TACo: Token-aware Cascade Contrastive Learning for Video-Text Alignment

Jianwei Yang
Microsoft Research
jianwyan@microsoft.com

Yonatan Bisk
Carnegie Mellon University
ybisk@cs.cmu.edu

Jianfeng Gao
Microsoft Research
jfgao@microsoft.com

Abstract

Contrastive learning has been widely used to train transformer-based vision-language models for video-text alignment and multi-modal representation learning. This paper presents a new algorithm called Token-Aware Cascade contrastive learning (TACo) that improves contrastive learning using two novel techniques. The first is the token-aware contrastive loss which is computed by taking into account the syntactic classes of words. This is motivated by the observation that for a video-text pair, the content words in the text, such as nouns and verbs, are more likely to be aligned with the visual contents in the video than the function words. Second, a cascade sampling method is applied to generate a small set of hard negative examples for efficient loss estimation for multi-modal fusion layers. To validate the effectiveness of TACo, in our experiments we finetune pretrained models for a set of downstream tasks including text-video retrieval (YouCook2, MSR-VTT and ActivityNet), video action step localization (CrossTask), video action segmentation (COIN). The results show that our models attain consistent improvements across different experimental settings over previous methods, setting new state-of-the-art on three public text-video retrieval benchmarks of YouCook2, MSR-VTT and ActivityNet.

1. Introduction

Aligning or grounding language to videos is a challenging topic in the context of vision-language (VL) research as it requires the model to understand contents, dynamics, and causality presented in videos [3]. Inspired by the success of BERT [10] in natural language processing, there is a growing interest in applying transformer-based multi-modal models for video-text alignment and representation learning [40, 39, 59, 32, 14, 27]. These models are typically pretrained on large amounts of noisy video-text pairs using contrastive learning [34, 33], and then applied in a zero-shot manner or finetuned for various downstream tasks, such as text-video retrieval [51], video action step localization [60], video action segmentation [42], video question answering [43, 26] and video captioning [57].

In this paper, we present a new variant of contrastive learning, Token-Aware Cascade contrastive learning (TACo) to improve the video-text alignment for both large-scale pretraining and downstream specific tasks. As the name indicates, TACo makes two modifications to the conventional contrastive learning used in video-language domain. The first is the token-aware contrastive loss which is computed by taking into account the syntactic classes of words. This is motivated by the observation that, given a video and its corresponding text, content words, such as nouns and verbs, are more likely than function words to be aligned with (or grounded to) visual contents in the video. Conventional contrastive learning typically compute the loss after aggregating over all the words in the text and frames in the video (loss $L_1$ or $L_3$ in Fig. 1). In contrast, the token-aware contrastive loss is computed using only a subset of words whose syntactic classes belong to a predefined set (e.g., nouns and verbs), which forces the grounding of individual words to the video (loss $L_2$). For example, we pay particular attention to the words “add”, “tomatos”, “pan” and “stir” in Fig. 1.

Figure 1: The proposed token-aware cascade contrastive learning pipeline. We compute three contrastive losses: 1) sentence-level loss $L_1$ over all negative examples; 2) token-level loss $L_2$ on content words (noun, verb) over all negative examples; 3) sentence-level loss $L_3$ over hard negative examples sampled based on $L_1$ and $L_2$ online.
The second technique we introduce is a cascade sampling method to find a small set of hard negative examples for training the multi-modal fusion layers. Consider a batch of $K$ video-text pairs. For each of the video-text pairs, the ideal case is that we use the remaining $K - 1$ negative videos or texts to compute the contrastive loss after multi-modal fusion. However, the cost of computing the contrastive loss quickly becomes prohibitive when it is coupled with multi-modal fusion layers, considering its high complexity $O(K^2 \times L^2)$ where $L$ is total number of visual and textual tokens. A conventional way to address this is using random sampling to select a small subset of negative pairs. In this paper, instead of random sampling, we propose a cascade sampling method as shown in the top-right of Fig. 1 to efficiently select a small set of hard negative examples on the fly during training. It leverages the video-text alignment scores computed in $L_1$ and $L_2$ before multi-modal fusion layers, and helps to learn the multi-modal fusion layers more effectively without any extra overhead.

