有效性、和 AIM 是否可用于提升模型性能。

1.3.2 论文结构

论文共分为五章。

第一章绪论，主要介绍了目前视频文本检索领域和低参数量微调的主流方法及最新模型，并结合本实验的具体情况，分析了各个低参数量微调方案在本实验条件下的优点和缺点，提出了本文方法的依据和解决思路。

第二章为相关技术介绍，结合各自的论文，介绍了实验中使用的 CLIP[6]、CLIP4Clip[7]、AIM[23]三个模型。

第三章为算法设计与实现。首先提出算法，介绍了算法的整体结构，详细介绍了模型的结构与实现，并从原因、实现方法和优势等方面介绍两种关键帧采样方法、关键帧保存方案、向量数据库。然后进行多组实验对算法各个部分进行测试，详细介绍了加快模型训练、关键帧选取方案、AIM 有效性、提升模型性能、不同数据集得分等五个方面的实验及结果，并给出了相应结论。

第四章系统设计与实现，按照需求分析、总体设计、详细设计、实现、测试的逻辑组织顺序介绍了所设计的视频文本检索系统，给出了系统需求分析、架构、流程图、用例图，介绍了系统各个模型的功能、意义及实现。

第五章结论，总结了实验内容，给出了不足和展望。

2 相关技术介绍

本章中，结合各自论文，介绍了实验中使用的 CLIP[6]、CLIP4Clip[7]、AIM[23]三个模型。

2.1 CLIP

CLIP 包括一个图像编码器（ResNet 或 ViT）、一个文本编码器（Transformer），使用对比学习将文本和图像联系起来。首先将文本通过文本编码器进行编码、将图像通过图像编码器进行编码，然后正则化，计算所有文本特征和所有图像特征间余弦距离；因为要使同一图像文本对（矩阵中的对角线元素）的结果趋近于 1，不同图像文本对（矩阵其他位置元素）的结果趋近于 0，所以可以采用对称交叉熵（Symmetric Cross Entropy）。

CLIP 训练使用 32768 的 batch size 训练 32 个 epochs，准确率最高模型 ViT-L/14 需在 256 张 V100 卡上训练耗时 12 天。大数据量决定了训练的高难度复现性，因此采用微调或直接 zero-shot 在下游任务中使用。

针对模型的效果，论文首先讨论了数据规模问题。

预训练模型若使用有监督训练，则需要大量的数据标注，成本较高；自监督方法的好处是不再需要标注。

一方面，由于文本和图像属于两个完全不同的模态，所以过去，自监督训练在自然语言处理领域和图像领域的发展也存在很大差距。自然语言处理领域，有很多效果较好且容易实现的自监督学习的方法，例如自回归、语言掩码等，而且预训练模型可以很好地迁移到下游任务而无需微调。

另一方面，从过直接从互联网上采集文本，自然语言处理领域可以相对简单地、收集到质量较好的大规模数据集，很大程度上促进了模型性能的提升。所以图像领域也尝试通过从互联网收集图像文本对（显然比单纯的文本数量更少，而且文本和图像之间的匹配关系也可能不是那么好），例如谷歌从互联网收集的 JFT-300M 数据集，用来训练 ViT[4]获得预训练模型，便成功在 ImageNet 上刷新了 SOTA。不过，JFT-300M 数据集里将文本用 18291 个类别来表示，模型则依旧使用 18291 分类的 softmax 作为分类器，限制了 ViT 的迁移能力和扩展性（因为如果数据集的类别数不一样的话就需要重新微调，开销比较大）。

OpenAI 的 CLIP 团队认为，谷歌的方法之所以比之前类似实验得到的模型性能更好，一个关键点就在于规模，包括训练数据集的规模和算力规模；之所以之前实验的模型效果并不好，是因为训练时使用的数据集规模没有 JFT-300M 那么大。所以 CLIP 模型实验中，其训练数据集从网上收集了 4 亿条数据（相比之下 JFT-300M 只有一亿多），而且也准备了更大的算力。

然后论文讨论了为何采取对比学习。

CLIP 模型数据量的增大使得训练难度增大。OpenAI 首先尝试了生成式模型：给定图像，让模型预测图像的文本，但是发现这种方法的训练效果不尽人意。所以改为采用对比学习：一个 batch 内，一个文本编码只有与它对应的图像是正样本（也就是对角线上元素），其他的图像都是负样本，目标就是优化图片文本相似度矩阵和文本图片相似度矩阵，让对角线元素趋近于 1、其他位置趋近于 0。另外，论文作者也认为因为训练数据是从互联网收集来的，存在一定的噪音（文本和图像可能并不完全匹配），所以不排除适当降低训练目标可能更好收敛的可能。

然后指出了 CLIP 的优缺点：

(1) CLIP 的 zero-shot 性能和有监督的 ResNet50 相当，但还不是 SOTA，论文作者认为还需要增加一千倍的计算量才能让模型达到 SOTA。

(2) CLIP 在某些数据集上表现较差，主要为预训练中难以涵盖的部分，如细粒度分类，抽象任务、计数等。

(3) CLIP 虽然在广泛的数据上体现出了健壮性（例如图像识别任务，风格差异巨大的不同图片里的同一事物，CLIP 计算出的概率都基本不变，一直非常有把握，而参与对比的其他模型则会判别失败），但是如果测试集的数据分布和训练集相差特别大，则仍然会表现较差，其 zero-shot 能力很大程度上是因为在 4 亿的训练数据中已经基本囊括了现有的大多数概念。例如 CLIP 在 OCR 任务性能较优，但 MNIST 却非常差，CLIP 团队认为是类似 OCR 的样本对在预训练图片中很多，但 MNIST 在训练集中没有类似的，因此 CLIP 体现出的泛化性仍然不是本质的泛化性。

(4) 由于训练集的限制，出现了很多社会性偏见，比如对不同种族性别年龄的倾向。

正如 CLIP 论文中所展示的那样，因为该预训练模型体现出了很强泛化能力，所以可以应用到非常多的下游任务中。也使得在其发布后，很多领域的模型都直接获得了明显的性能提升，例如，可将其用于以下领域。

(1) zero-shot 的目标检测任务。例如当检测测试集中样本时，若包含了训练集中没有的类别时，就可以进行开放词汇的物体检测；

(2) 图像检索。基于文本搜索图像，如 CLIP 作为 DALL-E[24]的排序模型，从生成的图像中选择和文本相关性较高的；

(3) 图像生成。如用 CLIP 结合 StyleGAN[25]得到的 StyleCLIP[26]，通过输入文字实现对图像进行编辑；

(4) 视频理解。可以将 CLIP 的文本图像对知识迁移到文本视频，如 CLIP4Clip[7]等模型实现的视频文本检索。

2.2 CLIP4Clip

CLIP4Clip[7]主要解决了将 CLIP 模型端到端的应用到视频领域、将 CLIP 应用到视频数据集上该如何进行时序建模的问题。

CLIP4Clip 包含一个文本编码器（CLIP 的文本编码器 Transformer）、一个视频编码器（CLIP 的图像编码器 ViT）、一个相似性计算模块（无参数方法、序列型方法、密集型方法）。首先从视频片段中提取帧、通过视频编码器进行编码，然后使用相似性计算模块计算文本特征和各批帧间相似性，损失函数同样使用对称交叉熵。

相似性计算模块中：无参数方法使用平均池化（命名为 meanP）来聚合所有帧的特征；序列型方法使用 LSTM（命名为 seqLSTM）、或带位置嵌入的 Transformer（命名为 seqTransf）来聚合所有帧的特征；密集型方法将文本特征和各帧图像特征拼接后送入 Transformer（命名为 tightTransf）以实现所有帧特征的聚合、文本特征视频特征的交互。

CLIP4Clip 实验得到了以下结论：

(1) 只用单个图像来进行视频文本检索是不可行的；

(2) 在 CLIP 模型的基础上进行 post pre-training 可以提升性能；

(3) 不同的数据集上使用不同时序建模方式可能会影响到最终的性能表现：小数据集上最好采用平均池化机制，大数据集上最好引入更多参数以学习大型数据集的时间依赖性；

(4) 训练时，模型对学习率设置很敏感。

2.3 AIM

AIM[23]是 2023 年发布在 ICLR 上的论文，主要目的是使用低参数量微调中的 Adapter Tuning，添加 Adapter[13]将 CLIP 等预训练模型应用到视频动作识别领域；训练时冻结整个网络、只训练新添加的 Adapter 以降低训练开销，以极少的参数加强模型空间、时间信息学习能力。

AIM 论文作者认为若要将图像领域预训练模型迁移到视频领域（视频动作识别领域），需要让模型能良好地表示每个帧（空间建模 spatial modeling），并且要能良好地表示跨帧的时间信息（时间建模 temporal modeling）。所以在使用 Adapter Tuning 进行视频任务时，关键问题之一就是如何执行时间建模。

为了增强时间建模能力，一些研究向图像模型添加新的时间模块，例如相似性计算模块使用 tightTransf 的 CLIP4Clip，添加了一个新的 Transformer 用于所有帧特征的聚合、文本特征视频特征的交互；除此之外还有在文本编码器和视频编码器内进行交互、添加额外网络专门学习时间信息等等不同方式。但因为需要全微调，训练成本高，所以作者使用 Adapter Tuning 进行低参数量微调实验。

论文中设计了 Spatial Adaptation、Temporal Adaptation、Joint Adaptation 三种 Adapter 添加方式，并以 Frozen 模型的性能为基准，验证了三种 Adapter 的效果。

首先添加的是 Spatial Adaptation。

Spatial Adaptation 目的是加强模型的空间建模能力。在自注意层之后插入一个 Adapter。与未添加时间模块的 Frozen 全微调相比，添加 Spatial Adapter 的模型 Top-1 召回率从 36.2%提升为 36.7%（Frozen 直接使用预训练模型时 Top-1 召回率为 15.1%）。

在 Spatial Adaptation 的基础上，添加 Temporal Adaptation。

Temporal Adaptation 目的是建立模型的时间建模能力。作者将预训练模型中的多头自注意层 MSA 直接进行复用，用于空间建模的称作 S-MSA，用于时间建模的称作 T-MSA（T-MSA 在 S-MSA 之前），在 T-MSA 后插入一个 Adapter。当输入矩阵形状为[T, N + 1, D]时，首先转置为[N + 1, T, D]（其中 N 是一张图片的图像块数量，T 是一段视频取的关键帧数量数），然后输入到 T-MSA 中，尝试学习 T 帧之间的关系。

尽管 T-MSA 和 S-MSA 的输入维度不同，但是是同一个层（即预训练模型中的 MSA），在训练时冻结，从而实现增强模型时间建模能力的同时保持参数量不变。为

了保持初始化时添加了 Adapter 的网络输出与原始预训练模型输出相同，Temporal Adapter 权重初始化为零并且不使用跳跃连接，以去除训练开始时 Temporal Adapter 的影响。

与添加时间模块的 Frozen 全微调相比，添加 Spatial Adapter 和 Temporal Adapter 的模型 Top-1 召回率从 59.5%提升为 61.2%。

为了联合调整 Spatial Adapter 和 Temporal Adapter 学习的空间和时间建模能力，最后再加入 Joint Adaptation。

在 MLP 层旁边并联一个 Adapter。同样，为了保持初始化时添加了 Adapter 的网络输出与原始预训练模型输出相同，Joint Adapter 不使用跳跃连接。在进一步添加了 Joint Adapter 后，模型 Top-1 召回率从 61.2%提升为 62.0%。

本章总结：

实验中，使用的是 CLIP 作为模型主体，使用平均池化来聚合所有帧的特征，并且采用 AIM 的 Adapter 插入策略进行 Adapter 插入。将在算法设计与实现一章中详细介绍模型的结构与实现，并从 AIM 有效性、AIM 用于提升性能等方面进行实验，测试模型性能。

3 算法设计与实现

本章中将首先介绍算法的整体结构，然后依次介绍各部分的设计与实现，最后将进行多组实验，测试算法在训练速度、不同数据集上模型性能等方面的结果。

3.1 算法整体结构

算法主要由关键帧选取、关键帧保存、模型、向量数据库四个部分组成，如图 3.1 所示。

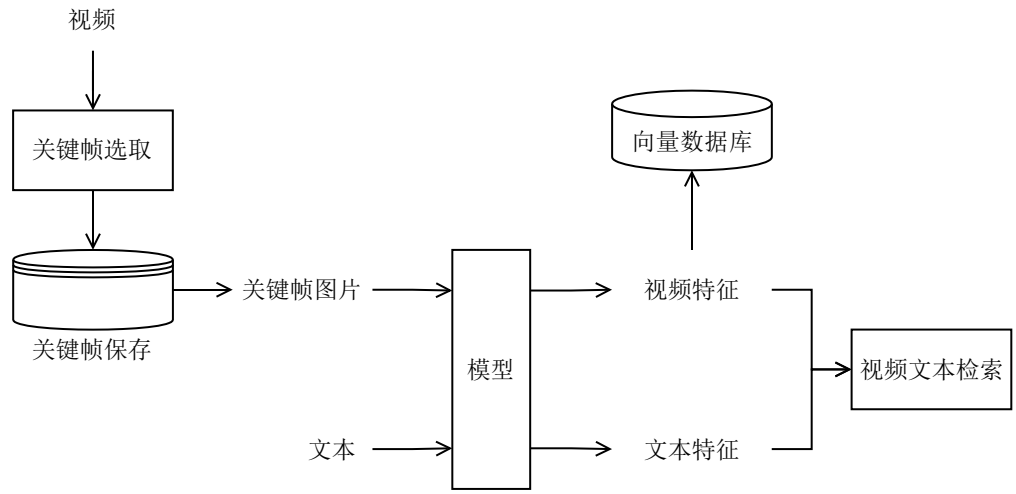

图 3.1 算法整体结构

其中关键帧选取获得每个视频的图像，共有平均选取和最大帧间差选取这两种方案。将在关键帧选取与保存一节中介绍其实现，并在算法测试的关键帧选取方案一节实验测试两种方案的效果。

关键帧保存将视频的关键帧以图片形式保存，以加快训练速度。将在关键帧选取与保存一节中介绍其实现，并在算法测试的耗时与可调参数一节中测试关键帧保存方案对训练的加速效果。

