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

Jie Lei*[1], Linjie Li*[2], Luowei Zhou[2], Zhe Gan[2], Tamara L. Berg[1], Mohit Bansal[1], Jingjing Liu[2]
[1]UNC Chapel Hill [2]Microsoft Dynamics 365 AI Research
{jielei, tlberg, mbansal}@cs.unc.edu
{lindesy.li, luowei.zhou, zhe.gan, jingjl}@microsoft.com

Abstract

*The canonical approach to video-and-language learning (*e.g.*, video question answering) dictates a neural model to learn from offline-extracted dense video features from vision models and text features from language models. These feature extractors are trained independently and usually on tasks different from the target domains, rendering these fixed features sub-optimal for downstream tasks. Moreover, due to the high computational overload of dense video features, it is often difficult (or infeasible) to plug feature extractors directly into existing approaches for easy finetuning. To provide a remedy to this dilemma, we propose a generic framework* CLIPBERT *that enables affordable end-to-end learning for video-and-language tasks, by employing* ***sparse sampling****, where only a single or a few sparsely sampled short clips from a video are used at each training step. Experiments on text-to-video retrieval and video question answering on six datasets demonstrate that* CLIPBERT *outperforms (or is on par with) existing methods that exploit full-length videos, suggesting that end-to-end learning with just a few sparsely sampled clips is often more accurate than using densely extracted offline features from full-length videos, proving the proverbial* ***less-is-more*** *principle. Videos in the datasets are from considerably different domains and lengths, ranging from 3-second generic-domain GIF videos to 180-second YouTube human activity videos, showing the generalization ability of our approach. Comprehensive ablation studies and thorough analyses are provided to dissect what factors lead to this success. Our code is publicly available.*[1]

1. Introduction

Humans communicate with each other in this interactive and dynamic visual world via languages, signs, and gestures. The ability to jointly understand both visual and textual clues is an essential ability for intelligent agents to interpret multimodal signals in the physical world. A wide range of tasks based on real-life videos have been designed to test such ability, including text-to-video retrieval [72, 26, 51], video captioning [51, 72, 79], video question answering [71, 21, 31, 32], and video moment retrieval [1, 17, 33]. The *de facto* paradigm to tackle these cross-modal tasks is to first extract dense video features from pre-trained vision models [19, 3] and text features from pre-trained language models [47, 10], then apply multimodal fusion to wrangle together these fixed representations in a shared embedding space (Figure 1 (*top*)).

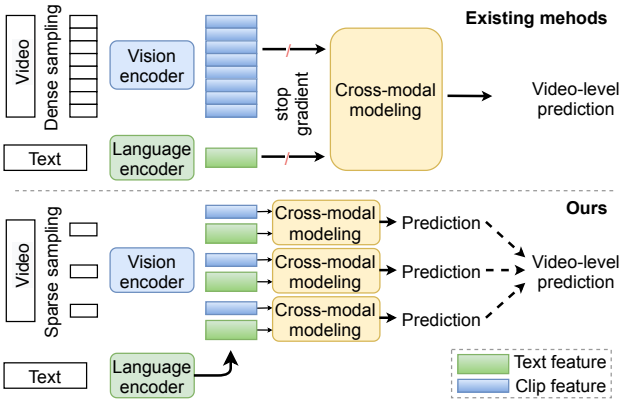

Figure 1: Comparison between popular video-and-language learning paradigm (*top*) and CLIPBERT (*bottom*). In contrast to most existing methods that utilize offline (*stop gradient*) extracted *dense video features* and *text features*, CLIPBERT uses *sparsely sampled clips* and *raw text tokens* for *end-to-end* modeling.

Existing approaches [21, 31, 80, 29] following this paradigm have achieved strong success, yet suffer from two main drawbacks: (i) *Disconnection in tasks/domains*: offline feature extractors are often trained on tasks and domains different from the target task, *e.g.*, features learned for action recognition on human activity videos [24] are incongruently applied to downstream video question answering on generic-domain GIF videos [21]. (ii) *Disconnection*

* Equal contribution.
[1] https://github.com/jayleicn/ClipBERT

in multimodal features: features from different modalities are learned independent of each other, *e.g.*, action recognition models [59, 63, 3] are typically trained on pure videos without textual input, yet are applied to video+language tasks. End-to-end task-specific finetuning offers a way to mitigate these inherent disconnections. However, extracting features from the full sequence of video frames, as in most existing work, casts excessive demand on memory and computation, rendering it difficult or even infeasible to directly plug feature extractors into a video+language learning framework for efficient end-to-end finetuning.

Motivated by this, we propose CLIPBERT, a generic and efficient framework for end-to-end video-and-language learning (Figure 1 (*bottom*)). Two aspects distinguish CLIPBERT from previous work. First, in contrast to densely extracting video features (adopted by most existing methods), CLIPBERT sparsely samples only one single or a few short clips from the full-length videos at each training step. The hypothesis is that sparse clips already capture key visual and semantic information in the video, as consecutive clips usually contain similar semantics from a continuous scene. Thus, a handful of clips are sufficient for training, instead of using the full video. Then, predictions from multiple densely-sampled clips are aggregated to obtain the final video-level prediction during inference, which is less computational demanding. This *sparse-training*-then-*dense-inference* strategy greatly reduces memory needs and computations, allowing economical end-to-end learning from raw video frame pixels and language tokens.

The second differentiating aspect concerns the initialization of model weights (*i.e.*, transfer through pre-training). In recent literature, image-text pre-training (*e.g.*, using COCO Captions [4] or Visual Genome Captions [27]) has been applied to image-text tasks [58, 41, 5, 55, 20, 34, 78], and video-text pre-training (*e.g.*, using HowTo100M [43]) to video-related tasks [56, 80, 14, 35]. There has been no study to cross-examine the effect of image-text pre-training on video-text tasks. Intuitively, visual features learned through pre-training from large-scale image datasets should also help video understanding tasks that rely on visual clues in static video frames. To investigate this, we use 2D (*e.g.*, ResNet-50 [19]) instead of 3D architectures [59, 3, 70] as our visual backbone for video encoding, allowing us to harness the power of image-text pre-training for video-text understanding along with the advantages of low memory cost and runtime efficiency. Empirically, we observe that the knowledge learned in image-text pre-training indeed helps video-text tasks; this simple strategy helps us achieve better or comparable performance to previous state of the art on text-to-video retrieval and video question answering tasks.