We perform a comprehensive empirical study to validate the effectiveness of TACo in both pretraining and dataset-specific scenarios. We apply TACo and different variants of contrastive losses to train or pretrain and finetune on various downstream tasks including text-video retrieval (YouCook2, MSR-VTT and ActivityNet) [57, 51, 12], video action step localization (CrossTask) [60] and action segmentation (COIN) [42]. Our results show that TACo improves the text-video retrieval performance over current state-of-the-art across three benchmarks. Furthermore, the learned multi-modal representation and video representations can be effectively transferred to CrossTask and COIN, and achieve better or comparable performance to current state-of-the-art methods.

2. Related work

Video-language pretraining. Realistic application scenarios around videos have prompted emergence of various video-language tasks, such as text-video retrieval [29, 54, 52], video question answering [20, 26], video captioning [53, 58], etc. Inspired by the success of BERT for large-scale pretraining in language domain [10], transformers have been employed in the video-language domain [40, 59, 32, 27] as well as image-language domain [41, 31, 56, 28]. Combined with large scale datasets, e.g. Howto100M [34] this approach has proven to be effective on various downstream tasks. Depending on the tasks of interest, some approaches train a multi-modal transformer using a combination of multiple losses including video-text alignment [40, 59, 32, 27], masked token (words/frames/objects) prediction [40, 59, 32], and frame order prediction [27], etc. Some other approaches exploited various contrastive learning techniques to directly optimize the feature space without multi-modal fusion [34, 33, 30, 14]. In most of previous works, these two approaches were explored separately. Very recently, an updated version of [32] used two independent alignment losses before and after multi-modal fusion in a single framework. In this paper, however, these two losses cooperate closely with each other during training in that the earlier stage helps to discover the hard negatives while the multi-modal layers with more capacity help to tackle those hard samples particularly.

Video-text alignment. Aligning videos to text requires the model to understand motion and temporal coherence. Some works have relied on attention mechanisms to extract key information from videos [44, 54], while others preserve visual information by composing pairwise joint representation using 3D tensors [52] or use multi-level video encoders to separately encode the spatial and temporal cues [11]. These models usually rely on a rank or margin loss to learn the correct alignment for video-text pairs. Another line of work learns fine-grained or hierarchical alignment between videos and texts [55, 48, 6]. In [48], the authors proposed a fine-grained alignment by extracting the nouns and verbs from action phrase in a sentence and projecting them into a shared space with videos. Alternatively, the authors in [6] extract a hierarchical semantic graph and apply graph reasoning to achieve the alignment at different levels. Similar ideas have also been proposed in the image-text alignment by decomposing the images and texts into sub-tokens [25, 49]. Thus far, it has not been studied how these task-specific architectures can be integrated into large-scale pretraining. In this paper, we are the first to propose a simple yet effective token-aware contrastive loss for fine-grained alignment for pretraining and downstream tasks.

Negative sampling. Key to efficient contrastive training is a good source of negative examples. Most of current approaches use random sampling strategies for training video-text alignment [59, 32]. However, in the domain of image-text retrieval, a few works tried hard negative sampling to choose the hardest negatives for training. In [2, 13], the authors computed the alignment scores for all image-text pairs in a mini-batch and use the hardest negative sample to compute the marginal loss. However, this strategy can only be applied without multi-modal fusion. In those models which have multi-modal fusion layers for better representations [31, 8], the authors instead compute the matching score offline and then use it to sample hard negatives for finetuning image-text retrieval model, which however is difficult for large-scale pretraining due to the high computational cost. In this paper, our cascade hard negative mining is particularly designed to address these issues as we efficiently select the hard negative samples online before multi-modal fusion and send them to the fusion layers for computing the loss. As we will show in our experiments, this technique can be seamlessly applied to both large-scale pretraining and downstream tasks.
3. Method