模型是使用的网络模型，可以对文本编码获得文本特征，对视频的关键帧图片编码获得视频特征。将在模型一节中根据 CLIP→CLIP4Clip→AIM 的模型发展顺序介绍其实现，并在算法测试的耗时与可调参数一节中测试低参数量微调的 AIM 对训练的加速效果；在算法测试的 AIM 有效性一节中测试 AIM 的 Adapter 插入方案是否在视频文本检索领域中有效；然后在算法测试的提升性能一节中测试 AIM 的 Adapter 插入方案是否可以用于提升模型的性能；最后在算法测试的 MSVD 数据集得分一节中测试模型在其他数据集上的性能。

向量数据库将视频特征保存起来，这样模型在测试时，便可以直接从向量数据库加载视频特征，而无需重新计算，从而加快模型在测试时的速度。将在向量数据库一节中介绍其实现，并在后续的系统设计与实现中介绍其使用。

3.2 模型

3.2.1 模型整体结构

模型整体结构如图 3.2 所示。实验中，使用的是 CLIP 系列模型最小的 CLIP ViT-B/32 作为模型主体，采用 CLIP4Clip 中将视频转化为 CLIP 输入的思路构造实验所用模型，并且实验所用模型可采用 AIM 的 Adapter 插入策略进行 Adapter 插入。

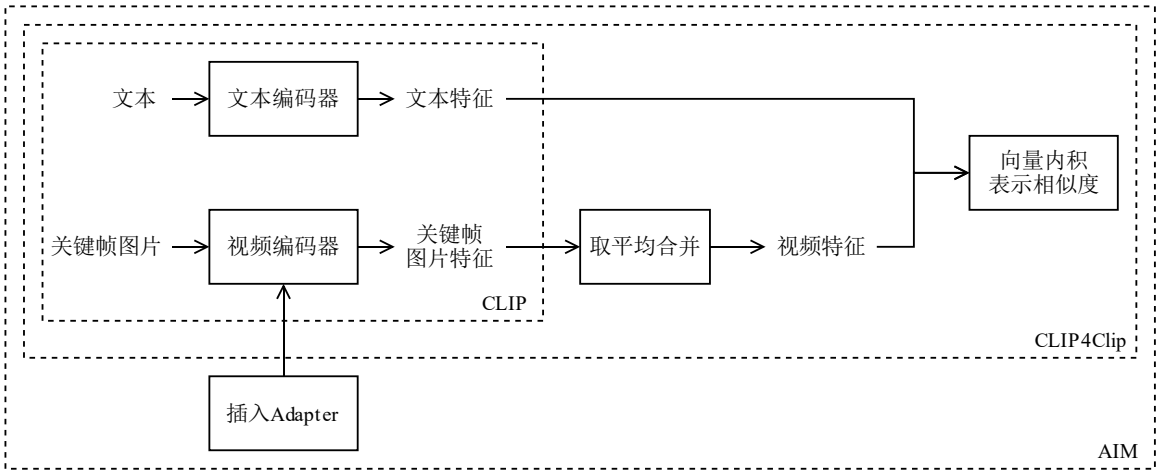

图 3.2 模型整体结构

“取平均合并”这一步将一个视频所有关键帧图像特征通过取平均值合并成一个，并将这一个平均值向量作为视频特征，将添加了此步骤的 CLIP 记为 CLIP4Clip；“插入 Adapter”这一步就是向 CLIP 的视频编码器的 Transformer 层中插入 Adapter，将添加了此步骤的 CLIP4Clip 记为 AIM。

接下来，本节将按照 CLIP→CLIP4Clip→AIM 的模型发展顺序，一步一步地构建出实验所用模型，详细介绍其结构与实现。

3.2.2 CLIP 结构与实现

实验中使用的是 CLIP 系列中规模最小的 ViT-B/32，文本编码器、图像编码器主体各为一个 12 层、8 头的 Transformer。

3.2.2.1 Transformer 层

Transformer 层结构如图 3.3 所示，在 CLIP 代码中，其被命名为 Residual Attention Block，多个 Residual Attention Block 串联就得到了所需的 Transformer，如图 3.4 所

示。其结构中：

(1) 均使用下三角 mask，使得自注意力中只能看到过去信息；

(2) 使用 Pre-Norm，2 个 Pre-Norm 分别位于多头自注意力层（MSA）、全连接前馈层（FFN）之前；

(3) 在多头自注意力层和全连接前馈层外使用跳跃连接；

(4) 模型整体使用半浮点（float16）以降低计算时间。当然按照惯例，在遇到指数等计算时，为使数值不那么容易溢出，暂时将数据转为浮点（float32）用于计算，然后计算结果再转回半浮点（float16），这包括模型中所有 layer norm 层、对编码器得到的文字特征和图像特征进行 L2 正则等过程。

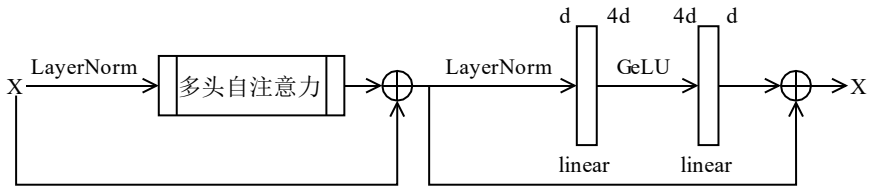

图 3.3 Residual Attention Block 结构图

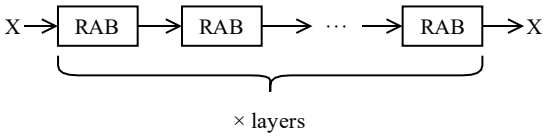

图 3.4 多个 Residual Attention Block 串联得到 Transformer

3.2.2.2 多头自注意力层

将自注意力层[27]的输入、输出分别记为 X、Z，函数关系见公式(3.1)：

$$Z = softmax\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) \times V, Q = XW_Q, K = XW_K, V = XW_V \tag{3.1}$$

其中 Q、K、V 分别为由 X 计算得到的（X 所对应的）查询值 Query、键值 Key、信息 Value，d_k为常数（网络超参数），因此整个自注意力过程中，网络能学习的参数就是W_Q, W_K, W_V矩阵。

$Q \times K^T$得到 X 各单词 i 与其他所有各单词 j 的相似度矩阵，再乘上 V（相当于单词 i 计算与所有各单词 j 的加权平均）就得到融入了相关单词信息的结果。

进一步，可以使用多个 Self-Attention 模块来计算一个 Z。将 k 个（超参数）自注意力机制得到的 k 个Z_i拼接起来，然后再与一个矩阵W_C相乘得到最终的 Z。其中 X, Z_i, Z 的维度都是[batch size, sentence length, embedding size]，矩阵W_C为新引入的一个可学习的参数矩阵。此过程称为多头自注意力机制（MSA），流程示意图如图 3.5

所示。

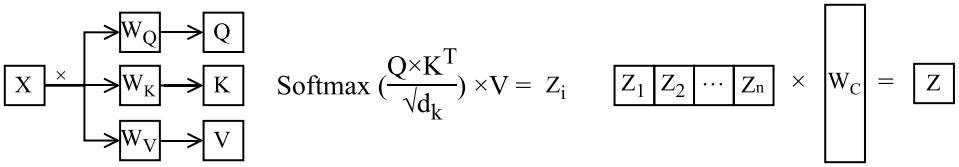

图 3.5 多头自注意力机制流程示意图

3.2.2.3 文本编码器

使用的 CLIP 的文本编码器是一个 Transformer。

对于输入的句子，首先需要使用 tokenizer 进行分词、分子词划分、token 化等操作，转化为 token 序列。

CLIP 使用 BPE 进行分子词划分，使用的分子词字典大小为 49408。为了合理控制字典大小，同时避免出现 unknown（未知分子词，即未出现在字典中的分子词），tokenizer 的字典构造使用以下方式：

(1) 句子中字符使用 Unicode，首先使用 UTF-8 进行编码（每个 Unicode 字符，UTF-8 用 1~4 个字节编码，1Bit=8bit）；

(2) 然后拆分为单个字节，从而实现将未见过的 UTF-8 字符拆碎成字节；

(3) 分子词字典{字节值: Unicode 字符}中相应包含字节值 0~255 的 256 个字符，这 256 个中是[a A 1 , 乱码 控制字符]等基本字符。例如 CLIP 代码中的字典内只有英文分子词，那么训练出的模型在遇到中文输入时，一个汉字字符就会被拆为 3 个单独的字节：

a) 以汉字‘一’为例，其 UTF-8 编码是 b'\xe4\xb8\x80'，就会被拆分为 b'\xe4', b'\xb8', b'\x80'三个字符；

b) 其字节值分别是 228, 184, 128；

c) 各字节值对应的字符是：'ä', ',', '\x80'。

(4) 并且使用 BPE 进行分子词划分，拼接"</w>"表示一个分子词位于一个完整单词末尾。因为在将分子词合并回单词时，需要标志物区分下一个分子词是否和当前属于同一个单词：

a) 如果当前分子词不是以"</w>"结尾：是同一个单词，就直接合并；

b) 如果以"</w>"结尾：不是同一个单词，以英语为例，就是将"</w>"替换为空格；

因此再添加各基本字符+"</w>"，同样是 256 个；

(5) 除此之外，CLIP 团队使用大量语料统计得到了最常见 262144 个分子词对，取前面最常见的 48894 个分子词加入到字典；

(6) 最后，添加句子开始标志："<|startoftext|>"，句子结束标志："<|endoftext|>"。

tokenizer 的文字编码流程如下：

(1) 对于输入的一个句子，首先依据空白字符（如空格、tab、换行等）、标点符、数字等进行单词划分，得到单词序列；

(2) 然后将单词序列使用 tokenizer 进行分子词划分，并将分子词转化为分子词在字典中的序号 index，得到 index(分子词)序列；

(3) 在 index(分子词)序列前后分别添加句子开始标志序列（49406）和句子结束标志的序列（49407），得到 token 序列；

(4) 对于 token 序列短于 77 的句子，用 0 填充补齐长度至 77，对于 token 序列长于 77 的句子，直接将超出的部分舍去不要得到模型的文字输入；

将得到的文字输入送入文本编码器，如图 3.6 所示，其内部流程如下：

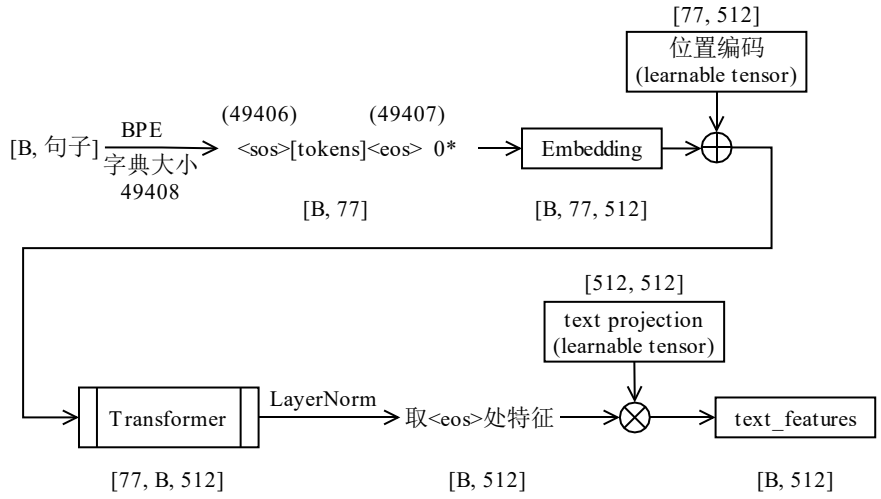

图 3.6 CLIP 的文本编码器流程

(1) 文字输入形状为[77]，在文本编码器中，首先通过 Embedding 层，映射为形状[77, 512]；

(2) 然后加上位置编码（CLIP 中采用可学习的矩阵），送入 Transformer；

(3) Transformer 输出特征矩阵形状为[77, 512]，首先进行 layer norm，然后取句子结束标志处特征作为该句子的特征，形状变为[512]；

(4) 最终通过 text projection（一个可学习矩阵）映射，得到形状[512]的 text feature。

3.2.2.4 图像编码器

使用的 CLIP 的图像编码器是一个 ViT-B/32。

图像预处理流程为：

(1) 对于输入的一张 RGB 彩色图片，首先将其最小边缩放为 224，另外一边按照比例计算出相应长度，然后中心裁剪出边长为 224 的正方形，得到的图片形状为[3, 224, 224]，每个元素均为 uint8（该颜色通道上的像素值）；

(2) 然后将各 uint8 值映射成 0~1 之间的浮点数（float32），并且进行正则化（将 3 个通道的均值调整为 0.48145466, 0.4578275, 0.40821073，将 3 个通道的标准差调整为 0.26862954, 0.26130258, 0.27577711），得到模型的图像输入；

将得到的图像输入送入图像编码器，如图 3.7 所示，其内部流程如下：

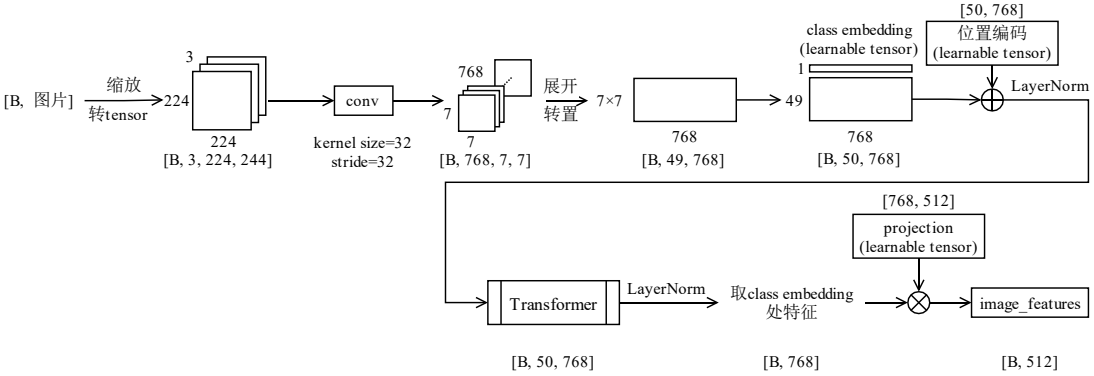

图 3.7 CLIP 的图像编码器流程

(1) 图像输入形状为[3, 244, 244]，在 ViT-B/32 中，首先通过 kernel size=stride=32 的、3→768 通道的卷积，映射为[768, 7, 7]，这就实现了 ViT 中将一张图片划分为大小 32×32 图像块的目的；