Our contributions are three-fold: (i) We propose CLIPBERT, a new end-to-end learning framework for video+language tasks. Experiments show that CLIPBERT achieves superior (or on par) performance than existing methods across diverse video-text tasks, where the average video length ranges from a few seconds to three minutes. (ii) Our work suggests *"less is more"*: the proposed end-to-end training strategy with a single or a few (*less*) sparsely sampled clips is often *more* accurate than traditional approaches that employ densely extracted video features. (iii) We demonstrate that image-text pre-training benefits video-text tasks. We also provide comprehensive ablation studies to reveal the key factors that lead to the success of CLIPBERT, in hope of inspiring more future work.

2. Related Work

Video and Language Understanding. Popular video-and-language tasks include text-to-video retrieval [72, 26, 51], video captioning [72, 79, 26, 51, 37], video question answering [71, 21, 31], and moment retrieval [1, 17, 33]. Standard approaches [21, 71, 16, 77, 31, 11, 29, 30] leverage offline extracted video and text features from action recognition models [24, 63, 3, 70], image recognition models [9, 19], and language models [44, 47, 10, 39]. Aligned with the success of transformer-based [61] language pre-training [10, 39, 73, 49, 28, 7] and image-text pre-training [58, 41, 5, 34, 20, 78, 15, 6], video-text pre-training [56, 80, 14, 35, 42, 43] has shown promising results on video-and-language tasks. Beyond using fixed features and same-domain data (*i.e.*, video-text pre-training only for video-text tasks), our work focuses on end-to-end training and applying image-text pre-training for video-text tasks.

Action Recognition. Modern video action recognition architectures are typically designed with deep 2D [53, 57, 19] or 3D [59, 3, 70] convolutional networks. These backbones are often computation and memory heavy, making it extremely difficult to directly process videos of considerable length. To ease this difficulty, instead of training on full-length videos, models are often trained with randomly sampled short clips from the videos [52, 59, 48, 70, 64, 13, 12, 63]. At inference time, predictions from multiple uniformly sampled clips are aggregated together as the final video-level prediction. In relation to these works, we adopt a similar strategy to perform sparse training and dense inference to reduce overhead on video processing, but focus on video-and-language tasks with cross-modal modeling of video and language, in contrast to pure video modeling.

3. CLIPBERT with Sparse Sampling

We propose CLIPBERT, a general framework that enables end-to-end learning on video and language data, by learning joint representations directly from video frame pixels and raw text tokens, instead of from offline-extracted single-modality features. Figure 1 (*bottom*) gives an overview of CLIPBERT framework. It adopts a sparse sam-

pling strategy using only a single or a few sampled clips at each training step, instead of full-length videos. Each sampled clip is independently encoded with a vision backbone model, the visual features from which are then fed to a cross-modal module that extracts relations between the clip and its associated text representations. Independent predictions from all the sampled clips are fused together (*e.g.*, *through mean-pooling*) to derive a consensus at the video level. A task-specific loss is calculated based on this consensus to learn model parameters. During inference, CLIPBERT densely samples a sequence of clips from the video and aggregates their predictions as the final prediction.

Most existing work [21, 31, 80, 29] models offline-extracted dense video features and text features. We denote a video-text pair as V (video) and S (text). V is further expressed as a list of N clips of equal duration $[c_1, c_2, ..., c_N]$. This standard paradigm can be formulated as:

$$p = \mathcal{H}([\mathcal{F}_v^{SG}(c_1), \mathcal{F}_v^{SG}(c_2), ..., \mathcal{F}_v^{SG}(c_N)], \mathcal{F}_l^{SG}(S)), \quad (1)$$

where $\mathcal{F}_v^{SG}$ and $\mathcal{F}_l^{SG}$ are vision and language encoder, respectively. The superscript SG denotes *Stop Gradient*, meaning that gradients cannot be back-propagated through the two encoders. $\mathcal{H}$ is a cross-modal encoder and predictor, which models the relations between the encoded video/language inputs and makes predictions. p is the video-level prediction. A task-specific loss function $\mathcal{L}^{task}$ is then applied to calculate the loss value l^{task} based on this prediction and its corresponding ground-truth q:

$$l^{task} = \mathcal{L}^{task}(p, q). \quad (2)$$

Sparse Sampling for Training. Instead of using all the N clips, CLIPBERT sparsely (and randomly) samples N_{train} clips $\{c_{\tau_i}\}_{i=1}^{N_{train}}$ from V for training. N_{train} is typically much smaller than N. We model a sampled clip c_{τ_i} together with text input S to produce a prediction p_{τ_i}:

$$p_{\tau_i} = \mathcal{H}(\mathcal{F}_v(c_{\tau_i}), \mathcal{F}_l(S)), \quad (3)$$

where $\mathcal{F}_v$ and $\mathcal{F}_l$ are vision/language encoders. Different from Equation 1 that uses offline vision/language encoders, CLIPBERT is end-to-end trainable, allowing task-specific loss to further finetune the encoders, learning better representations. Independent predictions from all sampled clips are aggregated to derive a consensus. The loss value l^{task} is calculated based on this video-level consensus:

$$l^{task} = \mathcal{L}^{task}(\mathcal{G}(p_{\tau_1}, p_{\tau_2}, ..., p_{\tau_{N_{train}}}), q), \quad (4)$$

where $\mathcal{G}$ is the prediction/score aggregation function (*e.g.*, mean-pooling). When N_{train} is larger than one, this formulation can be regarded as a form of multiple instance learning (MIL) [67]. At inference, we uniformly sample N_{test} clips of the same duration as training clips, then aggregate predictions from all N_{test} clips to form our final prediction.

