CVPR2021最佳学生论文提名：Less is More

原创 CV开发者都爱看的 极市平台 2021-07-11 22:00:00 手机阅读

收录于话题
#CVPR2021 5 #视觉顶会 50

作者丨小马
编辑丨极市平台

极市导读

本文介绍的是今年CVPR的最佳学生论文提名的工作：ClipBert。这篇论文解决了以前工作中对于视频-语言任务训练消耗大、性能不高、多模态特征提取时没有交互等问题。 >>加入极市CV技术交流群，走在计算机视觉的最前沿

【写在前面】

本文介绍的是今年CVPR的最佳学生论文提名的工作：ClipBert。这篇论文解决了以前工作中对于视频-语言任务训练消耗大、性能不高、多模态特征提取时没有交互等问题。另外，这是一篇用Image-Text 预训练的模型去解决Video-Text的任务。以前的Video-Text任务大多是对视频进行Dense采样，而本文通过预训练的Image-Text模型，对视频进行稀疏采样，只需要很少的帧数，就能超过密集采样的效果，进而提出了本文标题中的 **“Less is More”**。

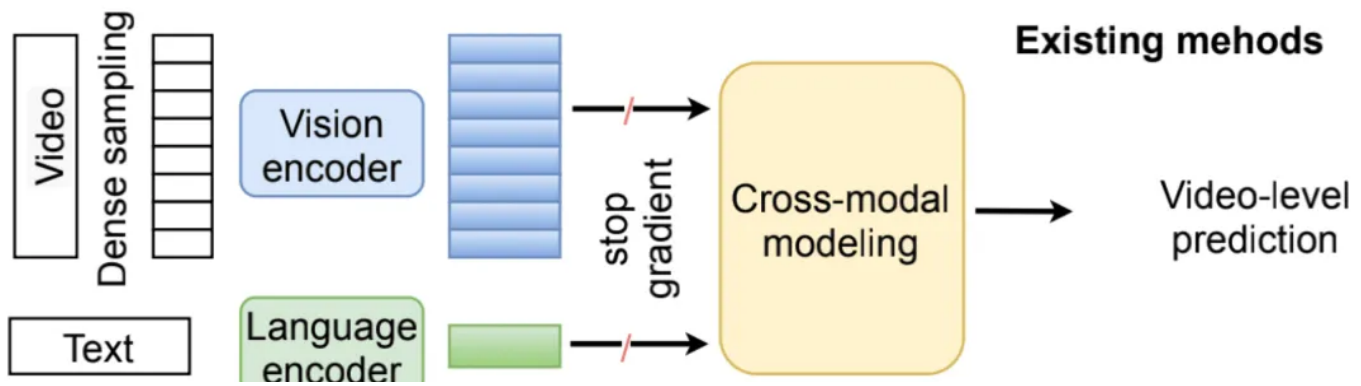
1. 论文和代码地址

Less is More: CLIPBERT for Video-and-Language Learning via Sparse Sampling

论文地址：https://arxiv.org/abs/2102.06183

代码地址：https://github.com/jayleicn/ClipBERT

2. Motivation



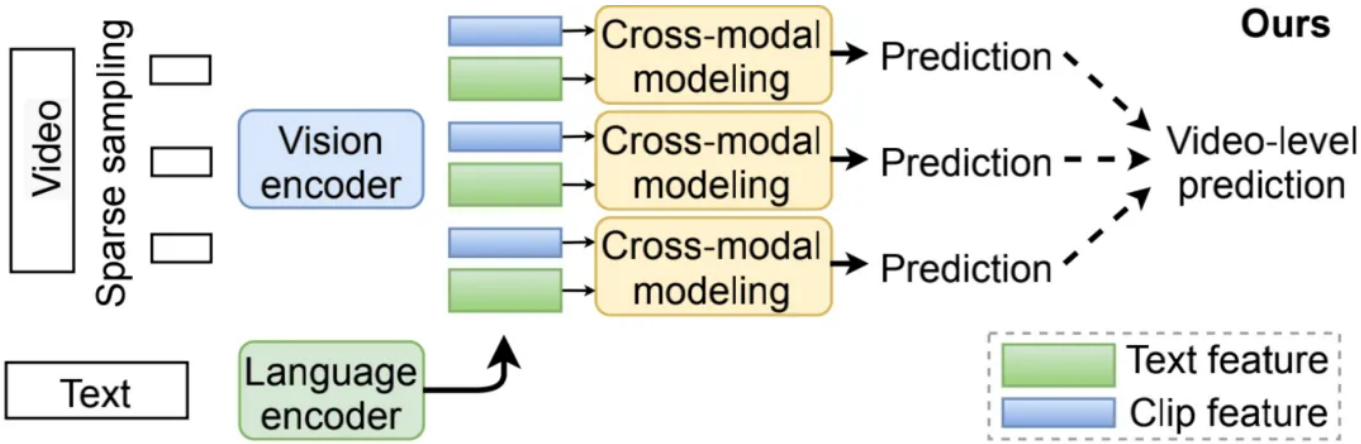
在这篇文章之前，大多数解决Video-Language任务的工作的框架图都是如上图所示的这样，对视频和语言分别提取特征，然后通过多模态信息的融合结构，将两个模态的信息embedding到一个共同的空间。

这样的结构具有两个方面的问题：

1） Disconnection in tasks/domains：缺少了任务之间的联系，比如在动作识别任务中提取的特征，并不一定使用于下游任务（比如：Video Captioning），由于存在任务之间的gap，这样的特征提取方式会有导致sub-optimal的问题。