(2) 然后将最后 2 个通道（7×7）展开，得到特征矩阵形状为 [768, 49]，其中 49 即为一张图片划分得到的图像块数量，可视为该图片的特征向量通道数（channel），768 可视为一个图像块的特征向量长度（feature）；因此再次 channel 和 feature 维度转置，得到一张图片对应特征矩阵形状为 [49, 768]；

(3) 对于每个图像，类似于自然语言处理中使用句子结束标志处的特征作为一个句子的特征，ViT 在 49 个通道前拼接一个通道（class embedding，可学习矩阵）成为[50, 768]，并在最后用该 class embedding 处的特征向量作为该图像的特征向量；

(4) 然后加上位置编码（CLIP 中采用可学习的矩阵），进行一次 layer norm 后送入 Transformer；

(5) Transformer 输出特征矩阵形状为[50, 768]，首先进行 layer norm，然后取 class embedding 处特征作为该图像的特征，形状变为[768]；

(6) 最终通过 projection（可学习矩阵）映射，得到形状[512]的 image feature。

3.2.2.5 文本图像检索

流程示意如图 3.8 所示，文本编码器输出得到文本特征（text feature，当是批处理时即为 text features），图像编码器输出得到图像特征（image feature，当是批处理时即为 image features），首先将文本特征、图像特征进行 L2 正则化。

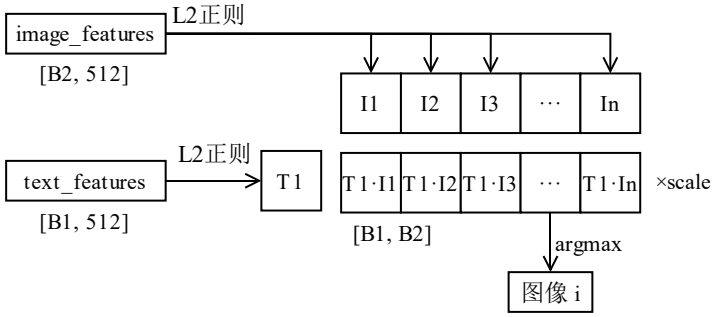

图 3.8 CLIP 文本图像检索流程示意图

当使用文本检索图像时，文本特征矩阵与图像特征矩阵的转置相乘，即可得到文本-图像相似度矩阵，每行对应该句文本与各图像的相似度，在一行中取相似度最大值对应的图像，即可作为该文本检索到的图像。

当使用图像检索文本时，图像特征矩阵与文本特征矩阵的转置相乘，即可得到图像-文本相似度矩阵，每行对应该张图像与各句文本的相似度，在一行中取相似度最大值对应的文本，即可作为该图像检索到的文本。

当使用批处理时，若输入的文本 batch size 为 B1、输入的图像 batch size 为 B2，那么得到的文本-图像相似度矩阵形状为[B1, B2]、图像-文本相似度矩阵形状为[B2, B1]。

3.2.3 CLIP4Clip 结构与实现

实验中使用相似性计算模块采用 meanP 的 CLIP4Clip，其文本编码器使用 CLIP ViT-B/32 版本的文本编码器，其视频编码器使用 CLIP ViT-B/32 版本的图像编码器。其中，CLIP4Clip 的文本编码器和 CLIP 的文本编码器结构、工作流程完全相同，CLIP4Clip 的视频编码器和 CLIP 的图像编码器结构完全相同。

3.2.3.1 视频编码器

视频加载流程为：

(1) 对于一个视频，惯例方法是按照关键帧选取方案抽取得到 k 张关键帧图片（例如按照帧间差排序选取帧间差最大的、按照时间间隔平均抽取，将在后续详细介绍关键帧选取方案及实验），实验中设置一个视频的关键帧图片数量为 12；

(2) 然后将所有关键帧图片使用 CLIP 中的图片预处理流程进行缩放、中心裁剪、转为 tensor、正则化，若一个视频的关键帧数量小于 12，则用全 0 矩阵补齐至数量为 12，若一个视频的关键帧数量大于 12，则从中随机选取 12 个，得到矩阵形状为[batch size, 12, 3, 244, 244]；

(3) 因为 CLIP 图像编码器的输入矩阵是 4 维，因此将关键帧数量 12 合并到 batch size 一维中，最终得到模型的视频输入，形状为[batch size×12, 3, 244, 244]。

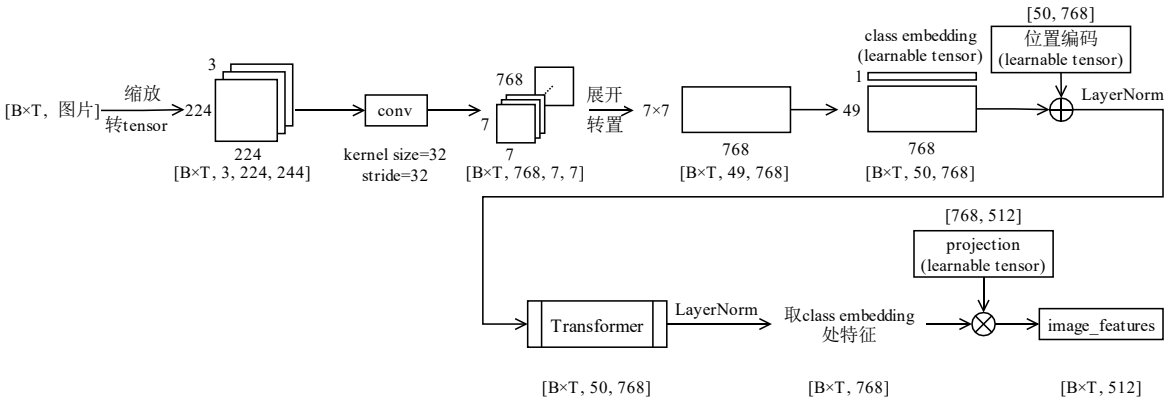

图 3.9 CLIP4Clip 的视频编码器流程

因为 ViT 内部 batch size 维度上各矩阵无信息交互，且所有视频关键帧数量都相同，所以只需[batch size×12]拆为[batch size, 12]即可恢复。这样，就实现了 CLIP4Clip 的视频输入和 CLIP 的图片输入维度相同，直接将视频输入送入视频编码器 ViT 即可，如图 3.9 所示，其内部流程与 CLIP 的图像编码器一致，视频多出的时间维度 T（T＝12）始终合并在 batch size 中，最终获得视频特征 image features 形状为 [batch size×12, 512]。

3.2.3.2 文本视频检索

文本编码器输出得到文本特征 text features（形状为[batch size, 512]）进行 L2 正则化。视频编码器输出得到视频特征 image features，形状为[batch size×12, 512]，首先需将时间维度 T（T＝12）分离得到[batch size, 12, 512]，然后按照相似性计算模块 meanP 方法在时间维度进行平均池化，即将每个视频对应的 12 条[512]特征向量取算术平均，即可得到最终视频特征 video features（形状为[batch size, 512]），并进

行 L2 正则化。

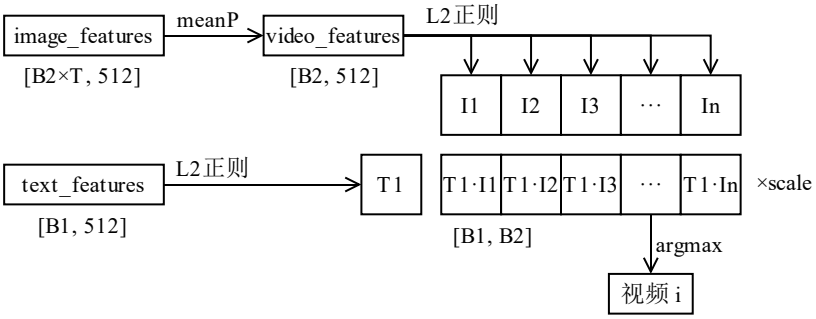

图 3.10 CLIP4Clip 文本视频检索流程示意图

当使用文本检索视频时，文本特征矩阵与视频特征矩阵的转置相乘，即可得到文本-视频相似度矩阵，每行对应该句文本与各视频的相似度，在一行中取相似度最大值对应的视频，即可作为该文本检索到的视频。

当使用视频检索文本时，视频特征矩阵与文本特征矩阵的转置相乘，即可得到视频-文本相似度矩阵，每行对应该段视频与各句文本的相似度，在一行中取相似度最大值对应的文本，即可作为该视频检索到的文本。

当使用批处理时，若输入的文本 batch size 为 B1、输入的视频 batch size 为 B2，那么得到的文本-视频相似度矩阵形状为[B1, B2]、视频-文本相似度矩阵形状为[B2, B1]，流程示意如图 3.10 所示。

3.2.4 AIM 结构与实现

对于 CLIP4Clip 的 ViT，因为特征宽度为 768（ViT 的网络宽度），一个 Adapter 只含有约 768×192×2 个参数，因此添加的总参数量为 1430 万，而原 CLIP4Clip 总参数量为 1 亿 5000 万，训练参数量较全微调减少至 9.5%；而与 CLIP4Clip 的 seqTransf 相似性计算模块相比，其是一个 6 层、宽度 512 的 Transformer，同样有约 3800 万参数量，只有其 37.8%。

3.2.4.1 Adapter 结构

Adapter 作为低参数量微调，其重要目标之一是使用较少的参数量，因此与全连接前馈层（FFN）相比，其隐藏层维度是输入维度的 4 倍下采样，而非 4 倍上采样。其结构为：全连接层 FC1 四倍下采样→激活层 GeLU→全连接层 FC2 四倍上采样。结构如图 3.11 所示。

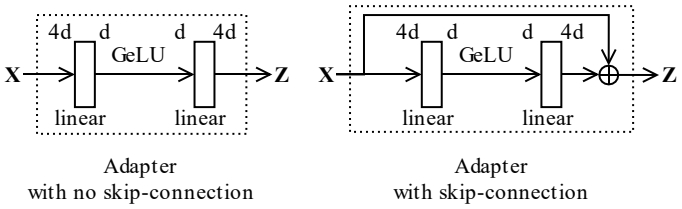

图 3.11 Adapter 结构图

将 Adapter 输入记为x，那么当不使用跳跃连接时 Adapter 层表示函数见公式(3.2)，使用跳跃连接时 Adapter 层表示函数见公式(3.3)：

$$Adapter_{no\ skip}(x) = FC2\left(GeLU\left(FC1(x)\right)\right) \tag{3.2}$$

$$Adapter_{skip}(x) = x + FC2\left(GeLU\left(FC1(x)\right)\right) \tag{3.3}$$

3.2.4.2 Adapter 插入

原始 ViT 的 Residual Attention Block 由 1 个 MSA 层、1 个 FFN 层组成，网络输入 x 形状是[B×T, N+1, D]，其中 B 是 batch size，T 是一段视频取的关键帧数量数（T=12），N 是一张图片的图像块数量（N=49），D 是特征宽度（D=768）。Adapter 插入方式如下：

(1) Temporal Adapter，插入到 MSA 层之后。先将 x 转置为[B×(N+1), T, D]并进行 layer norm，然后依次输入 MSA 和 Temporal Adapter，然后与 x 相加得到 xt（因为 ViT 的 MSA 旁有跳跃连接）；

(2) Spatial Adapter，插入到复用的 MSA 层之后。先将 xt 转置回[B×T, N+1, D]，然后依次输入 MSA 和 Spatial Adapter，然后与 xt 相加得到 xs（因为 ViT 的 MSA 旁有跳跃连接）；

(3) Joint Adapter，串联到 FFN 旁边。先将 xs 进行 layer norm，然后分别输入到 Joint Adapter 和 FFN，将 2 个输出之和 xs 相加得到最后的输出 z（因为 ViT 的 FFN 旁有跳跃连接）。

为了使修改后的模型初始化时模型输出与原始模型一致[13]，所以所有 Adapter 内 FC2 权重初始化为 0；Temporal Adapter 和 Joint Adapter 不使用跳跃连接；Spatial Adapter 使用跳跃连接。

这样，初始化状态下，Temporal Adapter 的输出为 0，将 Temporal 信息屏蔽，原始 x 进入 S-MSA；Spatial Adapter 的输出为 0，Spatial Adapter 内的跳跃连接将 S-MSA 的输出直接传递到 FFN；Joint Adapter 的输出为 0，只有 FFN 的输出进入下一层。修改后的 Residual Attention Block 输入 x 和输出 z 关系表示见公式(3.4)：

$$x \in \mathbb{R}^{[B\times T,N+1,D]},\ x^T \in \mathbb{R}^{[B\times(N+1),T,D]}$$
$$xt = Adapter_{no\ skip}\left(MSA\left(LN(x^T)\right)\right) + x$$
$$xs = Adapter_{skip}\left(MSA(xt^T)\right) + xt$$
$$xn = LN(xs)$$
$$z = Adapter_{no\ skip}(xn) + FFN(xn) + xs \tag{3.4}$$

结构示意图如图 3.12 所示。

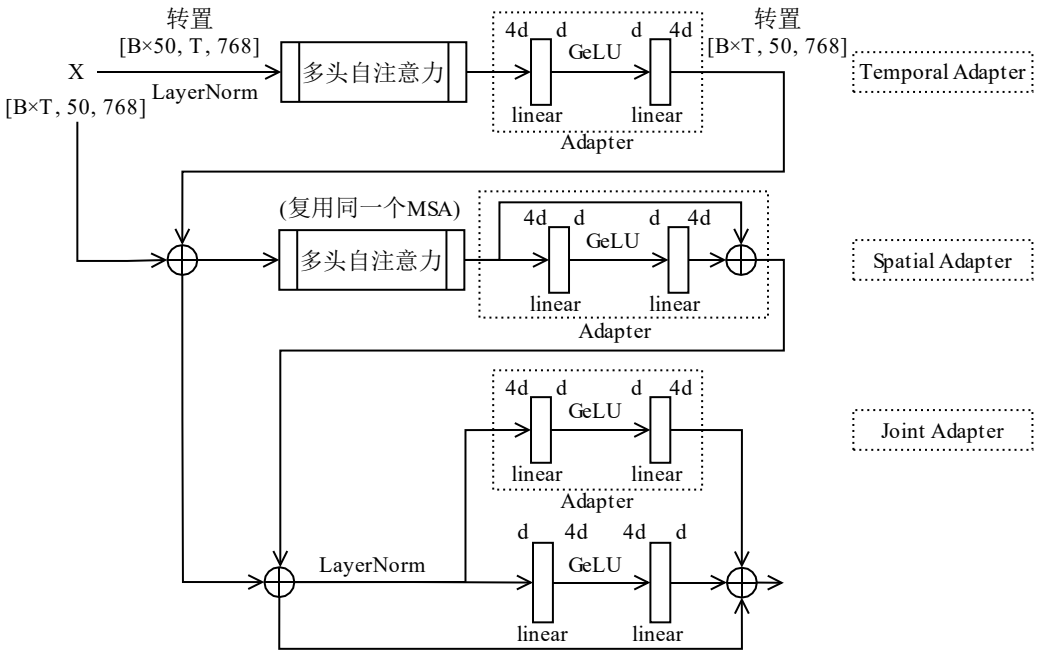

图 3.12 插入 Adapter 后的 Residual Attention Block 结构图

另外，因为要使网络学习时间学习，AIM 论文中还在 ViT 中添加了时间编码（使用可学习矩阵），在原 ViT 中 x 添加位置编码时一同添加。

3.3 关键帧选取与保存

实验中，将关键帧保存为图片可以极大加快数据加载速度，batch size 为 32 时可将训练时间缩短为 6.8%；但出乎意料的是，最大帧间差选取方案训练得到的模型效果不敌平均选取方案，其他条件相同时，全微调的 CLIP4Clip 召回率 R@1 从 42.0% 降低为 38.9%。