3.1. Framework

As depicted in Fig. 1, our model has three components:

**Video encoding module** $f_{\theta_v}$. It is implemented by a stack of self-attention layers parameterized by $\theta_v$. Here, we assume the input video features have been already extracted using some pre-trained models such as 2D CNN (e.g., ResNet [18]) or 3D CNN (e.g., I3D [4], S3D [50]). Given the input video embeddings, video encoder starts with a linear layer to project them to the same dimension $d$ as following self-attention layers. We denote the output of our video encoder for a video clip by a sequence of $m$ features, $x = \{x^1, \ldots, x^m\} \in \mathbb{R}^{m \times d}$. The number of features $m$ depends on the choice of sampling frame rate and the video feature extractor, which we will discuss in Sec. 4.

**Language encoding module** $f_{\theta_l}$. We use pretrained tokenizers [47] and BERT [10] to tokenize the input texts and extract textual features, respectively. Given a raw sentence, we append a “[CLS]” and “[SEP]” to the beginning and end, respectively. At the top, we can obtain a sequence of $n$ textual features $y = \{y^1, \ldots, y^n\} \in \mathbb{R}^{n \times d}$. We ensure the output feature dimension of video encoder to be identical to that of language encoder. During training, we update the parameters $\theta_l$ in our language encoder to adapt to the texts in specific domain, e.g., cooking instructions in YouCook2 [57].

**Multi-modal fusion module** $f_{\theta_m}$. It also consists of self-attention layers with learnable parameters $\theta_m$. It takes video features $x \in \mathbb{R}^{m \times d}$ and text features $y \in \mathbb{R}^{n \times d}$ from two separate modalities as inputs and output the $(m + n)$ features $z = \{z_1, \ldots, z_{m+n}\} \in \mathbb{R}^{(m+n) \times d}$. To help it to distinguish the video and language tokens, we use a token type embedding layer to learn two embeddings and add them to the visual and textual tokens, separately. Similar to original Transformer [46], we include a positional embedding layer to encode the absolute token positions in the input sequence.

The above three components comprise our video-text alignment model which is then trained with the proposed token-aware cascade contrastive loss. We start with a brief review of conventional contrastive learning and then introduce the proposed technique.

3.2. Contrastive learning: a revisit

Given a set of $N$ video-text pairs $\{(v_i, t_i)\}_{i=1}^N$, our goal is to learn an optimal scoring function $s$ such that paired video and text $(v_i, t_i)$ have higher scores than all the other unmatched pairs $(v_j, t_k), j \neq k$. From the probabilistic perspective, aligning $v_i$ to $t_i$ is equivalent to maximizing the conditional probability $p(v_i | t_i)$ while minimizing the probability for all negative pairs $p(v_j | t_i), j \neq i$. According to [15, 36], $p(v_j | t_i)$ can be approximated by:

$$p(v_j | t_i) \approx \frac{\exp^s(v_j, t_i)}{\sum_{k=1}^{N} \exp^s(v_k, t_i)}$$

where $s(v, t)$ is the alignment score between $v$ and $t$; the denominator is a sum over all possible videos, which is a partition function for normalization. Adding cross-entropy loss on $p(v_j | t_i)$, we can then derive the NCE loss [15]:

$$L_{nce} = \sum_{i=1}^{N} \log p(v_i | t_i) \sim \log \left(\frac{\exp^s(v_i, t_i)}{\sum_{k \neq i} \exp^s(v_k, t_i)}\right)$$