2） Disconnection in multimodal features：缺少了特征之间的联系，两个特征提取分支在提取特征过程中没有进行交互，导致视觉和语言信息并没有得到充分联系。

除此之外，在提取视频特征的时候，由于相比于图片特征，视频会多一个时间维度，因此提取视频特征是非常耗时、并且计算量是非常大的。



基于以上的问题，本文提出了CLIPBERT，一个端到端的视频-语言学习框架（结构如上图）。相比于以前的框架，本文主要有下面几个方面的不同：

1）以前的方法是对原始视频以dense的方式提取特征，非常耗时、耗计算量。但是众所周知，视频中大多数的帧其实都是非常相似的，对这些相似的帧进行特征提取确实比较浪费（而且就算提取了信息，模型也不一定能够学习到，如果模型的学习能力不够，过多的冗余信息反而会起到反作用）。因此，本文对视频采用稀疏采样，只采用很少的几张图片。

2）如果对这些采样的clip拼接后同时计算，这就相当于又多了一个维度，就会增加计算的负担。因此本文是对每一个clip分别计算后，然后再将计算结果融合，来减少计算量和显存使用，从而来实现端到端的视频-语言任务。

另外，本文还有一个创新点就是用图片-语言数据集进行预训练，然后在视频-语言数据上微调，因此本文将图片-语言数据集上学习到的信息转换到了视频-语言这个下游任务中，并且效果非常好。

3. 方法

3.1. Previous work

以往的方法对于视频-文本任务，往往都是直接对密集的视频V和文本S提取特征，每个视频V可以被分成N个clip，因此，以前视频-文本任务的模型可以被建模成下面的公式：

$$p = \mathcal{H}([\mathcal{F}_v^{SG}(c_1), \mathcal{F}_v^{SG}(c_2), ..., \mathcal{F}_v^{SG}(c_N)], \mathcal{F}_l^{SG}(S)),$$

模型的损失函数为基于预测值p和Ground Truth q的特定任务损失函数：

$$l^{task} = \mathcal{L}^{task}(p, q).$$

3.2. 随机稀疏采样

为了减少计算量，作者对视频进行了稀疏采样，只采样了很少的视频帧数，来表示视频的信息，因此本文的模型可以用下面的公式表示（第i个clip的信息可以被表示成c_{τ_i}）：