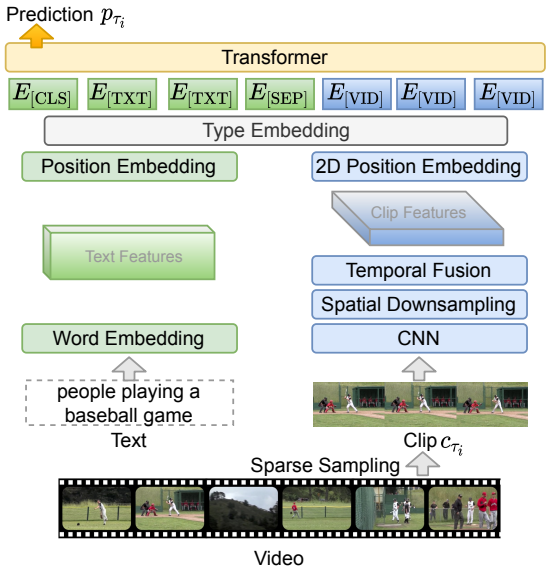

Figure 2: Overview of CLIPBERT architecture. For simplicity, we only show an illustration of producing prediction for a single sampled clip. When multiple clips are used, their predictions are fused together as the final prediction.

CLIPBERT's sparse training strategy can be interpreted as a type of data augmentation: different subsets of clips from the same video are used at different training steps, which improves the model's generalization ability. In this sense, it is analogous to random cropping [53, 19] commonly used in image classification tasks. It also takes inspiration from action recognition methods [52, 59, 63, 13], where a video classifier is trained on sampled clips.

Model Architecture. Figure 2 gives an overview of CLIPBERT architecture. For the vision encoder $\mathcal{F}_v$, we use a 2D CNN architecture ResNet-50 [19] instead of 3D architectures (such as C3D [59] or I3D [3]), because 2D models typically consume less memory and run faster. Besides, 2D CNNs have proved to work reasonably well on video understanding tasks such as action recognition [63, 48]. Specifically, we take the first 5 Conv blocks of ResNet-50 [19] and add an extra Conv layer and a 2×2 max-pooling layer to reduce its output feature depth and spatial size, following Pixel-BERT [20]. For each sampled clip, we uniformly sample T frames and obtain T feature maps. A temporal fusion layer $\mathcal{M}$ (*e.g.*, mean-pooling) is applied to aggregate the frame-level feature maps at the temporal axis to obtain a single clip-level feature map. We then add a row-wise and a column-wise position embedding to each feature vector based on their 2D position. These embeddings are the same trainable position embeddings as in BERT [10]. Collectively, these two position embeddings are indicative of 2D spatial locations of the features, which can be viewed as a *2D position embedding*. The resulting feature map is flattened into an embedding sequence to represent the clip.

We use a trainable word embedding layer as our language encoder $\mathcal{F}_l$ to encode language tokens and add trainable position embeddings to encode positional information of the tokens. Next, we add different type embeddings [10] to both clip and text embeddings to indicate their source type. These two sequences are then concatenated as inputs to a 12-layer transformer [61, 10] for cross-modal fusion. Special tokens `[CLS]` and `[SEP]` are added in this process following [10]. Given a downstream task, we add a task-specific prediction head with the last-layer `[CLS]` representation as input (*e.g.*, using a two-layer MLP with softmax to produce scores for text-to-video retrieval).

Weight Initialization and Pre-training. We initialize the ResNet-50 layers with weights from grid-feat [22, 50]. It is trained on Visual Genome [27] for object detection and attribute classification, and produces effective grid features for image VQA tasks [2, 18]. Input frames are resized to have a maximum longer side of L while keeping the aspect ratios, and the shorter side is zero-padded to be L as well [45]. We initialize the transformer and word embedding layers from BERT-base [10], pre-trained on BookCorpus [81] and Wikipedia. These weights are trained separately for their individual single-modality tasks, thus simply combining them together in a single framework for downstream task training may result in sub-optimal performance. Although pre-training the whole model end-to-end with large-scale video-text datasets such as HowTo100M [43] are appealing, we are restricted by the enormous computation cost.[2] Luckily, as we use 2D CNN as our vision encoder, CLIPBERT is able to directly take image-text pairs as inputs for training. Thus, we leverage large-scale image-text datasets (COCO Captions [4] and Visual Genome Captions [27]) to perform cross-modal pre-training [58, 41, 20]. Specifically, we use masked language modeling [10] and image-text matching [58, 41] objectives. By default, we finetune our model from this pre-trained weights for downstream video-text tasks. The impact of different weight initialization strategies is examined in Section 4.3.

Implementation Details. We perform image-text pre-training on COCO Captions [4] and Visual Genome Captions [27]. These two datasets contain a total of 5.6M training image-text pairs on 151K images. This is the same data used in UNITER's [5] in-domain pre-training. We use input image size L=768, and the resulting feature map from the vision encoder contains 144 pixels. To improve generalization and reduce computation cost, during pre-training, we follow Pixel-BERT [20] to use pixel random sampling that samples 100 pixels from the encoded feature map as the input to the transformer layers. Note that we only apply pixel random sampling for pre-training, and always use the full feature map for downstream tasks to avoid misalignment in training and inference [20]. We use WordPiece embeddings [68] and keep at most 20 tokens from the caption. We then randomly mask 15% of the tokens for masked language modeling. For each image-caption pair, with a probability of 0.5, we replace the ground-truth caption with a randomly sampled caption from another image to form a negative pair for image-text matching. We use AadmW [40] to optimize end-to-end model training, with an initial learning rate of 5e-5, β_1=0.9, β_2=0.98, and use learning rate warmup over the first 10% training steps followed by linear decay to 0. Our model is implemented in PyTorch [46] and transformers [66]. It is trained for 40 epochs with mixed precision, on 8 NVIDIA V100 GPUs with a batch size of 32 per GPU. The whole training process takes 4 days to complete.