3.3.1 平均选取方案

这是 CLIP4Clip 论文以及其他大多数视频文本检索论文中使用的帧选取方式，按时间平均抽取：

(1) 每个视频首先按秒划分，取每秒的第 1 帧；

(2) 短于 12 秒的视频关键帧数量不足 12（在 DataLoader 中用全 0 矩阵补齐至数量为 12）；

(3) 长于 12 秒的视频关键帧数量大于 12（在 DataLoader 中从中随机选取 12 个）。

3.3.2 最大帧间差选取方案

因为考虑到也许按时间平均抽取可能会无法取到能体现画面重大变化的帧，所以想探索一下其他关键帧提取方案：将所有帧按照帧间差从大到小排序，因为帧间差大表示该帧与上一帧有较大差距，那么这也就意味着视频画面出现了重大变化，因此主观上认为，这些帧能更好地集中体现视频内容：

(1) 先将视频中心裁剪为正方形以更好地贴近模型所看到的视野，然后遍历视频内所有帧；

(2) 首先将画面进行一次高斯模糊，以降低噪点的影响；

(3) 然后计算与前一帧的帧间差，将帧间差求和，并记录当前是第几帧（用于排序后将目标帧恢复回原来的时间顺序）；

(4) 然后将所有帧按照帧间差之和从大到小排序，取前 12 个即为目标帧；

(5) 将目标帧再按照其是第几帧从前往后排列，依次到原视频里取出，即得到关键帧。

3.3.3 关键帧保存

在实验中对硬件的使用率进行检测发现，若 DataLoader 直接以视频为起点、选取关键帧、将关键帧使用 CLIP 的图片预处理流程转为模型输入，那么绝大部分时间都用于进行数据加载了，GPU 长时间处于空闲状态，这表明视频数据加载成了整个模型训练的性能瓶颈。

又因为对于同一个视频，同一种关键帧选取方案得到的关键帧总是相同的，因此考虑事完成关键帧选取，缩放和中心裁剪成为边长 224，并将结果作为图片文件（JPG）保存起来。

这样 DataLoader 就只需要加载图片并直接转为 tensor 即可，将 OpenCV 库解析视频、关键帧选取、三线性插值的图片缩放、图片裁剪等所有 DataLoader 中需要消耗大量 IO 和 CPU 资源的过程全部去除了，可以极大地缩短 DataLoader 所耗费的时间，而且减少了很多重复计算。

以本实验的设备上运行时间为例，batch size 为 32 时 CLIP4Clip 的全微调一次

step：

(1) 不使用关键帧保存方案耗时约 5.99s；

(2) 使用关键帧保存方案耗时约 0.41s，耗时约为原来的 6.8%；

可见关键帧保存方案可以极大地加快训练速度。

再考虑硬盘空间占用情况，以 MSR-VTT-100K 数据集为例，使用最大帧间差选取方案：

(1) 原始视频总大小为 6.25GB；

(2) 当用 JPG 格式保存时，抽取到关键帧图片总大小为 1.87GB，约为原来的 29.9%；

因此同样可以非常有效地减小数据的硬盘占用。

3.4 向量数据库

向量数据库核心目的是保存视频特征，从而让系统不需要每次启动时都重新计算一次视频库的视频特征，充分利用模型双塔结构的特点，实现检索速度的提升。

除此之外，为实现其作为一个“数据库”的本质，使用了可以存储多种矩阵的设计，通过提供数据库名，可以存储和加载多个矩阵及其索引，从而让其也可用于测试集召回率计算、基于模型输出的下游网络训练等过程的加速。

3.4.1 对外接口

向量数据库有两个对外接口“store”和“load”，分别用于向量的保存和加载；底层操作使用 pandas 库实现，所有 DataFrame 相关数据、操作对外界透明，抽象为 numpy 库矩阵（array）的存储和访问。

store 只在向量数据库构建中使用，传入数据库名（database_name）和想要保存的矩阵，即可完成存储；load 只在文字视频检索（search）中使用，只需传入数据库名（database_name），模块即可自动将保存的文件转化回保存时的数据并返回。

3.4.2 数据库名

数据库名（database_name）的命名规则是“数据集名+关键字”，关键字包括 test、raw、train、video、text，含义如下：

(1) test 表示矩阵是数据集的测试集的特征；

(2) train 表示矩阵是数据集的训练集的特征；

(3) raw 表示矩阵中的视频特征未使用相似度计算模块融合，是视频编码器输出

的 12 帧关键帧的图像特征；

(4) video 表示传入矩阵只有视频特征和对应视频编号，没有文本特征；

(5) text 表示传入矩阵只有文本特征和对应视频编号，没有视频特征；

不同数据库名及对应传入矩阵含义如表 3.1 所示，其中：

表 3.1 数据库命名及对应数据项

数据库名	数据项
数据集名_test	video_id, text_feature, video_feature
数据集名_test_raw	video_id, text_feature, video_feature_raw
数据集名_train_video	video_id, video_feature
数据集名_train_video_raw	video_id, video_feature_raw
数据集名_text	video_id, text_feature

(1) video_id 表示视频编号矩阵。视频编号是一个整数，且所有数据库中的所有视频，在整个系统中都有其唯一且固定的视频编号；

(2) text_feature 表示文本特征矩阵，且已经过 L2 正则；

(3) video_feature 和 video_feature_raw 表示视频特征矩阵。video_feature 已经过相似度计算模块融合和 L2 正则化，带有 raw 关键字的则未使用相似度计算模块融合。

3.4.3 数据保存方式

关于 Database 要用什么来进行操作、用什么文件形式保存、以什么数据格式保存等问题，做了以下实验，最终发现使用 Pandas 库的 DataFrame 操作、用 pickle 存储 numpy.dnarray 具有最快的速度，因此才选定最终 Database 的数据保存方式。

表 3.2 不同存储方式的耗时与文件大小

存储方式	时间/s	文件大小/MB	备注
方式 1	1.91	253	正常运行
方式 2	0.83	253	正常运行
方式 3	∞	≥25.1GB	运行，MemoryError 终止
方式 4	0.72	254	正常运行

四种保存方式分别为：

(1) 方式 1：pickle 保存 torch.tensor；

(2) 方式 2：torch.tensor 转为 numpy.array 后，再 pickle 保存；

(3) 方式 3：Pandas DataFrame 中存储 torch.tensor，然后 to_pickle 保存；

(4) 方式 4：torch.tensor 转为 numpy.array 后，Pandas DataFrame 中存储 numpy.array，然后 to_pickle 保存。

使用模拟数据量的数据类型和矩阵形状如下，共计 266.28 million 字节：

(1) video_id：numpy.array[int]，[10000]；

(2) text_feature：torch.tensor(torch.float32)，[10000, 1, 512]；

(3) video_feature：torch.tensor(torch.float32)，[10000, 12, 512]。

使用的测试操作为：全部数据进行一次保存、一次加载、转回 torch.tensor 后、进行 1 次切片、2 次索引、一次[512]@[512]矩阵乘法。

实验结果如表 3.2 所示：

(1) 方式 1、方式 2 对比，可见将 torch.tensor 转为 numpy.array 可以明显加快数据的存取速度；

(2) 方式 4 较方式 2 相比，额外添加了 pandas 库来将矩阵拆分为 10000 个特征向量，可以进一步缩短耗费时间，且文件大小也只增大了 1MB，综合性能是四种方案里最优的；

(3) 方式 3 用 pandas 的 DataFrame 直接存储 torch.tensor 显然无法实现了，触发了某种错误导致存储文件数据不断增长，于是最终导致触发 MemoryError 使异常程序终止。

另外，Pandas 保存为 feather 文件可以进一步提高数据存储、读取速度，但 feather 文件格式不支持保存矩阵，因此不适用；而对于快速向量检索框架（例如 Facebook 发布的向量数据库 FAISS），在 10 万以下的数据量级上，直接用所有向量相乘或遍历的时间开销仍然很低，所以没有使用向量检索框架的必要性。

3.5 算法测试

3.5.1 实验条件与参数设置

3.5.1.1 数据集

实验主要使用的是 MSR-VTT-10K 数据集[28]，其在文字视频检索任务中常用的是 9k 训练集[29]、1k 测试集的划分[30]，是一个相对视频种类更丰富、文本描述更完善的数据集。

MSR-VTT 从 YouTube 上收集了 20 个类别的 7180 个视频，每个视频分割成若干

视频片段后，人工从每个视频选出不超过 3 个片段以保障多样性。最终 10000 个片段，每个片段时长 10~30 秒，总时长 41.2 小时。然后聘用 1317 个 AMT 工人（Amazon Mechanical Turk）给每个片段人工写文字描述，去除过短和重复的后每个片段剩 20 句文字描述，因此数据集一共 1 万个视频片段，20 万个文本视频对。

此外，还在 MSVD[31]数据集上测试了模型性能。其他常用数据集中，LSMDC 因为网络环境原因暂时无法下载，ActivityNet 和 DiDeMo 需要更高的 GPU 算力实验条件暂时不允许。

3.5.1.2 实验条件

内存 24GB 的 NVIDIA GeForce RTX 3090 GPU 显卡，1 张。

3.5.1.3 batch size

在 CLIP4Clip 论文中，各数据集上 batch size 均为 128，MSR-VTT、MSVD、LSMDC 数据集上每个视频关键帧数量设为 12，需要使用 4 张 NVIDIA Tesla V100 GPU；ActivityNet、DiDeMo 数据集上每个视频关键帧数量设为 64，需要使用 16 张 NVIDIA Tesla V100 GPU。

为控制变量视频关键帧数量同样设为 12；因为 CLIP4Clip 论文参数设置下，在每张显卡上分到的 batch size 都是 32，所以未说明的情况下本实验 batch size 设为 32。

3.5.1.4 其他参数

为控制变量，优化器、训练、模型参数等与 CLIP4Clip 论文基本保持一致。在未说明的情况下：

(1) 优化器使用 Adam[32]；

(2) 调度器使用 CLIP 中的余弦退火；

(3) 位于文本编码器和视频编码器内的参数学习率为 1e-7；

(4) 损失函数使用 CLIP4Clip 中的对称交叉熵；

(5) 全微调时 epochs 为 5，低参数量微调时提前在第 2 个 epoch 完成后终止（经多次实验统计，学习率为 1e-7 时，在低参数量微调过程中模型在第 2 个 epoch 内部达到收敛，在测试集上召回率最高）。

(6) 模型的文本 token 序列最大长度为 77，视频最大关键帧数量为 12，预训练模型为 CLIP ViT-B/32 版本。

3.5.2 耗时与可调参数

面对实验条件限制和预训练模型规模不断增大这两点现实因素，切切实实地缩短

训练所需时间、降低训练所需内存开销是本次实验中的一项重点目标和内容。在本实验条件下，关键帧保存方案和 AIM 低参数量微调结合，可将训练速度提升为 34 倍。

3.5.2.1 数据

正如在关键帧保存中所介绍的，关键帧保存方案通过加快数据加载，可以极大地提高训练速度，且非常有效地减小数据的硬盘占用；Adapter Tuning 通过减少可调参数量，可以有效地加快训练速度。

表 3.3 训练耗时与可调参数量

	CLIP4Clip	CLIP4Clip +帧保存	AIM +帧保存
耗时/s/step	5.99	0.41	0.44
epoch	5	5	2
可调参数量/M	151.28	151.28	14.21
耗时× 相对比例	100%	6.84%	2.94%
epoch 相对倍率	1.00	14.61	34.03

表 3.3 中记录了其他参数均为默认参数情况下，不同情况的耗时与可调参数量信息。虽然硬件设备、运行环境、温度散热等众多因素都会影响具体数值，但其值、尤其是相对比例仍具有重要参考作用。

首先说明表中各列含义：

(1) “CLIP4Clip”：使用不插入 Adapter 的模型进行全微调，DataLoader 从原始视频加载数据；

(2) “CLIP4Clip+帧保存”：使用不插入 Adapter 的模型进行全微调，使用帧保存方案，DataLoader 直接加载保存好的关键帧图片；

(3) “AIM+帧保存”：使用插入 Adapter 的模型进行低参数量微调（只训练新插入的 Adapter），使用帧保存方案，DataLoader 直接加载保存好的关键帧图片；

表中各行含义：

(1) “耗时”：在默认 batch size 下，每个 step 平均耗时秒数（1 个 step 表示进行 1 次 batch size 大小的网络训练，因此每个 epoch 内有 180000÷batch size 个 step）。训练时日志每次 100 个 step 输出一次平均耗时，表中结果为去除一个最大值、去除一个最小值后的平均数；

(2) “epoch”：完成训练所需要的 epoch 数；

(3) “可调参数量”：整个模型中没有冻结的参数的总个数，单位为百万；

(4) “耗时×epoch 相对比例”和“耗时×epoch 相对倍率”：因为 batch size、训练集大小都相同，所以总 step 数也相同，所以耗时与 epoch 之积与训练全过程所需时间成正比。再以“CLIP4Clip”为基准，计算出各列的相对比例，直接体现出各列的时间节省情况。

3.5.2.2 结论

由表中数据可得到以下结论：

(1) 1、2 列的耗时对比，体现出帧保存方案可以极大地提升训练速度，都是使用 CLIP4Clip 进行全微调时，帧保存方案将训练时每 step 耗时从 5.99 秒降低到 0.41 秒，只有原来的 6.84%，相当于速度提升为 14.6 倍；