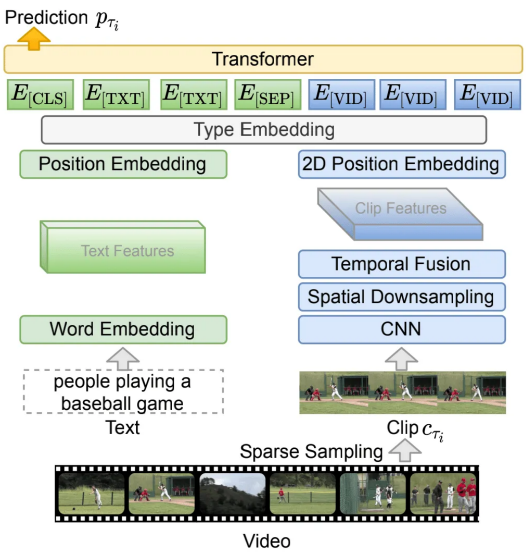
$$p_{\tau_i} = \mathcal{H}(\mathcal{F}_v(c_{\tau_i}), \mathcal{F}_l(S)),$$

作者对每一个clip的特征分别进行预测，然后在将每个clip的预测信息进行融合，本文模型的损失函数如下（G是聚合不同clip信息的函数）：

$$l^{task} = \mathcal{L}^{task}(\mathcal{G}(p_{\tau_1}, p_{\tau_2}, ..., p_{\tau_{N_{train}}}), q),$$

由于在训练中，同一视频每次采样的图片并不一样，所以本文使用的稀疏采样方法也可以被看作是一种数据增强。因为在以往的方法中，是用整个视频进行训练的，全部的视频信息都暴露在模型中；而随机稀疏采样每次都暴露几个clip的信息，并且每次暴露的clip还是不同的，因此就起到了数据增强的效果（就像CV中的Random Crop，不采用数据增强的话，就将整张图片暴露给模型；如果使用Random Crop，每次都会对图片进行随机裁剪，只暴露图片的一部分信息，并且由于裁剪是随机的，所以每次暴露的信息也是不一样的）。

3.3. 模型结构



本文的模型结果如上图所示：

对于视觉编码器，本文采用的是2D CNN结构ResNet-50，因为相比于3D的CNN结构，2D的结构使用的显存更少、速度更快。

图中的Spatial Downsampling模块，作者采用的是2x2的max-pooling。

图中的Temporl Fusion模块，本文采用的mean-pooling，来聚合每个clip中T个帧的信息，得到clip级别的特征表示。

然后采用row-wise和column-wise的position embedding来实现2D的Position Embedding。

对于语言编码器，作者采用了一个work embedding层进行本文信息的映射，然后采用position embedding进行特征位置的编码。

接着，作者又加入了一个type embedding层，对输入特征的类型（视觉或语言）进行编码。

最后，只需要让这些视觉和语言特征进入一个12层的Transformer中进行训练即可。

4. 实验

4.1. Image Size

为了确定最合适的输入图片的大小，作者首先在Image Size上做了实验：

L	MSRVTT Retrieval				MSRVTT-QA Acc.
	R1	R5	R10	MdR	
224	6.8	24.4	35.8	20.0	**35.78**
448	**10.2**	**28.6**	**40.5**	**17.0**	**35.73**
768	**11.0**	27.8	**40.9**	**16.0**	**35.73**
1000	10.0	**28.4**	39.4	18.0	35.19

Table 1: Impact of **input image size** L.

从图中可以看出，图片大小从224→448的过程中，性能提高显著；但将448的图片继续放大，性能提升就不太显著了，甚至部分指标已经开始下降了。

4.2. 帧信息的聚合

实验中，每个clip采样了T帧，为了探究如何对这些帧的信息进行聚合，获得clip级别的特征表示，作者尝试了以下方法：

$\mathcal{M}$	T	MSRVTT Retrieval				MSRVTT-QA Acc.
		R1	R5	R10	MdR	
-	1	10.2	28.6	40.5	17.0	35.73
Mean Pooling	2	**11.3**	31.7	44.9	14.0	**36.02**
	4	10.8	30.0	43.6	14.0	35.83
	8	10.6	**32.5**	**45.0**	**13.0**	35.69
	16	**11.6**	**33.9**	**45.8**	**13.0**	**36.05**
Conv3D	2	8.7	27.3	40.2	17.0	34.85
	16	10.1	28.9	41.7	16.0	35.03
Conv(2+1)D	2	7.3	24.1	35.6	22.0	34.13
	16	9.9	27.3	39.6	17.0	33.92

Table 2: Impact of **#frames** (T) **and temporal fusion function** ($\mathcal{M}$)**.** We use a 1-second clip for all experiments.