For downstream finetuning, we use the same training and optimizer configurations except that the default input image size is set to 448 (due to the typically lower resolution of videos compared to images). Since downstream datasets vary in scale and domain, we use task-specific learning rates and training epochs based on validation performance.

4. Experiments

In this section, we evaluate CLIPBERT on two popular video-and-language tasks, text-to-video retrieval and video question answering, across six different datasets. We also provide extensive ablation studies to analyze the key factors that contribute to CLIPBERT's success.

4.1. Downstream Tasks

Text-to-Video Retrieval. (i) **MSRVTT** [72] contains 10K YouTube videos with 200K descriptions. We follow [74, 43], using 7k train+val videos for training and report results on the 1K test set [74]. We also create a local val set by sampling 1K video-caption pairs from unused test videos for our ablation study. (ii) **DiDeMo** [1] contains 10K Flickr videos annotated with 40K sentences. (iii) **ActivityNet Captions** [26] contains 20K YouTube videos annotated with 100K sentences. The training set contains 10K videos, and we use val1 set with 4.9K videos to report results. For MSRVTT, we evaluate standard sentence-to-video retrieval. For DiDeMo and ActivityNet Captions, we follow [77, 38] to evaluate paragraph-to-video retrieval, where all sentence descriptions for a video are concatenated to form a paragraph for retrieval. We use average recall at K (R@K) and median rank (MdR) to measure performance.

Video Question Answering. (i) **TGIF-QA** [21] contains 165K QA pairs on 72K GIF videos. We experiment with 3 TGIF-QA tasks: *Repeating Action* and *State Transition* for multiple-choice QA, and *Frame QA* for open-ended QA. We leave the *Count* task as future work as it requires directly modeling full-length videos. (ii) **MSRVTT-QA** [71]

[2] [42] reports that pre-training I3D [3] with offline extracted text features on HowTo100M requires ~3 days with 64 Cloud TPUs v3.

Figure 3: Average video length in different datasets.

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.

is created based on videos and captions in MSRVTT, containing 10K videos and 243K open-ended questions. (iii) **MSRVTT multiple-choice test** [74] is a multiple-choice task with videos as questions, and captions as answers. Each video contains 5 captions, with only one positive match. For all the QA tasks, we use standard train/val/test splits and report accuracy to measure performance.

Figure 3 shows a comparison of average video length of evaluated datasets. Videos across datasets demonstrate considerable difference in domains and lengths, ranging from 3-second generic-domain GIF videos in TGIF-QA to 180-second human activity videos in ActivityNet Captions.

4.2. Analysis of Sparse Sampling

We conduct comprehensive ablation studies concerning various aspects of CLIPBERT's design in this section and Section 4.3. If not otherwise stated, we randomly sample a single frame (N_{train}=1 and T=1) from full-length videos for training, and use the middle frame (N_{test}=1) for inference, with input image size L=448. All ablation results are on MSRVTT retrieval local val and MSRVTT-QA val sets.

Do we need larger input image size? We compare models with different input image sizes $L \in \{224, 448, 768, 1000\}$, results shown in Table 1. Compared to the model with L=224, larger input resolution improves performance on the retrieval task, while maintaining a similar performance on the QA task. The best performance is achieved at around L=448. Further increasing the resolution does not provide significant performance boost. [22] shows that increasing input image size from 448 to 1333 always improves image VQA [2] performance with no sign of saturation, while we observe the performance converges at around 448 for MSRVTT retrieval and QA. This is potentially because VQA images are typically of higher raw resolution than MSRVTT videos (we are only able to obtain videos at a maximum height of 240 pixels). We expect higher resolution videos could further improve model performance.

Do we need densely sampled frames? A common practice for video understanding and video+language understanding is to model densely sampled frames from the original video (*e.g.*, [3, 70] sample frames at 25 frames per second). To understand the impact of using densely sampled frames, we conduct a set of controlled experiments. Specifically, we randomly sample a fixed-length 1-second clip from the video, then evenly sample T=$\{1, 2, 4, 8, 16\}$ frames within this clip for training. For inference, we use the middle clip of the video. When multiple frames are used (*i.e.*, T>1), we use mean pooling for temporal fusion.

$\mathcal{M}$	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.

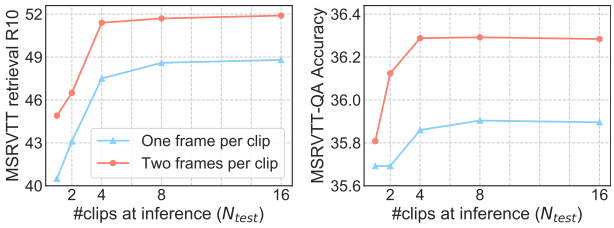

Figure 4: Impact of **#inference clips** (N_{test})**.**

We also experiment with variants using additional 3D convolutions before mean pooling: (i) Conv3D: a standard 3D convolution layer with kernel size 3, stride 1; (ii) Conv(2+1)D: a spatial and temporal separable 3D convolution [60, 70]. Adding 3D convolutions to 2D convolutions essentially leads to a design similar to Top-Heavy S3D architecture [70], which shows better performance than full 3D convolutions on video action recognition and runs faster.

Results are shown in Table 2. Overall, models that use 3D convolutions perform worse than models that adopt a simple mean pooling. For mean pooling, we observe that using two frames provides a notable improvement over using a single frame. However, models that use more than two frames perform similarly compared to the one using two frames, suggesting that two frames already represent enough local temporal information for the tasks.

Do more clips at inference help? At inference, we aggregate prediction scores from multiple densely sampled clips as the final score. To show how this strategy affects performance, we evenly sample $N_{test} \in \{1, 2, 4, 8, 16\}$ clips from a video and average their individual predictions at inference. For this experiment, we provide two models trained with different numbers of training frames per clip: one with a single frame and the other with two frames. Both models

$\mathcal{G}$	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.