The denominator in Eq. 2 requires a sum over all videos in a dataset, which is intractable in practice. Therefore, we usually compute the NCE loss on a mini-batch of $K (K \ll N)$ video-text pairs sampled from the whole dataset. Ideally, we want to learn the parameters $\theta = \{\theta_v, \theta_l, \theta_m\}$ of the model to minimize the above NCE loss, such that $\Delta = s(v_i, t_i) - s(v_j, t_i)$ is maximized over all tuples $(t_i, v_i, v_j), j \neq i$. A number of previous works used the above formula for contrastive learning [33, 59]. Meanwhile, there are some variants of computing contrastive loss in video-language representation learning. For example, [27, 14] omits the denominator and incorporate a margin $s(v_i, t_i) > s(v_j, t_i) + \delta, \forall j \neq i$ in a mini-batch. [32] optimizes binary cross-entropy (BCE) by assigning $(v_i, t_i)$ a positive label (1) and other pairs a negative label (0).

3.3. **TACo**: our approach

The way of using contrastive learning in previous works has two issues. First, the loss is computed at sentence-level by taking ‘[CLS]’ token [14] or the maximum over all tokens [33] in a sentence. Clearly, the content words (e.g., nouns, verbs) are more likely to align with the visual contents or concepts in the videos compared with function words (e.g., stop words). Second, the high computational cost in multi-modal fusion layers hinder the usage of large batch of negative samples, which however is essential to contrastive learning [33, 17, 7]. Motivated by these two issues, we introduce **TACo**, a simple yet effective method to improve the contrastive learning. We elaborate below how these contrastive losses are computed.

Given the $K$ video-text pairs $\{(v_i, t_i)\}_{i=1}^K$ in a mini-batch, we first use our video encoder $f_{\theta_v}$ and language encoder $f_{\theta_l}$ to obtain a batch of video features $X = \{x_1, \ldots, x_K\} \in \mathbb{R}^{K \times m \times d}$ and text features $Y = \{y_1, \ldots, y_K\} \in \mathbb{R}^{K \times n \times d}$, respectively. Then, we average all tokens of a video clip $v_i$ to get $\bar{x}_i \in \mathbb{R}^{1 \times d}$, and take the first ‘[CLS]’ token for each text $t_i$ to get $\bar{y}_i \in \mathbb{R}^{1 \times d}$. Based
on \( x \) and \( \bar{y}_i \), we compute the sentence-level contrastive loss:

\[
L_1 = - \sum_{i=1}^{K} \log \left( \frac{\exp \xi_i, y_i / \tau_1}{\exp \xi_i, y_i / \tau_1 + \sum_{j \neq i} \exp \xi_j, y_j / \tau_1} \right) \tag{3}
\]

where \( \tau_1 \) is a scalar temperature parameter. In Eq. 3, the computation is simply a number of dot-products between video and text features. Giving such efficiency, we can use all the \( K - 1 \) negative samples in a mini-batch to compute the loss. Through this, we optimize \( \theta_s \) and \( \theta_t \) so as to project the video and text samples into an aligned feature space.

The ‘[CLS]’ token and average of video tokens in Eq. 3 overlooks the differences across tokens and frames, and thus may not provide the pressure to push individual tokens (e.g., nouns and verbs) to ground on the specific video contents. To encourage correct alignment, in addition to the sentence-level loss, we introduce a token-level contrastive loss:

\[
L_2 = - \sum_{i=1}^{K} \sum_{p \in P_i} \log \left( \frac{\exp s(x_i, y'_p)^{1/\tau_2}}{\exp s(x_i, y'_p)^{1/\tau_2} + \sum_{j \neq i} \exp s(x_j, y'_j)^{1/\tau_2}} \right) \tag{4}
\]

where \( \tau_2 \) is another scalar temperature parameter; \( P_i \) is the indices of tokens of interest in \( i \)-th text, and \( y'_p \) is the \( p \)-th token embedding in \( i \)-th text. \( s(\cdot) \) measures the similarity between video features and specific token embedding \( y'_p \). It first computes the dot-product between \( y'_p \in \mathcal{R}^{1 \times d} \) and all \( m \) video tokens \( x \in \mathcal{R}^{m \times d} \), and then take the maximum over \( m \) scores to get the final alignment score. Through Eq. 4, the model uses individual tokens as anchors to align with video, which is complementary to the sentence-level loss in Eq. 3. Similar to Eq. 3, we can compute this token-level contrastive loss efficiently, and thus use all the \( K - 1 \) negative samples. As a whole, these two losses are used to optimize \( \theta_s \) and \( \theta_t \) in a token-aware manner.