(2) 2、3 列的耗时对比，AIM 使得模型的层数增加，使得耗时稍有增加，从 0.41 秒增加为 0.44 秒，但速度仍然达到了第一列的 13.6 倍；

(3) 但 2、3 列还有重要的一点是，因为可调参数量从 151M 减少为 14M，为原来的 9.4%，使得能够提前收敛，从而训练所需的 epoch 从 5 减少为 2。所以使用 AIM 后，一共耗费的时间从 6.84%个单位时间缩短为 2.94%个单位时间，相当于速度提升为 2.3 倍（达到第一列的 34.0 倍）。

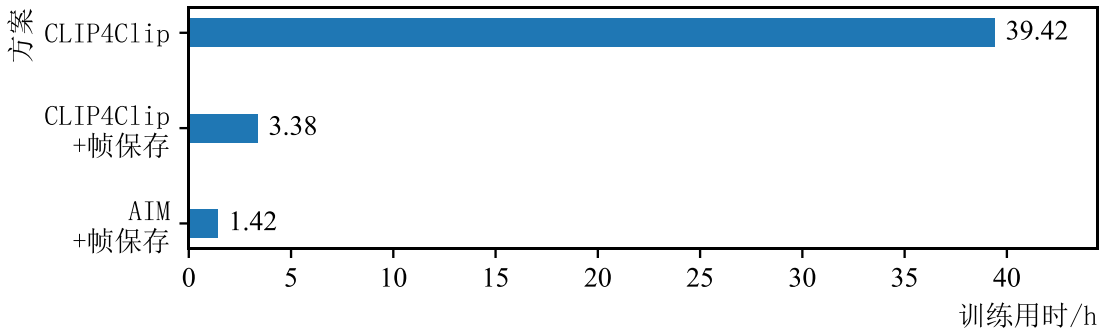

图 3.13 各方案训练实际耗时

如图 3.13 中的训练用时所示，34 倍的速度提升，体现在现实情况中：使用 CLIP4Clip、从视频加载数据、全微调训练一次耗时预期约为 48 小时（因为每个 epoch 结束后还要进行测试集上召回率计算用于评估模型），也因为实在太久了，在用此方案训练时将 batch size 设置为 64，将 48 小时缩短为实际 39 小时 25 分钟；而使用 AIM 和关键帧保存方案训练一次实际只花费 1 小时 25 分钟。

在有限的硬件条件下，只有以次此基础才能使得后续模型结构、关键帧提取方案、学习率调整等各种实验的进行成为可能。

文字视频检索任务中一般使用召回率来评估模型的性能：召回率 R@K（数值越大越好）、中位数排名 MdR（数值越小越好）、平均排名 MnR（数值越小越好）。其中 R@K 计算在前 K 个检索结果中有多少 ground truth 被找到，例如使用 R@1、R@5 和 R@10；中位数排名 MdR 计算所有 ground truth 的排名中位数；平均排名 MnR 计算所有 ground truth 的平均排名。

下面将就关键帧选取方案、AIM 有效性、提升性能、MSVD 数据集得分等几方面介绍所进行的实验。

3.5.3 关键帧选取方案

其他参数相同情况下，分别使用平均帧选取方案、最大帧间差选取方案，进行 CLIP4Clip 全微调来，测试两种关键帧选取方案对训练出的模型性能的影响，结果如表 3.4 所示。

表 3.4 不同关键帧选取方案训练出的模型性能

训练集	测试集	R@1↑	R@5↑	R@10↑	MdR↓	MnR↓
最大帧间差选取	最大帧间差选取	38.9	64.5	76.7	2	20.0
最大帧间差选取	平均选取	41.3	67.9	77.4	2	17.5
平均选取	最大帧间差选取	39.6	66.1	75.9	2	19.9
平均选取	平均选取	42.0	68.9	79.6	2	16.2

在各自对应的数据集上，平均帧选取相较帧间差抽取的 R@1 从 38.9%提升为 42.0%，可见训练集使用平均帧选取方案抽取关键帧时，训练出来的模型性能更好；而且如果测试集使用平均帧选取方案抽取关键帧，那么模型得分也比使用最大帧间差选取方案时的更高。因此，平均帧选取方案比最大帧间差选取方案更好，后续实验中的模型若无特别说明，那么其训练集上关键帧选取方案均采用平均帧选取方案。

造成这种现象的原因，推测之一是关键帧之间差异增大，导致模型拟合能力不足，或是样本波动过大；推测之二是关键帧之间时间间隔从 1 秒变成了随机分布，从而使得模型无法很好地学习时间信息。当然，实际原因还需更多的实验或相关资料佐证，有待进一步探究。

3.5.4 AIM 有效性

首先，测试 AIM 低参数量微调是否可达到全微调的效果，以 CLIP zero-shot 和全微调的 CLIP4Clip 得分为基准。表 3.5 中记录了训练不同模块时，获得的模型的性能。

表 3.5 训练不同模块时的性能

训练的模块	R@1↑	R@5↑	R@10↑	MdR↓	MnR↓	可调参数量/M	训练用时/h
CLIP zero-shot	31.2	53.7	64.2	4	-	-	-
CLIP4Clip	41.7	68.7	79.6	2	16.6	151.3	39.4
AIM	37.9	63.7	74.7	2	20.3	14.2	1.4
文本编码器+AIM	41.5	70.2	79.9	2	16.2	77.6	1.5

第 1 行是原始的 CLIP 预训练模型，不经任何微调直接 zero-shot。R@1 为 31.2%，此数据来自 CLIP4Clip 论文[7]，作为基准模型。

第 2 行是 3.2 节中“CLIP4Clip”方案训练出的 CLIP4Clip。R@1 为 41.7%，除了 batch size（改为 64）和 GPU 数量（改为 1）的调整，代码及运行命令均与 CLIP4Clip 论文[7]发布的一致，未进行任何改动，所以将此模型作为基准模型。

第 3 行是在视频编码器中插入 Adapter 的 AIM，且只训练新插入的 Adapter。R@1 为 37.9%，较 CLIP 提升了 6.7%，但还是比 CLIP4Clip 低了 3.8%。推测有可能是因为整个模型中只有视频编码器进行了微调，而文本编码器没有微调，导致整体的性能无法达到全微调。

为证实这一推测，将视频编码器中插入的 Adapter、和文本编码器两部分都进行调整。如第 4 行所示，与 CLIP4Clip 相比，R@1 基本相同，只低了 0.2%，而 R@5、R@10、Mean R 等指标的性能分别有 1.5%、0.4%、0.3%的少量提升，因此推测成立。同时，因为使用的是 AIM 代替视频编码器的全微调，所以也证明了 AIM 的有效性。

3.5.5 提升性能

既然 AIM 有效，低参数量微调可以达到全微调的效果，考虑到 AIM 的设计有专门针时间信息学习的 Temporal Adaptation，所以探究：AIM 是否可以用于进一步提升模型整体性能。

实验思路是：假设全微调得到的 CLIP4Clip 对于时间信息的处理能力仍有进步空间，所以以该 CLIP4Clip 的参数为预训练模型，加入 AIM 进行低参数量微调，看模型性能能否有提升。

3.2 节中“CLIP4Clip”方案进行了 5 个 epoch 的全微调，每个 epoch 结束后都会保存一次模型作为 checkpoint，第 5 个 epoch 后得到的模型 R@1 是 41.7%，但这并不是 5 个 checkpoint 里的最佳得分；得分最高的，是第 2 个 epoch 后得到的模型，其 R@1 是 42.2%。

将第 5 个 epoch 后得到的 checkpoint 记为 CLIP4Clip-5，第 2 个 epoch 后得到的 checkpoint 记为 CLIP4Clip-2，分别将其作为预训练模型，加入 AIM 进行低参数量微调，得到的模型性能如表 3.6 所示：

表 3.6 使用 AIM 提升模型性能

训练的模块	预训练模型	lr	R@1↑	R@5↑	R@10↑	MdR↓	MnR↓
CLIP4Clip-5	CLIP	1e-7	41.7	68.7	79.6	2	16.6
CLIP4Clip-2	CLIP	1e-7	42.2	70.2	80.9	2	15.3
AIM	CLIP4Clip-5	1e-7	42.5	69.1	79.2	2	17.1
AIM	CLIP4Clip-2	1e-7	43.1	71.2	80.1	2	16.3
AIM	CLIP4Clip-2	5e-8	43.4	71.1	80.0	2	16.3

(1) 第 3 行，以 CLIP4Clip-5 为预训练模型的 AIM 的 R@1 从 41.7%提升为 42.5%，增加了 0.8%，R@5 增加了 0.4%；

(2) 第 4 行，以 CLIP4Clip-2 为预训练模型的 AIM 的 R@1 从 42.2%提升为 43.1%，增加了 0.9%，R@5 增加了 1.0%；

(3) 第 5 行，还测试了一组修改学习率为 5e-8 的 AIM，同样以 CLIP4Clip-2 为预训练模型，R@1 从 42.2%提升为 43.4%，增加了 1.2%，R@5 增加了 0.9%。这是所有实验组中 R@1 召回率最高的。

可见，AIM 确实也可用于提升模型性能，虽然提升幅度有限，但考虑到其训练只需要花费 1.4 小时，也不失为一种实验条件有限情况下的不错选择。

3.5.6 MSVD 数据集得分

表 3.7 AIM 在 MSVD 数据集上的得分

模型	预训练模型	R@1↑	R@5↑	R@10↑	MdR↓	MnR↓
CLIP zero-shot	CLIP	36.3	63.3	73.2	3	20.6
AIM	CLIP	42.5	72.0	81.7	2	12.4
CLIP4Clip zero-shot	CLIP4Clip	43.4	72.8	82.5	2	12.0
AIM	CLIP4Clip	44.4	74.5	84.2	2	11.0

最后，在 MSVD 数据集上，以 CLIP 和在 MSR-VTT 上训练得到的 CLIP4Clip 为基准模型，简单地测试了一下 AIM 的得分，如表 3.7 所示：

(1) 第 1 行是原始的 CLIP，不经任何微调直接 zero-shot，以其作为基准；

(2) 第 2 行是以 CLIP 为预训练模型、视频编码器中插入 Adapter 的 AIM，只训

练新插入的 Adapter。R@1 从 36.3%提升为 42.5%，相较于 CLIP 增加了 6.2%，R@5 增加了 8.7%；

(3) 第 3 行是在 MSR-VTT 上训练得到的 CLIP4Clip，不经任何微调直接 zero-shot，以其作为基准；

(4) 第 4 行是以在 MSR-VTT 上训练得到的 CLIP4Clip 为预训练模型、视频编码器中插入 Adapter 的 AIM，只训练新插入的 Adapter。R@1 从 43.4%提升为 44.4%，相较于 CLIP4Clip 增加了 1.0%，R@5 增加了 1.7%。

可见，AIM 方案在 MSVD 数据集上展示出了和在 MSR-VTT 数据集上类似的性能，都能提高 CLIP 和 CLIP4Clip 的性能。

本章总结：

首先详细介绍了算法中关键帧选取、关键帧保存、模型、向量数据库四个部分的结构与实现。

实验部分，首先进行了耗时与可调参数的实验，展示了关键帧保存方案和 AIM 方案对训练的加速作用，将训练速度提升了 34 倍；然后进行了关键帧选取方案的实验，结果显示平均选取方案比最大帧间差选取方案更有利于提升模型性能；然后继续实验证明了 AIM 方案在视频文本检索领域上的有效性；并实验展示了将 AIM 用于对模型性能提升时的效果，将模型在 MSR-VTT 数据集上的 R@1 从 42.2%提升至 43.4%，R@5 从 70.2%提升为 71.1%；最后，在 MSVD 数据集上测试了进行 AIM 低参数量微调的效果，可将 zero-shot 的 CLIP 的 R@1 从 36.3%提升为 42.5%，将在 MSR-VTT 上训练得到的 CLIP4Clip 的 R@1 从 43.4%提升为 44.4%。

4 系统设计与实现

4.1 需求分析

4.1.1 功能性分析

设计的文本视频检索系统，以 MSR-VTT 数据集的测试集作为视频库，主要包括以下几点功能性需求：

(1) 输入文字检索视频。每当用户输入一句文本，就使用这句文本在视频库中进行检索，找到内容相关视频。

(2) 检索结果展示与视频播放。将检索到的视频展示给用户，并且用户点击展示的视频中任意一个时，跳转到该视频的播放界面进行播放。

(3) 检索速度。每次视频检索过程需要很快地完成，应该和常规视频检索平台具有相近的时间数量级。

(4) 可选择展示文字匹配结果作为参考。在当前系统中，为模拟模型在现实情况下的部署情况，视频检索只通过：使用模型直接将文本与视频匹配得到检索结果，即视频库中视频无标题、标签、描述等任何文字形式信息。但作为展示模型，所以额外添加一个模式选项，让用户可以看到：使用用户输入的文本，直接通过与每个视频的（MSR-VTT 数据集提供的）文字描述进行匹配，得到的检索结果。用“文字匹配结果”作为参考，方便用户判断视频库中是否存在符合该文本描述的视频、符合该文本描述的视频是否出现（或缺失）在“视频检索结果”中、“视频检索结果”是否比“文字匹配结果”更全面、“视频检索结果”中视频内容是否更符合文字描述等，从而方便用户对当前模型的性能做出判断。