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.

use a single clip for training. Results are shown in Figure 4. Adding more clips generally improves performance, especially the first few additions, but after a certain point performance saturates. For example, in Figure 4 (*left*), MSRVTT retrieval performance increases substantially as we use two and four clips, compared to using a single clip; then the performance gain gradually becomes marginal.

Do more clips in training help? We randomly sample N_{train} clips and aggregate scores from the clips with aggregation function $\mathcal{G}$ as the final score to calculate the training loss. When multiple clips are used, information from these samples is aggregated through multiple instance learning to maximize the utility of these clips. To understand how this strategy affects model performance, we evaluate model variants that use $N_{train} \in \{1, 2, 4, 8, 16\}$ at training. We also consider 3 different commonly used score aggregation functions for $\mathcal{G}$: mean pooling, max pooling, and LogSumExp [42]. In mean pooling and max pooling, the cross-clip pooling is performed over logits, followed by a softmax operator. In LogSumExp, logits from each clip are first fed through an element-wise exponential operator, followed by a cross-clip mean pooling. The aggregated output is further normalized by its own sum to make it eligible as a probability distribution. For simplicity, we always use the same aggregation function for training and inference. For a fair comparison, all models use a single frame per clip for training and 16 clips for inference, *i.e.*, T=1 and N_{test}=16.

Results are shown in Table 3. In general, adding more clips helps, and the second added clip gives the most performance gain. For example, for models with LogSumExp, N_{train}=2 improves retrieval R1 score of N_{train}=1 by 2.8%, while N_{train}=16 improves only 1.9% upon N_{train}=2, even though it adds much more clips. As for

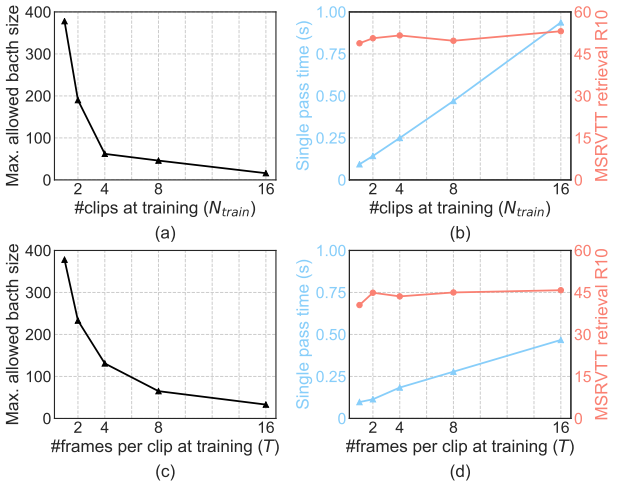

Figure 5: Memory and computation cost comparison w.r.t. different numbers of clips (N_{train}) or frames (T) at training. **(a)**: Maximum allowed batch size with fixed T=1. **(b)**: Time cost for a single forward and backward pass with fixed T=1, batch size 8. **(c)**: Maximum allowed batch size with fixed N_{train}=1. **(d)**: Time cost for a single forward and backward pass with fixed N_{train}=1, batch size 8. All experiments are conducted on a single NVIDIA V100 GPU with 32GB memory. MSRVTT retrieval performance is also added in (b, d) for reference. Best viewed in color.

score aggregation function $\mathcal{G}$, LogSumExp works the best.

Sparse Random Sampling *vs.* Dense Uniform Sampling. At each training step, CLIPBERT randomly samples only a single or a few short clips from a full-length video. Intuitively, this sparse random sampling strategy can be interpreted as a type of data augmentation where different subsets of clips for a video are used to calculate the loss at different training steps. To show the effectiveness of this approach, we compare CLIPBERT with a variant that uses uniformly sampled dense clips. Specifically, we use the same CLIPBERT architecture as before but always uses 16 uniformly sampled clips for training. Table 4 shows the comparison. Sparse random sampling with only 4 clips outperforms dense uniformly sampling with 16 clips across all metrics in both retrieval and QA tasks. Meanwhile, using 4 clips is much more memory and computation efficient than using 16 clips, as we show in the next paragraph. In addition to these two sampling approaches, it is also possible to choose clips using content-based methods such as [69]. However, this requires an extra non-trivial selection step, and may also remove some of the data augmentation effect brought by random sampling.

Memory and Computation Cost. Figure 5 shows a comparison of memory and computation cost *w.r.t.* different numbers of clips (N_{train}) or frames (T) at training. We observe that using more clips or more frames at training considerably increases memory demand and computational cost. For example, in Figure 5 (*a*), we see that the maximum allowed batch size for a single NVIDIA V100

Weight Initialization		MSRVTT Retrieval				MSRVTT-QA Acc.
CNN	transformer	R1	R5	R10	MdR	
random	random	0.3	0.4	0.9	506.0	28.05
random	$\text{BERT}_{\text{BASE}}$	0.0	0.2	0.7	505.0	31.72
TSN, K700	$\text{BERT}_{\text{BASE}}$	5.7	22.1	33.1	23.0	35.40
ImageNet	$\text{BERT}_{\text{BASE}}$	7.2	23.3	35.6	21.0	35.01
grid-feat	$\text{BERT}_{\text{BASE}}$	7.4	21.0	30.7	26.0	35.27
image-text pre-training		**10.2**	**28.6**	**40.5**	**17.0**	**35.73**

Table 5: Impact of **weight initialization strategy**.

GPU is 190 when N_{train}=2, compared to that of 16 when N_{train}=16. Meanwhile, in Figure 5 (*b*), we see that the time cost increases almost linearly with N_{train}, while the performance improvement on MSRVTT retrieval is less significant. These comparisons demonstrate the efficiency and effectiveness of the proposed sparse training strategy.