**Token of interest.** In Eq. 4, we need to decide which tokens should be included in \( P_i \). In this paper, we heuristically select nouns and verbs as the targets considering they are more “concrete” in the videos. In practice, nouns or verbs usually have different discriminativenss even if they are all the same type. For example, “man” is a noun but is less informative than “gymnast”. To reflect this, we further assign different words with different weights by computing their inverse document frequency (idf) [21]. A higher idf means it is more unique across the corpus, and hence will weigh more when computing the token-level contrastive loss. Another practical issue for computing the loss is that the tokens are usually sub-words due to the BERT tokenizer. Hence, for all tokens that belongs to the same word, we will assign the same weights accordingly.

After computing the token-aware contrastive loss, we feed the features from separate modalities to multi-modal fusion layers to enable more interactions between them two. Similar to previous work [59], we take the feature corresponding to the “[CLS]” in the \((m+n)\) outputs. We regard this as the summary of two modalities and then compute the contrastive loss:

\[
L_3 = - \sum_{i=1}^{K} \log \left( \frac{\exp w_{z^c_{j,i},v}^{1/\tau_3}}{\exp w_{z^c_{j,i},v}^{1/\tau_3} + \sum_{j \neq i} \exp w_{z^c_{j,i},v}^{1/\tau_3}} \right) \tag{5}
\]

where \( z^c_{j,i} \) is the multi-modal fusion output for “[CLS]” token taking \( x_j \) and \( y_i \) as inputs; \( w \in \mathcal{R}^{1 \times d} \) is the parameter in a linear layer\(^1\). Based on Eq. 5, we optimize all parameters in our model \( \theta = \{\theta_v, \theta_t, \theta_m\} \) in collaboration with Eq. 3 and Eq. 4.

In Eq. 5, a practical challenge is that we can hardly use all \((K-1)\) negative samples in the mini-batch, due to the high computational and memory cost in the multi-modal fusion. The \(O(d(m+n)^2)\) complexity of self-attention layer makes it intractable to pass all \( K \times K \) pairs into the multi-modal layers. Previous work solved this by performing random sampling to cut the number of negative samples to \( K'\). However, randomly choosing negative samples may result in sub-optimal learning since the pairs are scarce. We therefore introduce a cascade sampling strategy to find hard negatives instead of random ones.

**Cascade hard negative sampling.** To reduce the computational cost in Eq. 5, we choose among all possible video-text pairs a small subset which are most difficult. However, computing the alignment scores for all pairs using Eq. 5 and then select the hard negatives is a “chicken-and-egg” problem. Instead, we propose to use the similarities between all video-text pairs computed in Eq. 3 and Eq. 4 as the guidance. Specifically, for each text-video pair \((v_j, t_i)\), we take their global similarity \( \bar{x}_j, \bar{y}_i \) computed in Eq. 3 and token-level similarity by aggregating \( \sum_{p \in P_i} s(x_j, y'_p) \) for all tokens of interest in \( t_i \). Then we sum the two similarities as the alignment score for the given pair. For each text, we choose the top \( K' \) aligned negative videos and vice versa. The resulting \( 2K \times (K' + 1) \) pairs are then fed into the multi-modal fusion layers. Through this strategy, we can effectively select the difficult negative samples on the fly at no extra cost. Since the multi-modal fusion layers have more capacity (parameters) to distinguish these hard negatives from positive ones, our sampling strategy naturally prompts the cooperation between the three contrastive losses.