4.1.2 非功能性分析

(1) 易用性。应确保系统的功能界面设计清晰、功能按钮布局合理、功能逻辑符合直觉，让用户能简单快速地上手使用。

(2) 美观性。显示界面应该布局清晰合理、排版整齐，且能适应不同窗口尺寸，做出相应的界面布局和排版调整。

(3) 代码可读性。程序代码应逻辑清晰、具有较低的代码重复率、风格统一、做好注释，函数接口应做好输入参数说明、返回值说明。

4.1.3 用例图

程序用例图如图 4.1 所示，参与者“用户”可发起的用例主要分为“搜索视频”、“点击视频”、“开启文字匹配”：

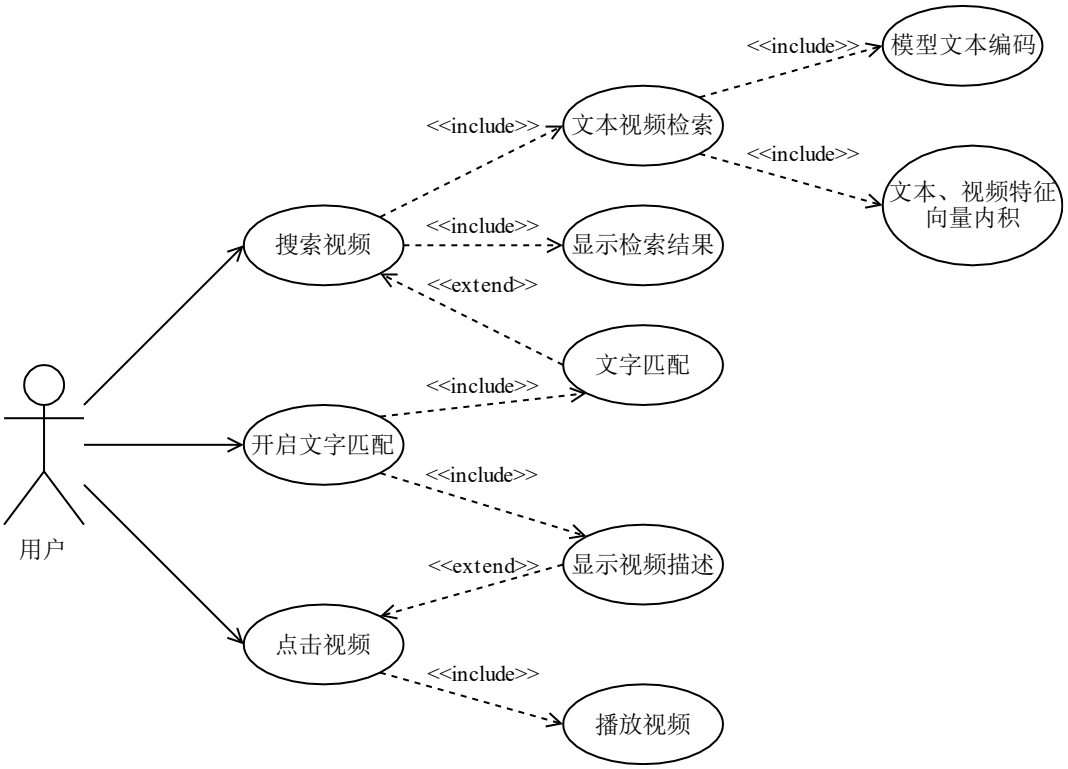

图 4.1 文本视频检索系统用例图

(1) “搜索视频”执行用户搜索视频流程，包含“文本视频检索”、“显示检索结果”两个用例，分别主要由视频检索模块、Web 模块完成；

(2) “点击视频”执行用户点击视频流程，含一个用例“播放视频”，主要由 Web 模块完成；

(3) 而当用户开启文字匹配模式后，将启用文字匹配模块，新加入“文字匹配”、“显示视频描述”两个用例，分别拓展到“搜索视频”用例、“点击视频”用例中。“文字匹配”将在执行用户搜索视频流程中，开启到文本数据库中查找用户输入文本的操作，得到文字匹配结果，和视频检索结果一起通过 Web 模块展示；“显示视频描述”将在执行用户点击视频流程中，开启到文本数据库中查找用户点击的视频的操作，得到用户点击视频所对应的视频描述，和视频一起通过 Web 模块展示。

4.2 总体设计

4.2.1 系统环境及部署

本实验的文本视频检索系统使用 Django 搭建。硬件环境为 1 张 Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz 和 8GB 内存。软件环境为 Microsoft Windows 10 家庭中文版。

编程语言为 Python3.9。编程环境为 Pycharm Community Edition。

4.2.2 系统架构

如图 4.2 的系统框架图所示，文本视频检索系统主要包括以下三个模块：

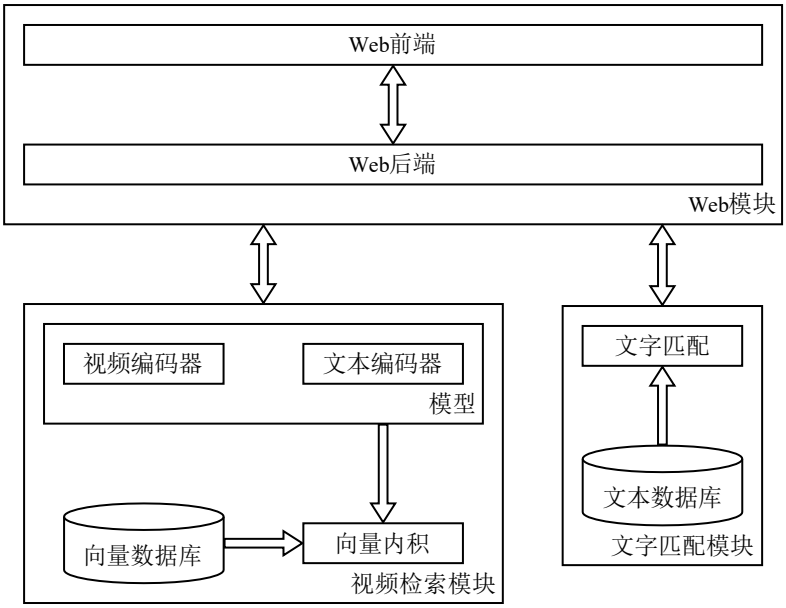

图 4.2 文本视频检索系统框架图

(1) 视频检索模块：主要包括模型和向量数据库两部分，针对满足“检索速度”的功能性需求而设计。模块接收文本，使用模型的文本编码器将文本编码为文本特征；然后与向量数据库中保存的（视频库中所有视频的）视频特征计算相似度（使用矩阵内积作为相似度指标）；并返回相似度最大的 k 个视频，作为视频检索结果。

(2) Web 模块：使用 Python 中的 Django 库搭建，主要包括 Web 前端和 Web 后端两部分。其中 Web 前端接收用户输入传递给 Web 后端，并接收 Web 后端传递的 HTML 界面信息进行展示；Web 后端处理 Web 前端传来的用户输入，然后调用相应视频检索模块和文字匹配模块计算得到结果，并将结果处理为 HTML 界面，返回给 Web 前端。

(3) 文字匹配模块：为满足让用户“可选择展示文字匹配结果作为参考”的功能性需求而设立的模块，主要包括文本数据库。文本数据库使用 SQLite 数据库（传统的关系型数据库）存储了视频库中所有视频的视频描述（即 MSR-VTT 数据集中的文字描述）。当用户开启文字匹配模式时，模块接收文本，并使用文本在文本数据库进行文字匹配；返回视频描述含有该文本的视频，作为文字匹配结果。

4.3 详细设计

4.3.1 视频检索模块

4.3.1.1 设计

视频检索模块包括模型和向量数据库，对外提供一个接口“search”用于发起文本视频检索。向量数据库的引入重复利用模型双塔结构的特点，使得视频库视频数量一万的情况下，一次视频检索的时间开销为 0.1 秒数量级；视频数量一千的情况下，一次视频检索的平均时间开销约为 0.06 秒。

该模块实现以下几点功能：

(1) 提供向量数据库 Tensor Database，分别实现快速、便捷的矩阵存储和读取。在向量数量规模较小时（数量级为十万及以下），使用 pandas 库进行 pickle 文件的存取，即可达到较为理想的存取速度；

(2) 系统启动前若向量数据库为空，则进行向量数据库的初始化构建。使用模型计算视频库中所有视频的视频特征，并将视频特征存储在向量数据库中；

(3) search 接口完成视频检索。首先将接口传入的文本使用 CLIP 中的 tokenizer 进行分子词划分、token 化、转换为 Tensor，得到文本输入；然后将文本输入使用模型编码得到文本特征；然后与向量数据库中所有视频特征计算向量内积，并取最大 k 个的视频编码，作为检索结果返回。

4.3.1.2 流程图

当用户发起一次视频搜索时，用户搜索视频流程如图 4.3 所示：

(1) 首先通过 Web 前端获取用户输入的文本、用户是否开启了文字匹配模式，提交给后端；

(2) 然后 Web 后端解析得到文本，将文本传递给视频检索模块；视频模块首先使用模型的文本编码器将文本编码得到文本特征，然后与向量数据库中存储的

所有视频特征计算向量内积，取向量内积最大的前 k 个所对应视频的视频编号，返回给 Web 后端；

(3) 若文字匹配模式处于开启状态，Web 后端同时将文本传递给文字匹配模块，在文本数据库中使用模糊匹配（忽略大小写，返回文字字段中包含指定查询文本的结果）找到所有视频描述包含该文本的视频，将对应视频编号返回给 Web 后端；

(4) 最后 Web 后端利用视频检索模块（和文字匹配模块）返回的视频编号，找到视频库中对应视频；利用视频相关信息（如视频编号、视频封面、视频链接等）构造出检索结果展示界面的 HTML 代码，返回给 Web 前端；

(5) Web 前端获得代码，更新所显示的 HTML 界面，将检索结果展示给用户。

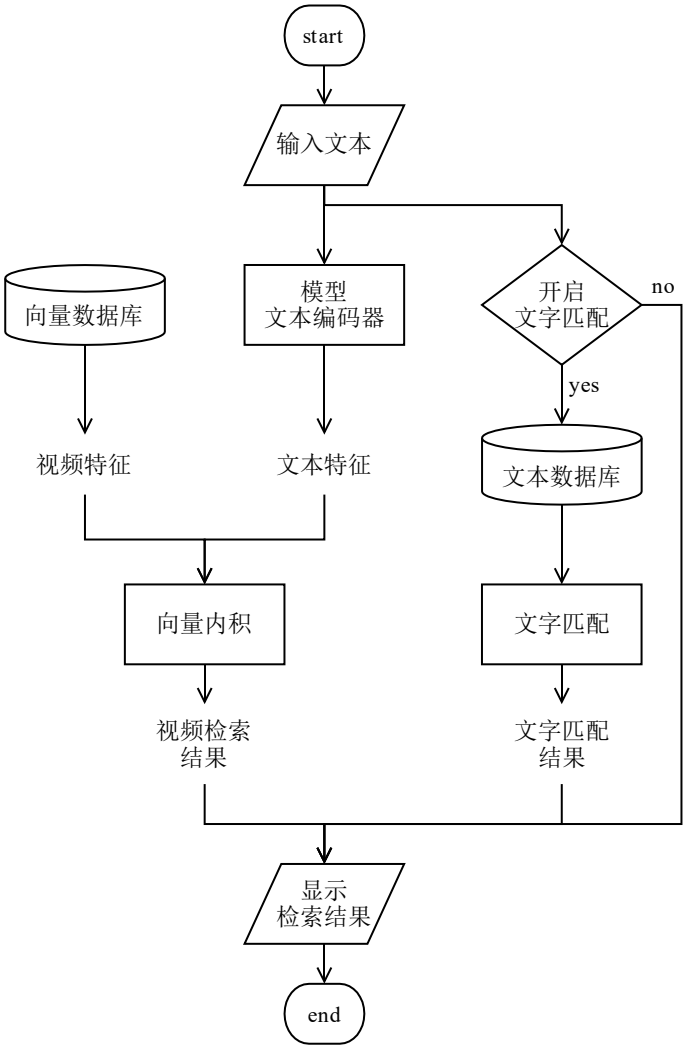

图 4.3 用户搜索视频流程图

4.3.2 Web 模块

4.3.2.1 设计

Django 框架采用 MTV（模型-模板-视图）设计架构。其中模型层（Model Layer）处理与数据库的交互，包括定义数据库关系表、存储和检索数据等；视图层（View Layer）响应 HTTP 请求和处理表单，确定响应应该如何返回，将数据从模型层传递到模板层；模板层（Template Layer）是一个渲染引擎，将数据从视图传递给 HTML 模板，同时还允许在 HTML 文件中插入程序逻辑，以提高可维护性和代码复用度。

模型层对应程序中的 models.py，是文字匹配模块的主要组成部分；视图层对应程序中的 views.py，主要对应 Web 后端；模板层对应程序中的 HTML 模板文件，主要对应 Web 前端。

4.3.2.2 流程图

Web 前端将视频搜索结果显示后，用户便可在界面上点击选择想要进行播放的视频。用户发起一次视频点击时，用户点击视频流程如图 4.4 所示：

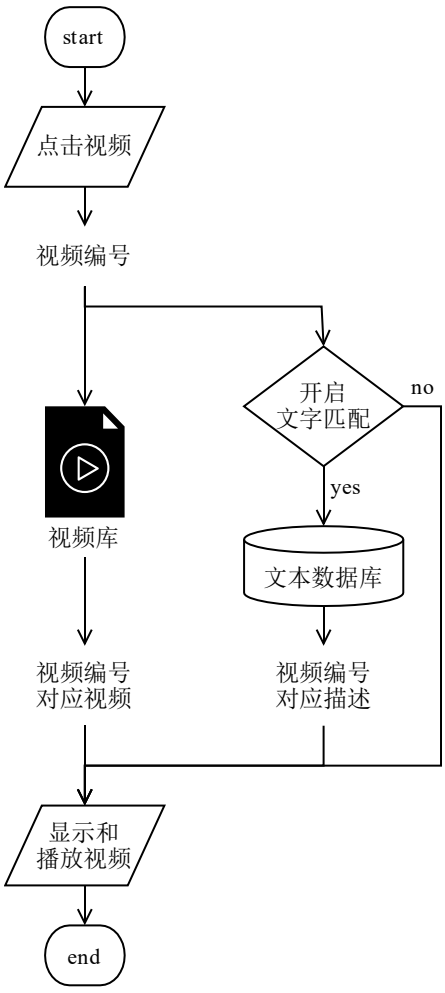

图 4.4 用户点击视频流程图

(1) Web 前端首先将用户点击视频的视频编号、用户是否开启了文字匹配模式提交给 Web 后端；

(2) 然后 Web 后端根据解析得到的视频编号，找到视频库中对应视频，构造相应的视频链接；

(3) 若文字匹配模式处于开启状态，Web 后端同时通过视频编号到文本数据库中找到该视频编号对应的视频描述；

(4) 并利用视频链接（和视频描述）构造出视频播放界面的 HTML 代码，返回给 Web 前端；

(5) Web 前端获得代码，更新所显示的 HTML 界面，将检索结果展示给用户。

4.3.3 文字匹配模块

为实现文字匹配，程序中选择了创建数据库的方案，数据库使用的是 Python 自带的轻量级数据库 SQLite，且其不需要安装额外驱动，提高了程序的兼容性。

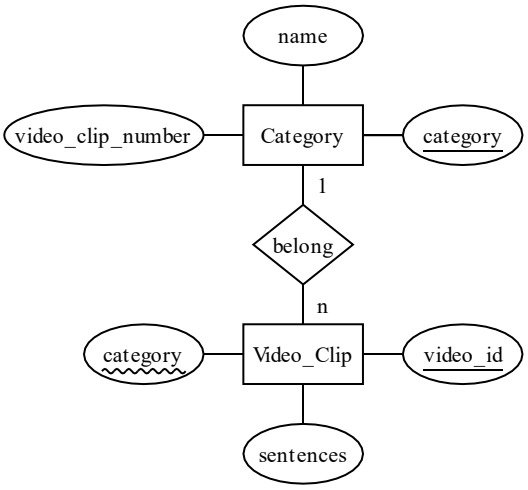

图 4.5 文本数据库 E-R 图

文本数据库的 E-R 图如图 4.5 所示。实体 Category 表示类别（MSR-VTT 数据集中共 20 个视频类别），其属性分别为：

(1) category：类别编号，是不可为空的正整数，且唯一，作为主键；

(2) name：类别名字，例如 music、gaming、sports、movie 等，是不长于 100 的字符串，可为空；

(3) video_clip_number：属于该类别的视频数量，可为空的正整数；

实体 Video_Clip 表示视频，主要存储数据集中的视频描述，用于进行文本匹配。Category 与 Video_Clip 是一对多的关系，SQLite 数据库中用外键表示一对多的关系，