4.3. Analysis of Pre-training/End-to-end Training

Impact of Image-text Pre-training. Our model is initialized with image-text pre-training on COCO and Visual Genome Captions, to obtain better-aligned visual and textual representations. To validate the effectiveness of using image-text pre-training for weight initialization, we evaluate variants that use other initialization strategies. For CNN, we use weights from random initialization, image classification model pre-trained on ImageNet [9], frame-wise action recognition model TSN [63, 8] pre-trained on Kinetics-700 [54, 3], or detection model grid-feat [22] pre-trained on Visual Genome [27]. For transformer and word embedding layers, we use weights from random initialization or pre-trained $\text{BERT}_{\text{BASE}}$ model [10]. For random initialization, we use the default setup in PyTorch [46] and Transformer [65] libraries for CNN and transformer layers, respectively. Results are summarized in Table 5. We notice that randomly initializing CNN leads to massive performance degradation or even training failure, we hypothesize that it is mostly because of the difficulty of training large models on relatively small datasets (*e.g.*, MSRVTT retrieval train split: 7K videos). The best performance is achieved using image-text pre-trained weights, clearly indicating the benefit of utilizing image-text pre-training for video-text tasks.

Impact of End-to-End Training. The standard paradigm for video-and-language understanding is to train models with offline extracted features, in which the task objective does not affect the video and text encoding process. In this work, we train our model in an end-to-end manner, allowing the model to finetune feature representations by leveraging task supervision. In Table 6, we compare our model with variants that freeze portions of the parameters. Overall, the best performance is achieved by our model, showing the importance of end-to-end training. Note that all the models in Table 6 are finetuned from our end-to-end image-text pre-trained model, which partly resolves the multimodal feature disconnection issue in Section 1. Thus, we expect smaller improvement from further end-to-end finetuning.

Parameters Trainable?		MSRVTT Retrieval				MSRVTT-QA Acc.
$\mathcal{F}_v$	$\mathcal{F}_l$	R1	R5	R10	MdR	
✗	✗	8.0	27.2	38.9	17.0	**35.78**
✗	✓	9.0	27.5	39.4	18.0	35.50
✓	✓	**10.2**	**28.6**	**40.5**	**17.0**	35.73

Table 6: Impact of **end-to-end training**.

Main Conclusions from the analyses in Section 4.2 and Section 4.3 are summarized as follows: (*i*) Larger input image size improves model performance, but the gain diminishes when image size is larger than 448; (*ii*) Sparsely sample 2 frames from each clip performs on par with densely sample 16 frames, showing that just one or two frames are sufficient for effective video-and-language training; mean-pooling is more effective than 3D Conv when fusing information from different frames; (*iii*) More clips at inference helps improve model performance; prediction aggregation strategy across clips affects final performance; (*iv*) Sparse (random) sampling is more effective than dense uniform sampling while being more memory and computation efficient; (*v*) Image-text pre-training benefits video-text tasks; and (*vi*) End-to-end training improves model performance.

4.4. Comparison to State-of-the-art

We compare CLIPBERT with state-of-the-art methods on text-to-video retrieval and video question answering. We denote models using different sampling methods as CLIPBERT $N_{train}\times T$ (randomly sample N_{train} 1-sec clips for training, each contains T uniformly sampled frames of size L=448). We use LogSumExp to aggregate scores from multiple clips. At inference, if not otherwise stated, we aggregate scores from N_{test}=16 uniformly sampled clips.

Text-to-Video Retrieval. Table 7 summarizes results on text-to-video retrieval. In Table 7a, CLIPBERT achieves significant performance gain over existing methods on MSRVTT retrieval, including HT [43], ActBERT [80], and HERO [35], which are pre-trained on 136M clip-caption pairs from HowTo100M [43]. Under a fair comparison, CLIPBERT 4×1 outperforms HERO [35] by 3.0% on R@1. Note that HERO uses SlowFast [13] features extracted from full-length videos at a very dense frame rate of 21 FPS (*i.e.*, on average 310 frames per video for MSRVTT), while CLIPBERT 4×1 uses only 4 randomly sampled frames. When more frames are used, CLIPBERT 8×2 achieves even higher performance, surpassing HERO by 5.2%. Compared to the HERO ASR model that uses extra input from Automatic Speech Recognition (ASR), CLIPBERT still obtains 1.5% higher R1 score.

Similarly, on DiDeMo and ActivityNet Captions paragraph-to-video retrieval tasks (Table 7b and Table 7c), we notice our best CLIPBERT models outputform CE [38] by 4.3% and 3.1% on R1, respectively, despite CE's use of

Method	R1	R5	R10	MdR
HERO [35] ASR, PT	20.5	47.6	60.9	-
JSFusion [74]	10.2	31.2	43.2	13.0
HT [43] PT	14.9	40.2	52.8	9.0
ActBERT [80] PT	16.3	42.8	56.9	10.0
HERO [35] 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 [38]	16.1	41.1	-	8.3
S2VT [62]	11.9	33.6	-	13.0
FSE [77]	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 [38]	18.2	47.7	-	6.0
MMT [14]	22.7	54.2	93.2	5.0
MMT [14] PT	28.7	61.4	94.5	3.3
Dense [26]	14.0	32.0	-	34.0
FSE [77]	18.2	44.8	-	7.0
HSE [77]	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 [21]	60.8	67.1	49.3
Co-Memory [16]	68.2	74.3	51.5
PSAC [36]	70.4	76.9	55.7
Heterogeneous Memory [11]	73.9	77.8	53.8
HCRN [29]	75.0	81.4	55.9
QueST [23]	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 [21] (by [11])	30.9
Co-Memory [16] (by [11])	32.0
AMU [71]	32.5
Heterogeneous Memory [11]	33.0
HCRN [29]	35.6
CLIPBERT 4×1	**37.0**
CLIPBERT 8×2	**37.4**

(b) MSRVTT-QA test set.

Method	Accuracy
SNUVL [75] (by [74])	65.4
ST-VQA [21] (by [74])	66.1
CT-SAN [76] (by [74])	66.4
MLB [25] (by [74])	76.1
JSFusion [74]	83.4
ActBERT [80] PT	85.7
CLIPBERT 4×1	**87.9**
CLIPBERT 8×2	**88.2**

(c) MSRVTT multiple-choice test.