Finally, we present a comprehensive comparison to differentiate our model with previous works with respect to the used contrastive learning method in Table 1.

### 3.4. Objective

The training objective in our method is finding optimal \( \theta = \{\theta_v, \theta_t, \theta_m\} \) by minimizing the combination of the above three contrastive losses:

\[
\arg \min_{\theta_v, \theta_t, \theta_m} \frac{1}{N} \sum_{i=1}^{N} (L_1 + \lambda_1 L_2 + L_3) \tag{6}
\]

\(^1\)for clarity, we omit the bias term in the formula.
Table 1: A comparison of video-language pretraining methods regarding contrastive learning strategies. “Early stage” and “Later stage” mean computing the loss before and after multi-modal fusion, respectively. “Cascade” means using cascade hard negative sampling.

<table>
<thead>
<tr>
<th>Method</th>
<th>Token-aware</th>
<th>Early stage</th>
<th>Later stage</th>
<th>Cascade</th>
<th>Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>VideoBert [40]</td>
<td>x</td>
<td>x</td>
<td>✓</td>
<td>x</td>
<td>BCE</td>
</tr>
<tr>
<td>CBT [39]</td>
<td>x</td>
<td>x</td>
<td>✓</td>
<td>x</td>
<td>NCE</td>
</tr>
<tr>
<td>TIVE [34]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>Margin</td>
</tr>
<tr>
<td>MIL-NCE [33]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>NCE</td>
</tr>
<tr>
<td>ActBert [59]</td>
<td>x</td>
<td>x</td>
<td>✓</td>
<td>x</td>
<td>BCE</td>
</tr>
<tr>
<td>UniVL [32]</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>NCE</td>
</tr>
<tr>
<td>TACo (Ours)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>Margin</td>
</tr>
<tr>
<td>TJVE [34]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>NCE</td>
</tr>
</tbody>
</table>

$\lambda_t$ is the weight of token-level loss (0.5 by default). During inference, we make the prediction by summing the alignment scores from all the three scoring functions.

4. Experimental setup

4.1. Datasets

In our experiments, we train and evaluate our model on the following established benchmarks:

- **YouCook2** [57] consists of 2k videos about routine cooking activities of 89 recipes. Each video contains multiple video clips annotated with text descriptions by human annotators. Following [34, 33], we train our models on the training split, and report the text-video retrieval performance on around 3.5k validation clips.

- **MSR-VTT** [51] contains 10k video clips associated with 200k sentences. There are two validation splits used in previous work. In [30, 14], the training set has 9k clip-text pairs with the remaining 1k pairs for evaluation, which we denote by *split1*. In [52, 34, 33], 1k clip-text pairs are sampled from the 3k pairs in test set for evaluation, while the original 7k pairs are used for training. We denote this by *split2*. We report text-video retrieval results using both splits.

- **ActivityNet** [24]. It consists of 20K YouTube videos, each of which is associated with multiple human-annotated captions. Following [55, 14], we concatenate all the captions for a video into a paragraph and evaluate the paragraph-video retrieval on the “val1” split.

- **Howto100M** [34]. We compare with previous work under the pretraining protocol on Howto100M [34, 33, 59, 32]. It was collected from YouTube and contains over 1.2M narrated videos associated with automatically generated transcripts. Each video contains over 100 clips on average.

To further verify the transferrability of our learned multimodal representation from Howto100M, we also evaluate the action step localization and action segmentation on CrossTask [60] and COIN [42], respectively.

4.2. Settings

Previous work use a variety of different video and language representations which we find significantly affect the final performance. We summarize different choices below:

- **Video representations.** For 2D CNN, Resnet-152 [18] is used to extract feature map and then globally pooled to 2048-d [34, 32]. For 3D features, commonly used models are I3D [5], R(2+1)D [45] and S3D [50]. In [59], the authors further extract objects from the video clips. In [30, 14], the authors use collaborative experts to extract features from audio, scene, OCR, face, speech, etc.