其属性分别为：

(1) video_id：视频编号，是不可为空的正整数，且唯一，作为主键；

(2) category：外键，指向 Category；

(3) sentences：视频描述，是不限长度的字符串，对于 MSR-VTT 数据集，每个视频有多个视频描述，则使用换行连接后存储。

4.4 实现

4.4.1 静态资源管理

Web 后端的一项关键任务是静态资源管理，Django 中，图片、CSS 等文件都属于静态文件，通过在配置文件（setting.py）中配置 STATICFILES_DIRS 和 STATIC_URL 实现加载。

例如在项目根目录下创建文件夹“image”用于存放图片，那么就应该将 image 的路径添加到 STATICFILES_DIRS 中，并可设置命名空间（prefix）为“命名空间 1”，用命名空间来区分不同文件夹下的同名文件。若有文件“image/子文件夹/图片.jpg”，那么在代码中通过"{% static '命名空间 1/子文件夹/图片.jpg' %}"或是"{% get_static_prefix %}命名空间 1/子文件夹/图片.jpg"访问。

而 STATIC_URL 是静态文件在网页中显示的链接的前缀，通常设为“static/”。例如上例中的“图片.jpg”，那么这时其在网页中所显示的完整链接就是“http://网页地址/static/命名空间 1/子文件夹/图片.jpg”。

在本程序中，主要使用到 2 个静态资源文件夹，分别是视频根目录和视频封面根目录。视频根目录下存放的是 MSR-VTT 数据集的原始视频，用于在视频播放页面进行视频播放，命名空间使用 videos；视频封面根目录下存放的是所有视频的封面，用于在搜索结果展示界面进行展示，命名空间使用 images。

因为 MSR-VTT 数据集的视频文件命名规则是“video 视频编号.mp4”，所以可以方便地设置视频对应封面文件名为“视频编号.jpg”，这样，只需要给定视频编号，通过"{% get_static_prefix %}videos/video 视频编号.mp4"和"{% get_static_prefix %}images/视频编号.jpg"便可访问到对应的视频和视频封面。

4.4.2 视图层

用户可发起的用例主要为搜索视频、点击视频、开启文字匹配这三个。当用户进行视频搜索时，Web 后端主要参与到流程图 4.3 的输入文本和显示检索结果这两个

步骤，通过视图层中的 home 函数实现；当用户进行点击视频时，流程图 4.4 的全过程都主要 Web 后端进行，通过视图层中的 video_player 函数实现；当用户开启文字匹配后，将额外启用文字匹配模块，Web 后端将加入相应文本数据库查询和数据传递步骤。

home 函数中，用户在搜索框输入的文本通过 GET 发送回后端，命名为 q；当 q 不为空时，表示用户发起了一次视频查询，调用视频检索模块的 search 接口，将 q 作为文本传入即可得到视频检索结果。将视频检索结果传递给模板层的 home.html 模板文件，便可根据其中的视频编号构造视频封面的静态文件链接，进行检索结果展示。

当用户点击展示结果的视频时，将跳转到视频播放页面，将该视频的视频编号传递给 video_player 函数。video_player 函数中将视频编号传递给模板层的 video_player.html 模板文件，便可构造视频的静态文件链接，进行视频播放。

4.4.3 文本数据库查询

当文字匹配开启时，视图层中函数还需要添加文本数据库中的查询结果。

在视图层的 home 函数中，若 q 不为空，则不区分大小写地查找所有 sentences 包含 q 的 Video_Clip，并将匹配结果也传入到模板层。SQL 查询语句见代码(4.1)，Django 中提供了 Q 函数用于对 SQL 语句进行封装，对应 Python 代码见代码(4.2)。

$$SELECT\ *\ FROM\ Video\_Clip\ WHERE\ Video\_Clip.sentences\ \text{ILIKE}\ '\%q\%' \quad (4.1)$$

$$Video\_Clip.objects.filter\big(Q(sentences\_\_icontains = q)\big) \quad (4.2)$$

在视图层的 video_player 函数中，还需使用视频编号（记为 pk）到文本数据库中找到该视频对应的视频描述，并将该视频描述也传入到模板层。SQL 查询语句见代码(4.3)，对应 Django 代码见代码(4.4)。

$$SELECT\ sentences\ FROM\ Video\_Clip\ WHERE\ video\_id = 'pk' \quad (4.3)$$

$$Video\_Clip.objects.get(video\_id = pk).sentences \quad (4.4)$$

4.5 测试

4.5.1 检索结果展示

检索结果展示页面主要分为导航栏（顶部）、视频检索结果栏（左侧）、文本匹配结果栏（右侧）三个部分，如图 4.6 所示。

导航栏中从左至右依次为：

(1) 主页按钮：点击，回到起始页面；

(2) 搜索框：输入要检索的文本，完成后回车发起检索；

(3) 文字匹配模式开关：点击，在打开文字匹配和关闭文字匹配间切换；

(4) 视频数量显示：显示当前视频库中总视频数量。

视频检索结果栏中列举了所有视频检索结果的视频封面，并在下方标出了网络计算出的向量内积；可点击视频封面进入对应视频的播放页面。文本匹配结果栏中列出了在文本数据库中文字匹配结果数量，并依次列出了各视频的一个视频描述；可点击视频描述进入对应视频的播放页面。

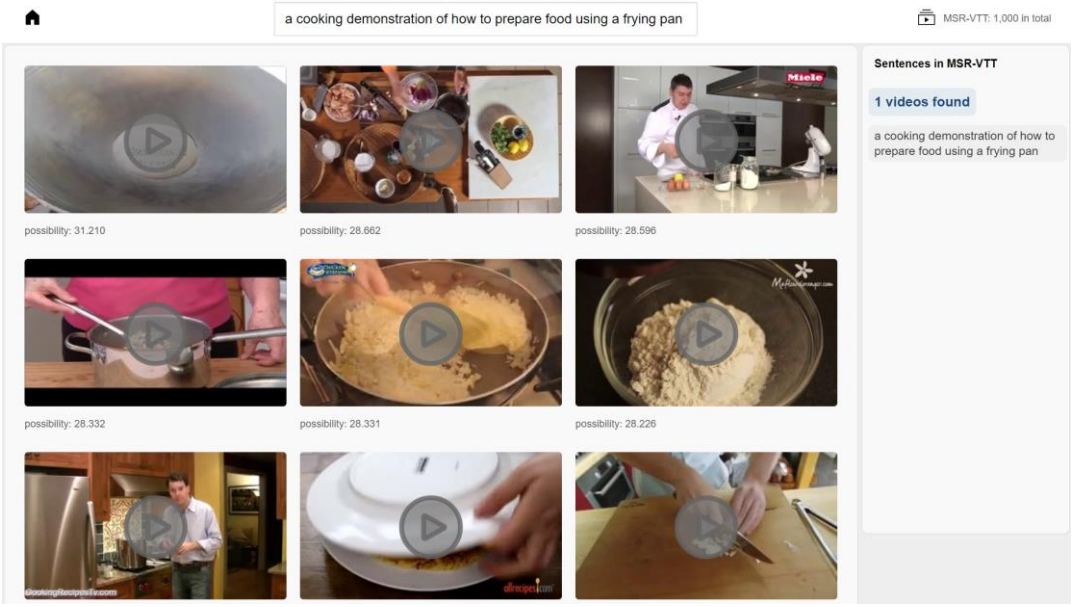

图 4.6 检索结果展示页面（文字匹配开启）

当文字匹配关闭时，文本匹配结果栏将不会显示，如图 4.7 所示。

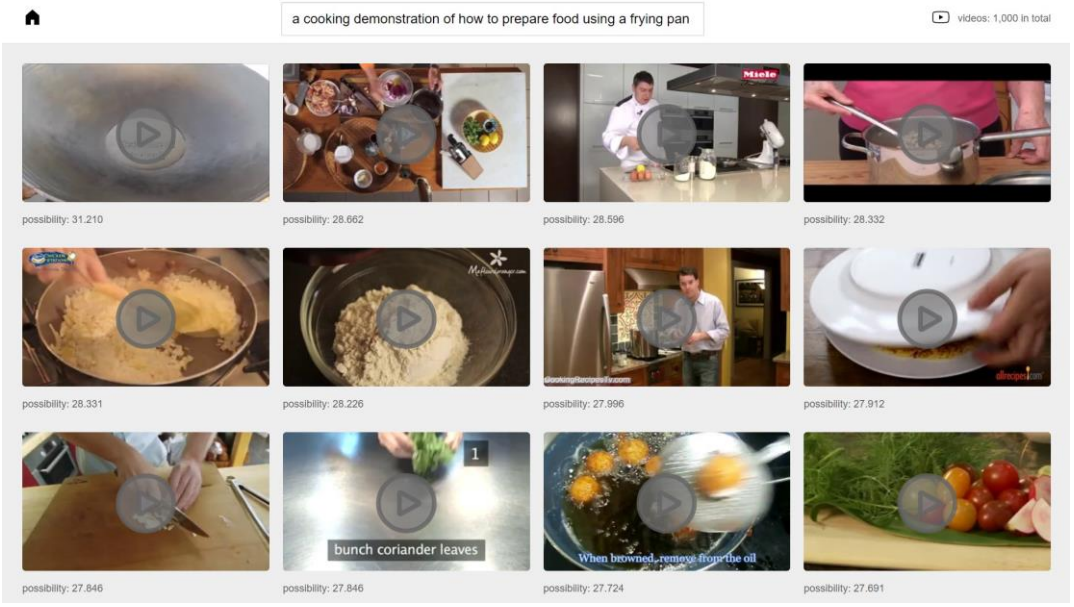

图 4.7 检索结果展示页面（文字匹配关闭）

4.5.2 测试用例

为方便测试人员进行测试，将 Web 后端设置成接收到用户输入为空时，就随机从文本数据库中选择一句视频描述，作为输入文本。这样，只需用户在搜索栏不输入任何内容，直接发起检索，就可以使用 MSR-VTT 测试集中随机一个视频描述作为测试用例。

而且测试人员可十分轻松地反复进行这一步骤，与文字匹配结果进行比较，以观测不同测试用例下的视频检索性能，Web 前端也会将这句文本显示在搜索栏中，以供用户参考。

如图 4.6 所示，就随机从 MSR-VTT 测试集中选出了文本“a cooking demonstration of how to prepare food using a frying pan”用于发起检索；从右侧的文本匹配结果栏中可以看到一共在文本数据库中检索到了一个结果，可以直接点击该结果查看其对应视频，也就是图 4.8 所示视频；再检查左侧的视频检索结果栏，可以看到视频检索结果中，模型计算的该视频与文本的相似度最高，位于检索结果的第一位；而且模型还找到了视频库中很多与“cook”、“food”、“pan”等相关的内容，显然比文字匹配得到的结果更好。

而对于测试用例的通过情况，其等价于模型在 MSR-VTT 测试集上的召回率，已经在算法测试一节中给出。

4.5.3 视频播放

视频播放页面主要分为导航栏（顶部）、视频播放栏（左侧）、视频描述栏（右侧）三个部分，如图 4.8 所示。导航栏与检索结果展示页面一致。视频播放栏中可以进行视频播放、音量调节、全屏等操作；左上角有一个返回按钮返回上一页面。视频描述栏中列出了该视频所属的视频类别、和其所有视频描述。