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

appearance, scene, motion, face, audio, OCR, ASR features densely extracted from 11 different models. ActivityNet Caption videos are on average 180-second long. In Table 7c we show CLIPBERT performs competitively with existing methods that model long-range relations in this dataset. Especially, CLIPBERT obtains 0.8% higher R1 than HSE [77] and is competitive compared to MMT [14] that uses extra audio features[3], even though CLIPBERT 4×2* (N_{test}=20) samples only 8-second clips from 180-second videos at each training step, and uses only 40-second content for inference. We expect CLIPBERT's performance to be further improved by sampling more clips during training and inference. Meanwhile, we also encourage future work to explore combining extra input signals, such as audio, into the CLIPBERT framework for better performance.

Video Question Answering. Table 8 shows results on video question answering. Across all three tasks, CLIPBERT achieves significant performance gain. In Table 8a, CLIPBERT 1×1 outperforms prior state-of-the-art QueST [23] by 6.9%, 6.8%, and 0.6% on TGIF-QA Action, Transition, and FrameQA tasks, respectively. This is especially surprising considering CLIPBERT 1×1 uses only a single randomly sampled frame from the videos at each training step, while QueST uses 10 uniformly sampled frames. Moreover, when using only a single frame (the middle frames of the videos) for inference, CLIPBERT 1×1 (N_{test}=1) already far outperforms QueST on Action and Transition, and is on par with QueST on FrameQA. In Table 8b, CLIPBERT 4×1 outperforms HCRN [29] by 1.4% on MSRVTT-QA. Note that HCRN adopts a sophisticated hierarchical relation modeling network over the entire video of 24 clips at training, while we use only four randomly sampled frames. Using more frames further increases this performance gap to 1.8%. Table 8c shows CLIPBERT 8×2 improves ActBERT [80] model pre-trained on HowTo100M by 2.5%, on MSRVTT multiple choice test task.

[3] [14] shows that adding audio features greatly improves performance.

5. Conclusion

We present CLIPBERT, a generic framework for end-to-end video-and-language learning, which adopts sparse sampling to use only a few sampled short clips from the videos at each training step. Experiments across diverse tasks show that CLIPBERT outperforms (or is on par with) state-of-the-art methods with densely sampled offline features, suggesting that the "*less is more*" principle is highly effective in practice. Comprehensive ablation studies reveal several key factors that lead to this success, including sparse sampling, end-to-end training, and image-text pre-training.

Acknowledgements: This research was partially done when Jie was an intern with Microsoft. He was later supported at UNC by NSF Award #1562098, DARPA KAIROS Grant #FA8750-19-2-1004, ARO-YIP Award #W911NF-18-1-0336, and Microsoft Investigator Fellowship. The views contained in this article are those of the authors and not of the funding agency.

References

[1] Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. Localizing moments in video with natural language. In *ICCV*, 2017.

[2] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *ICCV*, 2015.

[3] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017.

[4] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco captions: Data collection and evaluation server. *arXiv*, 2015.

[5] Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. Uniter: Learning universal image-text representations. In *ECCV*, 2020.

[6] Jaemin Cho, Jie Lei, Hao Tan, and Mohit Bansal. Unifying vision-and-language tasks via text generation. *arXiv*, 2021.

[7] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. In *ICLR*, 2020.

[8] MMAction2 Contributors. Openmmlab's next generation video understanding toolbox and benchmark. https://github.com/open-mmlab/mmaction2, 2020.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

[11] Chenyou Fan, Xiaofan Zhang, Shu Zhang, Wensheng Wang, Chi Zhang, and Heng Huang. Heterogeneous memory enhanced multimodal attention model for video question answering. In *CVPR*, 2019.

[12] Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition. In *CVPR*, 2020.

[13] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *ICCV*, 2019.

[14] Valentin Gabeur, Chen Sun, Karteek Alahari, and Cordelia Schmid. Multi-modal transformer for video retrieval. In *ECCV*, 2020.

[15] Zhe Gan, Yen-Chun Chen, Linjie Li, Chen Zhu, Yu Cheng, and Jingjing Liu. Large-scale adversarial training for vision-and-language representation learning. In *NeurIPS*, 2020.

[16] Jiyang Gao, Runzhou Ge, Kan Chen, and Ram Nevatia. Motion-appearance co-memory networks for video question answering. In *CVPR*, 2018.

[17] Jiyang Gao, Chen Sun, Zhenheng Yang, and Ram Nevatia. Tall: Temporal activity localization via language query. In *ICCV*, 2017.

[18] Danna Gurari, Qing Li, Abigale J Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P Bigham. Vizwiz grand challenge: Answering visual questions from blind people. In *CVPR*, 2018.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[20] Zhicheng Huang, Zhaoyang Zeng, Bei Liu, Dongmei Fu, and Jianlong Fu. Pixel-bert: Aligning image pixels with text by deep multi-modal transformers. *arXiv*, 2020.

[21] Yunseok Jang, Yale Song, Youngjae Yu, Youngjin Kim, and Gunhee Kim. Tgif-qa: Toward spatio-temporal reasoning in visual question answering. In *CVPR*, 2017.

[22] Huaizu Jiang, Ishan Misra, Marcus Rohrbach, Erik Learned-Miller, and Xinlei Chen. In defense of grid features for visual question answering. In *CVPR*, 2020.

[23] Jianwen Jiang, Ziqiang Chen, Haojie Lin, Xibin Zhao, and Yue Gao. Divide and conquer: Question-guided spatio-temporal contextual attention for video question answering. In *AAAI*, 2020.

[24] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv*, 2017.

[25] Jin-Hwa Kim, Kyoung-Woon On, Woosang Lim, Jeonghee Kim, Jung-Woo Ha, and Byoung-Tak Zhang. Hadamard product for low-rank bilinear pooling. In *ICLR*, 2016.

[26] Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. Dense-captioning events in videos. In *ICCV*, 2017.

[27] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *IJCV*, 2017.

[28] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *ICLR*, 2020.

[29] Thao Minh Le, Vuong Le, Svetha Venkatesh, and Truyen Tran. Hierarchical conditional relation networks for video question answering. In *CVPR*, 2020.