- **Language representations.** There are primarily four variants: 1) GoogleNews pretrained word2vec (w2v) [35] used in [30, 34, 33]; 2) LSTM or Bidirectional LSTM [19]; 3) pretrained BERT [10] used in [40, 59, 32, 14] and 4) OpenAI-GPT [37] used in [30].

In this paper, we use a pretrained BERT-base model for language representation as in [59, 32]. For video features, following [34, 33, 32], we extract 2D CNN features using Resnet-152 (R-152) pretrained on ImageNet [9]. For 3D CNN features, we use I3D (with Resnext-101 backbone) pretrained on Kinetics-400 [22] and S3D [50] pretrained on Howto100M [33]. The off-the-shelf pretrained weights are provided by [16] and [33]. For simplicity, we denote them by I3D-X101 and S3D-HM in the following.

Another discrepancy among different methods is the number of self-attention layers used in the model. In [59], the authors use 12 multi-modal self-attention layers while 6 video encoder layers and 2 multi-modal fusion layers are used in [32]. Differently, 4 multi-modal self-attention layers are used in [14]. In this paper, for all our ablation studies below, we use 1 and 2 self-attention layers for our video encoder and multi-modal fusion, respectively. To compare with previous work on specific dataset, we use 2 video encoding layers. While pretraining the model with large-scale dataset Howto100M [34], we increase to 4 video encoding layers for comparable model capacity to previous works [59, 32, 14]. Note that this largest model is still smaller than or on par with the aforementioned methods.

4.3. Implementation details

For YouCook2 and MSR-VTT, the maximum number of video and text tokens are set to 48 and 30, respectively. For paragraph-video retrieval on ActivityNet, we set them both to 256. The 2D R-152 feature is extracted for one frame per second, and then globally pooled to 2048-d. For 3D CNN features, we follow [34] to sample video frames at 24 fps and extract an I3D-X101 feature every 16 frames. This results in 1.5 2048-d feature per second. For Eq. 3 and 4, we set the temperatures $\tau_1$ and $\tau_2$ both equal to 1.

**Training on separate datasets.** In this setting, we train models from scratch using the training set provided in YouCook2, MSR-VTT and ActivityNet separately. We train
merely trained with L

Video representations. We train our model with different video representations as described above and compare it with the baseline model which has identical architecture but merely trained with $L_3$ as depicted in Eq. 5. The baseline

The model for $30k$ iterations with batch size $128$. For each training sample, we use our cascade sampling strategy to sample $8$ hard negatives. We use Adam $[23]$ as the optimizer with initial learning rate $1e^{-4}$. A linear learning rate decay is applied after $5k$ warm-up iterations. The weight decay is set to $1e^{-5}$.

Pretraining and finetuning. We pretrain our model on HowTo100M $[34]$. Since the original annotated video clips in HowTo100M are usually short with a few seconds, we merge the adjacent clips so that the resulted text has at least $10$ words. We use Adam $[23]$ as the optimizer with initial learning rate $1e^{-4}$. We train the model for $500k$ iterations with batch size $64$, and also sample $8$ hard negatives for each sample using our cascade sampling strategy. After pretraining, we finetune the pretrained models on different datasets using the same setting as above except for a lower initial learning rate $2e^{-5}$ and less finetuning iterations $20k$.

Evaluation metrics. For text-video retrieval, we use Recall at different points (Recall@n or Rn, with n as a specific number) and Median Rank (MR) as the metrics following previous works $[59, 32]$. In all tables, we use ↑ or ↓ to indicate higher or lower is better, respectively.

5. Results

We first evaluate text-video retrieval performance and then study whether the learned representations can be transferred to other tasks on CrossTask and COIN.

5.1. Text-video retrieval

5.1.1 Comparing with baselines

We first show the comparisons with baselines to inspect the effects of different components in our model.