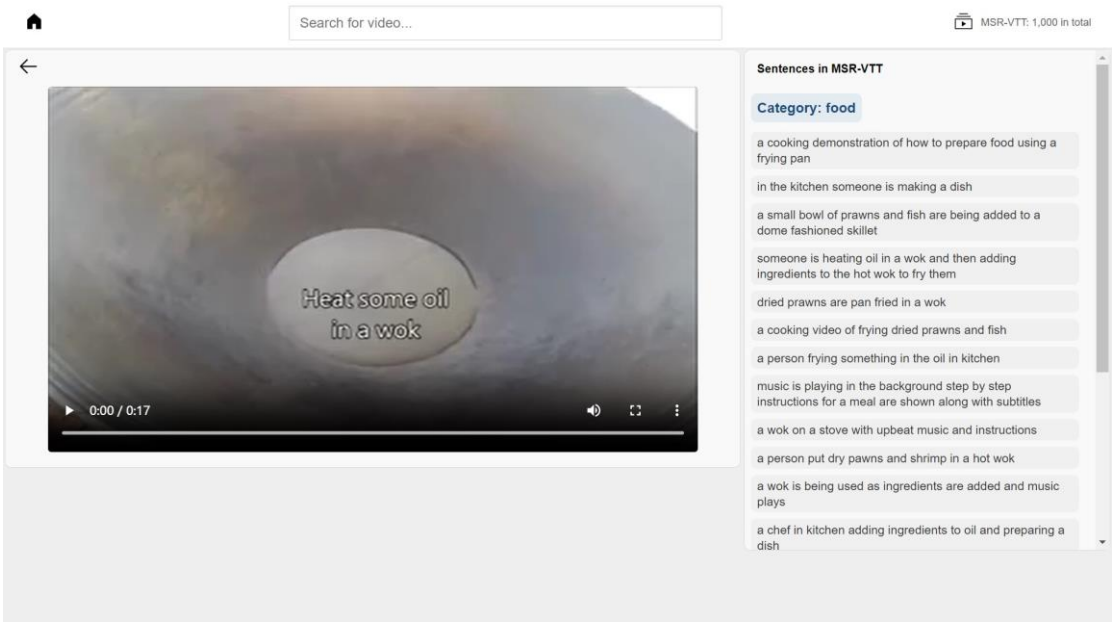

图 4.8 视频播放页面（文字匹配开启）

当文字匹配关闭时，视频描述栏将不会显示，如图 4.9 所示。

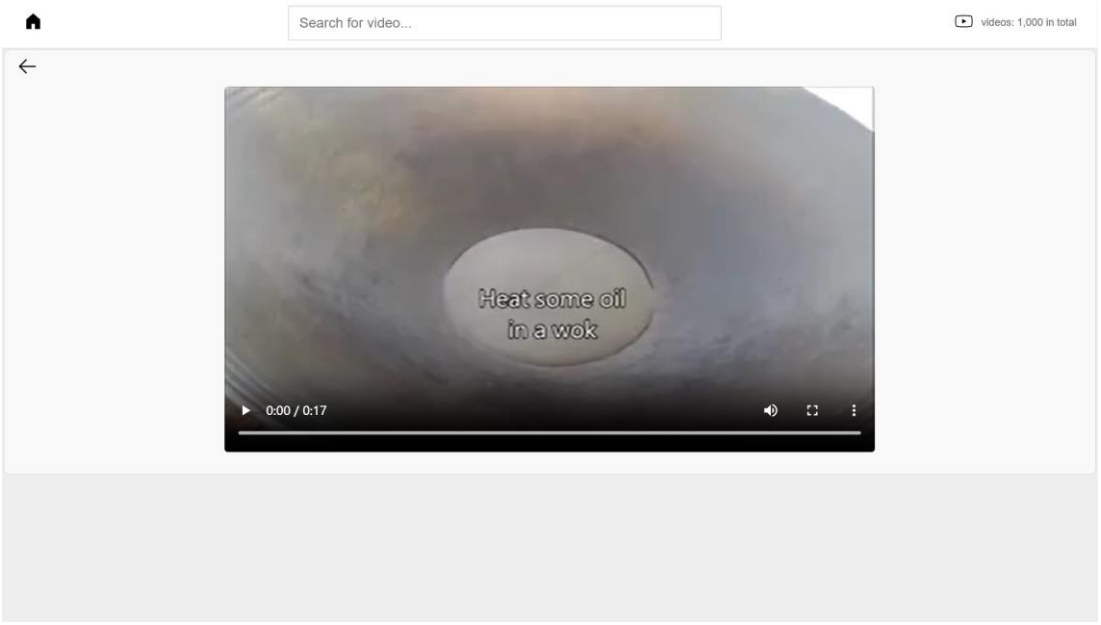

图 4.9 视频播放页面（文字匹配关闭）

4.5.4 页面布局自适应

为了让页面布局和排版在不同屏幕尺寸下做出合适的调整，在 CSS 文件中使用@media 获取当前屏幕尺寸信息，指定不同[min-width, max-width]区间下各标签的属性即可。如图 4.10 和图 4.11 所示，分别是检索结果展示页面和视频播放页面在较小的屏幕尺寸下的自适应。

图 4.10 检索结果展示页面自适应布局

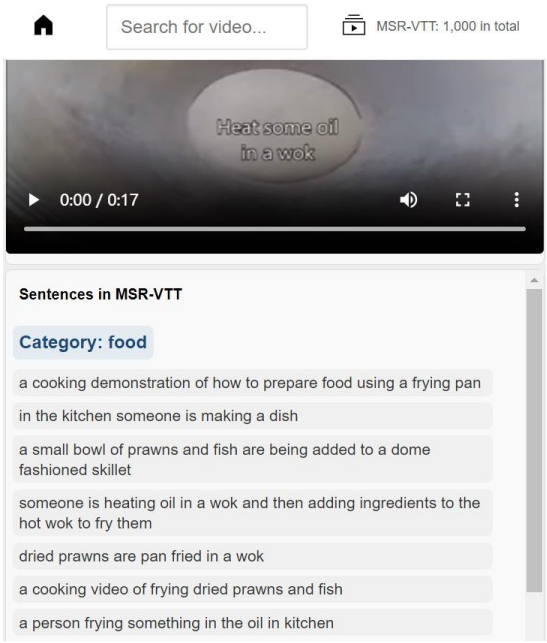

图 4.11 视频播放页面自适应布局

可见，当窗口宽度变小时，界面可以很好地做出调整，将原本处于右侧的文本匹配结果栏、视频描述栏置于下方，且检索结果展示页面一行展示的视频数量也会减少，以让用户能更好地看清界面内容。

5 结论

5.1 结论

CLIP 预训练模型提出后，视频文本检索领域有了很大发展，短短一两年内新模型层出不穷，每次刷新 SOTA 的大多都是基于 CLIP 预训练模型，但模型规模也是不断增大。实验最初时，当意识到在当前实验条件下规模最小的 CLIP4Clip 一次全微调需要耗时近 48 小时时，就将模型实验部分的重点，从提高模型性能，改为了缩短训练时间，其次才是提高模型性能。

(1) 训练速度提升为 34.0 倍：

第一个措施是采用帧保存方案。因为无论采用平均选取方案还是最大帧间差选取方案，同一个视频每次抽取出的关键帧都是相同的，所以事先将关键帧抽取好并以图片形式保存。不仅有效解决了数据加载是训练速度瓶颈的问题，将训练速度提高为 14.6 倍，同时可以减小数据集的硬盘占用。

第二个措施是采用低参数量微调。选择使用 Adapter Tuning，将动作识别领域的 AIM 应用到本实验中。因为低参数量使得模型能更快速地收敛，所以训练速度再次提高为 2.3 倍，配合上帧保存方案，一共将训练速度提升为 34.0 倍，一次低参数量微调只需要 1.4 小时。

(2) 模型性能最高 R@1 从 42.2%提升为 43.4%：

首先测试了两种关键帧选取方案对训练出的模型性能的影响。实验发现平均帧选取方案比最大帧间差选取方案更好，无论是训练集还是测试集，使用平均帧选取方案都能让模型获得更高得分。

然后证明了 AIM 的有效性。通过对比训练 AIM、训练文本编码器和 AIM、全微调 CLIP4Clip 三种不同训练方案下模型的性能，发现将训练整个视频检索模块替换为只训练 Adapter 时，可调参数量减少 49%的情况下，模型性能依然可以达到全微调，证明了 AIM 的有效性。

然后尝试使用 AIM 来提高模型性能。以 CLIP4Clip 作为预训练模型，添加 AIM 进行低参数量微调，将 R@1 从 42.2%提升为 43.4%，增加了 1.2%；R@5 从 70.2%提升为 71.1%，增加了 0.9%。可见 AIM 也提升模型性能，虽然提升幅度有限，但考虑到其训练只需要花费 1.4 小时，也不失为一种实验条件有限情况下的

不错选择。

最后还在 MSVD 数据集上测试了 AIM 的得分。若以 CLIP 为预训练模型，只训练新插入的 Adapter，R@1 从 36.3%提升为 42.5%，R@5 从 63.3%提升为 72.0%；若以在 MSR-VTT 上训练得到的 CLIP4Clip 为预训练模型，只训练新插入的 Adapter，R@1 从 43.4%提升为 44.4%，R@5 从 72.8%提升为 74.5%。

(3) 搭建了向量数据库，有效保障了视频检索系统的速度：

因为模型为双塔结构，所以当用文本进行视频检索时，文本编码器可以独立地计算出文本特征。所以系统部署前，可以将视频库中所有视频的视频特征均计算出来并保存，从而免去视频特征的计算开销。实验条件下，当视频数量为一千时，一次视频检索平均只需 0.06 秒。

设计并测试了不同数据保存方式的数据存取速度，并选取了最快的一种用于向量数据库的内部设计。且对其功能进行了拓展和封装，让该向量数据库能更方便地使用到更多应用场景下。

(4) 使用 Django 搭建了 Web 端应用：

为用户提供了界面美观、使用直观方便的 Web 界面，满足了用户视频检索、播放等多种功能需求。且添加数据库增加了文字匹配模式，开启后可展示文字与视频描述的匹配结果，为用户提供参考，方便用户判断视频检索结果的准确性。

5.2 展望

对于模型性能部分，实验中还未涉及对相似性计算模块的改进。若硬件条件改进，可以尝试使用更复杂的相似性计算模块，例如在 seqTransf 相似性计算模块的 CLIP4Clip 基础上（使用 6 层的 Transformer 来聚合帧图片特征，以获得视频特征）进一步进行改进。

对于关键帧选取方案，虽然已实验初步证明了最大帧间差采样差于平均采用，但具体原因还有待探究，而且还可以尝试寻找其他帧选取方案，或是探究每个视频应该采样多少帧最合适等问题。

对于视频检索系统部分，目前还未添加用户上传视频功能，而且不支持长视频检索。可以尝试将长视频裁剪成多段，以提高每个片段的帧采样密度，获得更细致的片段检索；配合上用户上传视频的功能，具有拓展成带视频检索的云端相册、电影简介视频快速制作工具等应用程序的潜力。

参考文献

[1] Lei J, Li L, Zhou L, et al. Less is more: Clipbert for video-and-language learning via sparse sampling[C], Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021: 7331-7341.

[2] Devlin J, Chang M-W, Lee K, et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding[C]//Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). Minneapolis, Minnesota: Association for Computational Linguistics, 2019: 4171-4186. doi: 10.18653/v1/N19-1423.

[3] Bain M, Nagrani A, Varol G, et al. Frozen in time: A joint video and image encoder for end-to-end retrieval[C], Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021: 1728-1738.

[4] Dosovitskiy A, Beyer L, Kolesnikov A, et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale[C]// 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net, 2021.

[5] Bertasius G, Wang H, Torresani L. Is space-time attention all you need for video understanding?[C], ICML. 2021, 2(3): 4.

[6] Radford A, Kim J W, Hallacy C, et al. Learning transferable visual models from natural language supervision[C], International conference on machine learning. PMLR, 2021: 8748-8763.

[7] Luo H, Ji L, Zhong M, et al. CLIP4Clip: An empirical study of CLIP for end to end video clip retrieval and captioning[J]. Neurocomputing, 2022, 508: 293-304.

[8] Gao Z, Liu J, Chen S, et al. Clip2tv: An empirical study on transformer-based methods for video-text retrieval[J]. 2021.

[9] Fang H, Xiong P, Xu L, et al. Clip2video: Mastering video-text retrieval via image clip[J]. 2021.

[10] Wang J, Ge Y, Cai G, et al. Object-aware video-language pre-training for retrieval[C], Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022: 3313-3322.

[11] Wang Q, Zhang Y, Zheng Y, et al. Disentangled representation learning for text-video retrieval[J]. 2022.

[12] Jiang J, Min S, Kong W, et al. Tencent Text-Video Retrieval: Hierarchical Cross-Modal Interactions with Multi-Level Representations[J]. IEEE Access, 2022.

[13] Houlsby N, Giurgiu A, Jastrzebski S, et al. Parameter-efficient transfer learning for NLP[C], International Conference on Machine Learning. PMLR, 2019: 2790-2799.

[14] Pfeiffer J, Rücklé A, Poth C, et al. AdapterHub: A Framework for Adapting Transformers[C]// Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. Online: Association for Computational Linguistics, 2020: 46-54. doi: 10.18653/v1/2020.emnlp-demos.7.

[15] Guo D, Rush A, Kim Y. Parameter-Efficient Transfer Learning with Diff Pruning[C]// Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). Online: Association for Computational Linguistics, 2021: 4884-4896. doi: 10.18653/v1/2021.acl-long.378.

[16] Li X L, Liang P. Prefix-Tuning: Optimizing Continuous Prompts for Generation[C]// Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). Online: Association for Computational Linguistics, 2021: 4582-4597. doi: 10.18653/v1/2021.acl-long.353.

[17] Lester B, Al-Rfou R, Constant N. The Power of Scale for Parameter-Efficient Prompt Tuning[C]// Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, 2021: 3045-3059. doi: 10.18653/v1/2021.emnlp-main.243.

[18] Hu E J, Shen Y, Wallis P, et al. LoRA: Low-Rank Adaptation of Large Language Models[C]// The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022. OpenReview.net, 2022.

[19] Liu Y, Ott M, Goyal N, et al. Roberta: A robustly optimized bert pretraining approach[J]. 2019.

[20] He P, Liu X, Gao J, et al. Deberta: decoding-Enhanced Bert with Disentangled Attention[C]// 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net, 2021.

[21] Radford A, Wu J, Child R, et al. Language models are unsupervised multitask learners[J]. OpenAI blog, 2019, 1(8): 9.

[22] Brown T, Mann B, Ryder N, et al. Language models are few-shot learners[J]. Advances in neural information processing systems, 2020, 33: 1877-1901.

[23] Yang T, Zhu Y, Xie Y, et al. AIM: Adapting Image Models for Efficient Video Action Recognition[C]// The Eleventh International Conference on Learning Representations. 2023.

[24] Ramesh A, Pavlov M, Goh G, et al. Zero-shot text-to-image generation[C]//International Conference on Machine Learning. PMLR, 2021: 8821-8831.

[25] Karras T, Laine S, Aila T. A style-based generator architecture for generative adversarial networks[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019: 4401-4410.

[26] Patashnik O, Wu Z, Shechtman E, et al. Styleclip: Text-driven manipulation of stylegan imagery[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021: 2085-2094.

[27] Ashish V, Noam S, Niki P, et al. Attention is all you need[C], NeurIPS. 2017: pages 5998–6008.

[28] Xu J, Mei T, Yao T, et al. Msr-vtt: A large video description dataset for bridging video and language[C], Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 5288-5296.

[29] Gabeur V, Sun C, Alahari K, et al. Multi-modal transformer for video retrieval[C], Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IV 16. Springer International Publishing, 2020: 214-229.

[30] Yu Y, Kim J, Kim G. A joint sequence fusion model for video question answering and retrieval[C], Proceedings of the European Conference on Computer Vision (ECCV). 2018: 471-487.

[31] Chen D, Dolan W B. Collecting highly parallel data for paraphrase evaluation[C], Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies. 2011: 190-200.

[32] Kingma D P, Ba J. Adam: A Method for Stochastic Optimization[C]// 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. Eds. Bengio Y and LeCun Y. 2015.

致谢

感谢栗伟老师的指导与帮助。

感谢董闯学长的指导与帮助。

感谢本科期间所有课程老师的教育，感谢学校 IPV6 网络不收费。

感谢同学、朋友、家人，感谢我自己。