[30] Jie Lei, Liwei Wang, Yelong Shen, Dong Yu, Tamara L Berg, and Mohit Bansal. Mart: Memory-augmented recurrent transformer for coherent video paragraph captioning. In *ACL*, 2020.

[31] Jie Lei, Licheng Yu, Mohit Bansal, and Tamara L Berg. Tvqa: Localized, compositional video question answering. In *EMNLP*, 2018.

[32] Jie Lei, Licheng Yu, Tamara L Berg, and Mohit Bansal. Tvqa+: Spatio-temporal grounding for video question answering. In *ACL*, 2020.

[33] Jie Lei, Licheng Yu, Tamara L Berg, and Mohit Bansal. Tvr: A large-scale dataset for video-subtitle moment retrieval. In *ECCV*, 2020.

[34] Gen Li, Nan Duan, Yuejian Fang, Ming Gong, Daxin Jiang, and Ming Zhou. Unicoder-vl: A universal encoder for vision and language by cross-modal pre-training. In *AAAI*, 2020.

[35] Linjie Li, Yen-Chun Chen, Yu Cheng, Zhe Gan, Licheng Yu, and Jingjing Liu. Hero: Hierarchical encoder for video+ language omni-representation pre-training. In *EMNLP*, 2020.

[36] Xiangpeng Li, Jingkuan Song, Lianli Gao, Xianglong Liu, Wenbing Huang, Xiangnan He, and Chuang Gan. Beyond rnns: Positional self-attention with co-attention for video question answering. In *AAAI*, 2019.

[37] Yuncheng Li, Yale Song, Liangliang Cao, Joel Tetreault, Larry Goldberg, Alejandro Jaimes, and Jiebo Luo. Tgif: A new dataset and benchmark on animated gif description. In *CVPR*, 2016.

[38] Yang Liu, Samuel Albanie, Arsha Nagrani, and Andrew Zisserman. Use what you have: Video retrieval using representations from collaborative experts. In *BMVC*, 2020.

[39] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv*, 2019.

[40] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

[41] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *NeurIPS*, 2019.

[42] Antoine Miech, Jean-Baptiste Alayrac, Lucas Smaira, Ivan Laptev, Josef Sivic, and Andrew Zisserman. End-to-end learning of visual representations from uncurated instructional videos. In *CVPR*, 2020.

[43] Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. In *ICCV*, 2019.

[44] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NeurIPS*, 2013.

[45] Duy-Kien Nguyen, Vedanuj Goswami, and Xinlei Chen. Revisiting modulated convolutions for visual counting and beyond. *arXiv*, 2020.

[46] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

[47] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014.

[48] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *CVPR*, 2017.

[49] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 2020.

[50] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015.

[51] Anna Rohrbach, Marcus Rohrbach, Niket Tandon, and Bernt Schiele. A dataset for movie description. In *CVPR*, 2015.

[52] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NeurIPS*, 2014.

[53] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[54] Lucas Smaira, João Carreira, Eric Noland, Ellen Clancy, Amy Wu, and Andrew Zisserman. A short note on the kinetics-700-2020 human action dataset. *arXiv*, 2020.

[55] Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. Vl-bert: Pre-training of generic visual-linguistic representations. In *ICLR*, 2020.

[56] Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. Videobert: A joint model for video and language representation learning. In *ICCV*, 2019.

[57] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[58] Hao Tan and Mohit Bansal. Lxmert: Learning cross-modality encoder representations from transformers. In *EMNLP*, 2019.

[59] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015.

[60] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*, 2018.

[61] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[62] Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney, and Kate Saenko. Translating videos to natural language using deep recurrent neural networks. In *NAACL*, 2015.

[63] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016.

[64] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018.

[65] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing. *arXiv*, 2019.

[66] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *EMNLP: System Demonstrations*, 2020.

[67] Jiajun Wu, Yinan Yu, Chang Huang, and Kai Yu. Deep multiple instance learning for image classification and auto-annotation. In *CVPR*, 2015.

[68] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv*, 2016.

[69] Zuxuan Wu, Caiming Xiong, Chih-Yao Ma, Richard Socher, and Larry S Davis. Adaframe: Adaptive frame selection for fast video recognition. In *CVPR*, 2019.

[70] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *ECCV*, 2018.

[71] Dejing Xu, Zhou Zhao, Jun Xiao, Fei Wu, Hanwang Zhang, Xiangnan He, and Yueting Zhuang. Video question answering via gradually refined attention over appearance and motion. In *ACM MM*, 2017.

[72] Jun Xu, Tao Mei, Ting Yao, and Yong Rui. Msr-vtt: A large video description dataset for bridging video and language. In *CVPR*, 2016.

[73] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, 2019.

[74] Youngjae Yu, Jongseok Kim, and Gunhee Kim. A joint sequence fusion model for video question answering and retrieval. In *ECCV*, 2018.

[75] Youngjae Yu, Hyungjin Ko, Jongwook Choi, and Gunhee Kim. Video captioning and retrieval models with semantic attention. *arXiv*, 2016.

[76] Youngjae Yu, Hyungjin Ko, Jongwook Choi, and Gunhee Kim. End-to-end concept word detection for video captioning, retrieval, and question answering. In *CVPR*, 2017.

[77] Bowen Zhang, Hexiang Hu, and Fei Sha. Cross-modal and hierarchical modeling of video and text. In *ECCV*, 2018.

[78] Luowei Zhou, Hamid Palangi, Lei Zhang, Houdong Hu, Jason J Corso, and Jianfeng Gao. Unified vision-language pre-training for image captioning and vqa. In *AAAI*, 2020.

[79] Luowei Zhou, Chenliang Xu, and Jason J Corso. Towards automatic learning of procedures from web instructional videos. In *AAAI*, 2018.

[80] Linchao Zhu and Yi Yang. Actbert: Learning global-local video-text representations. In *CVPR*, 2020.

[81] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *ICCV*, 2015.