可以看出，Conv3D和Conv(2+1)D的方法明显没有Mean Pooling好。

4.3. 测试时的clip数量

为了探究测试时采用多少clip的信息合适，作者进行了下面的实验：

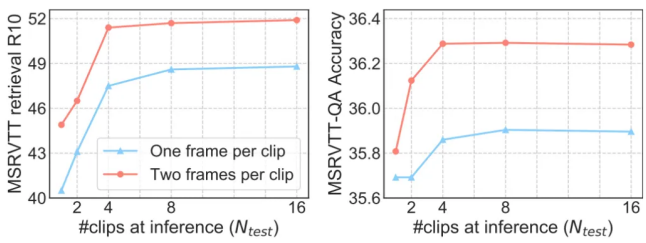


Figure 4: Impact of **#inference clips** (N_{test})**.**

可以看出，clip≤4，随着clip的增加，性能显著提升；clip>4，随着clip的增加，性能提升不显著。此外，每个clip采样两帧明显比采样一帧的效果好。

4.4. 训练时的clip数量

为了探究训练时采用多少clip的信息合适，作者进行了下面的实验：

$\mathcal{G}$	N_{train}	MSRVTT Retrieval				MSRVTT-QA Acc.
		R1	R5	R10	MdR	
-	1	12.7	34.5	48.8	11.0	36.24
Mean Pooling	2	13.3	37.1	50.6	10.0	35.94
	4	14.0	38.6	51.6	10.0	35.40
	8	13.4	36.4	49.7	11.0	35.76
	16	15.2	**39.4**	**53.1**	**9.0**	35.33
Max Pooling	2	8.5	28.7	42.2	14.0	**36.41**
	16	12.5	33.1	46.8	12.0	36.25
LogSumExp	2	**15.5**	38.4	52.6	**9.0**	**36.59**
	16	**17.4**	**41.5**	**55.5**	**8.0**	36.16

Table 3: Impact of **#training clips (N_{train}) and score aggregation function ($\mathcal{G}$).** All models use N_{test}=16 clips for inference.

总体上来说，训练时clip数量从1→2时，效益最显著。比如在LogSumExp函数下，clip从1增加到2，R1高了2.8%;clip从2增加到16，R1高了1.9%。

4.5. 稀疏随机采样 vs.密集均匀采样.

Sampling Method	N_{train}	MSRVTT Retrieval				MSRVTT-QA Acc.
		R1	R5	R10	MdR	
Dense Uniform	16	15.5	39.6	55.0	9.0	35.88
Sparse Random	1	12.7	34.5	48.8	11.0	36.24
	2	15.5	38.4	52.6	9.0	36.59
	4	**15.7**	**41.9**	**55.3**	**8.0**	**36.67**

Table 4: Sparse random sampling *vs.* dense uniform sampling.
All models use N_{test}=16 clips for inference.

可以看出采用4帧的随机采样，就比16帧的均匀采样要好。（因为随机采样有了随机性，就有数据增强的效果）

4.6. 相比于SOTA

Method	R1	R5	R10	MdR
HERO [37] ASR, PT	20.5	47.6	60.9	-
JSFusion [77]	10.2	31.2	43.2	13.0
HT [46] PT	14.9	40.2	52.8	9.0
ActBERT [83] PT	16.3	42.8	56.9	10.0
HERO [37] PT	16.8	43.4	**57.7**	-
CLIPBERT 4×1	**19.8**	**45.1**	57.5	**7.0**
CLIPBERT 8×2	**22.0**	**46.8**	**59.9**	**6.0**

(a) MSRVTT 1K test set.

Method	R1	R5	R10	MdR
CE [41]	16.1	41.1	-	8.3
S2VT [65]	11.9	33.6	-	13.0
FSE [80]	13.9	36.0	-	11.0
CLIPBERT 4×1	**19.9**	**44.5**	**56.7**	**7.0**
CLIPBERT 8×2	**20.4**	**48.0**	**60.8**	**6.0**

(b) DiDeMo test set.

Method	R1	R5	R10	MdR
CE [41]	18.2	47.7	-	6.0
MMT [15]	22.7	54.2	93.2	5.0
MMT [15] PT	28.7	61.4	94.5	3.3
Dense [28]	14.0	32.0	-	34.0
FSE [80]	18.2	44.8	-	7.0
HSE [80]	20.5	**49.3**	-	-
CLIPBERT 4×2*	**20.9**	48.6	**62.8**	**6.0**
CLIPBERT 4×2* (N_{test}=20)	**21.3**	**49.0**	**63.5**	**6.0**

(c) ActivityNet Captions val1 set.

Table 7: Comparison with state-of-the-art methods on text-to-video retrieval. CLIPBERT models with different training input sampling methods are denoted by $N_{train}\times T$. We use N_{test}=16 if not otherwise stated. We gray out models that used features other than appearance and motion for a fair comparison, *e.g.*, CE used appearance, scene, motion, face, audio, OCR, ASR features from 11 different models. *PT* indicates the model is pre-trained on HowTo100M. * denotes models use 2-second clips instead of the default 1-second clips.

Method	Action	Transition	FrameQA
ST-VQA [23]	60.8	67.1	49.3
Co-Memory [17]	68.2	74.3	51.5
PSAC [38]	70.4	76.9	55.7
Heterogeneous Memory [12]	73.9	77.8	53.8
HCRN [31]	75.0	81.4	55.9
QueST [25]	75.9	81.0	**59.7**
CLIPBERT 1×1 (N_{test}=1)	**82.9**	**87.5**	59.4
CLIPBERT 1×1	**82.8**	**87.8**	**60.3**

(a) TGIF-QA test set.

Method	Accuracy
ST-VQA [23] (by [12])	30.9
Co-Memory [17] (by [12])	32.0
AMU [74]	32.5
Heterogeneous Memory [12]	33.0
HCRN [31]	35.6
CLIPBERT 4×1	**37.0**
CLIPBERT 8×2	**37.4**

(b) MRSVTT-QA test set.

Method	Accuracy
SNUVL [78] (by [77])	65.4
ST-VQA [23] (by [77])	66.1
CT-SAN [79] (by [77])	66.4
MLB [27] (by [77])	76.1
JSFusion [77]	83.4
ActBERT [83] PT	85.7
CLIPBERT 4×1	**87.9**
CLIPBERT 8×2	**88.2**

(c) MRSVTT multiple-choice test.

Table 8: Comparison with state-of-the-art methods on video question answering.

在多个视频-语言任务上，本文提出方法的性能能够大大超过以往的SOTA模型，证明了本文方法的有效性。

5. 总结

本文提出了一个端到端的视频-语言训练框架，只采样了视频中的部分信息，就能超过以前密集采样的方法，证明了“less is more”思想的有效性。另外，本文的方法在多个数据集、多个任务上都远远超过以前的SOTA方法。

本文亮点总结

1.以前的方法是对原始视频以dense的方式提取特征，非常耗时、耗计算量。但是众所周知，视频中大多数的帧其实都是非常相似的，对这些相似的帧进行特征提取确实比较浪费（而且就算提取了信息，模型也不一定能够学习到，如果模型的学习能力不够，过多的冗余信息反而会起到反作用）。因此，本文对视频采用稀疏采样，只采用很少的几张图片。

2.如果对这些采样的clip拼接后同时计算，这就相当于又多了一个维度，就会增加计算的负担。因此本文是对每一个clip分别计算后，然后再将计算结果融合，来减少计算量和显存使用，从而来实现端到端的视频-语言任务。