Video representations. We train our model with different video representations as described above and compare it with the baseline model which has identical architecture but merely trained with $L_3$ as depicted in Eq. 5. The baseline

<table>
<thead>
<tr>
<th>Video Representation</th>
<th>YouCook2</th>
<th>MSR-VTT (split1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-152, Baseline</td>
<td>4.1 13.2 19.4 81.0</td>
<td>16.4 42.6 55.8 8.0</td>
</tr>
<tr>
<td>R-152, Ours</td>
<td>4.6 14.1 20.4 71.0</td>
<td>18.9 46.2 58.8 7.0</td>
</tr>
<tr>
<td>S3D-HM, Baseline</td>
<td>2.1 8.1 12.7 125.0</td>
<td>14.7 40.83 53.2 9.0</td>
</tr>
<tr>
<td>S3D-HM, Ours</td>
<td>2.6 8.9 13.2 115.0</td>
<td>20.6 44.0 56.9 7.0</td>
</tr>
</tbody>
</table>

Table 2: Text-video retrieval performance on YouCook2 and MSR-VTT with different feature types. S3D pretrained on HowTo100M outperforms others with large margin.

Contrastive learning method has been adopted in a number of previous works $[59, 32]$. This comparison can verify the effectiveness of our proposed contrastive learning method considering two models have exactly the same number of parameters. In Table 2, we can see our proposed method outperforms baseline across all feature types introduced in Sec. 4.2 on both YouCook2 and MSR-VTT. Note that our model uses exactly the same number of parameters to the baseline model. These consistent improvements demonstrate the effectiveness and generalization ability of our proposed method. As mentioned above, we also observe the text-video retrieval performance significantly depends on the feature types. We can find 3D features (I3D-X101 and S3D-HM) in general outperform 2D feature (R-152), which is expected since 2D feature does not capture the motions in the videos. Among all three feature types, S3D-HM outperforms the other two with large margin, which demonstrates the potential to learn good video representation by pretraining on large-scale noisy dataset (Howto100M $[34]$). Because Howto100M mainly contains instructional videos, it is more close to YouCook2 than MSR-VTT, and hence we see more gain on YouCook2. These comparisons indicate video representations matter much to the final performance.

Component Analysis. In our method, we combine $L_1$, $L_2$, and $L_3$ during training and inference. Here, we study how they perform separately and contribute to the final performance. In Table 2, we use R-152+S3D-HM as the video feature and report the results with different loss combina-
As we can see, solely using $L_1$ (row 1) or $L_2$ (row 2) for contrastive learning results in sub-optimal video-text alignment. Simply combining them together (row 3) improves the performance on two datasets. This implies that different levels of contrastive learning can be complementary to each other, which supports our earlier hypothesis that these two losses are synergistic with each other for a better video-text alignment. When incorporating the hard negative mining via our cascade sampling strategy (row 4), it further improves the performance. Finally, we can see adding token-level contrastive loss $L_3$ can further improve the performance across all settings (row 5).

**Tokens of Interest.** We further study the effect of different tokens of interest on the model performance. By default, our model uses the noun and verb as the tokens of interest to compute the token-level contrast loss. Here, we vary them to other types such as adposition (adp) and determiner (det) for investigation. In Table 4, we replace “noun+verb” with “det-adp”, “noun” and “verb” and report the numbers on two text-video retrieval datasets. As we can see, using “det-adp” as the target tokens is worse than the baseline without any token-level contrastive loss. “noun” and “verb” can both improve the performance while “noun” is slightly better than “verb”. Finally, combining noun and verb together achieves the best performance. These results align with our intuition to use nouns and verbs as the target token for fine-grained alignment between texts and videos considering they are usually grounded to video contents.

### 5.1.2 Comparing with state-of-the-art

We compare with previous works under three protocols: 1) training and evaluating on separate datasets; 2) pretraining on Howto100M and evaluating zero-shot performance and 3) finetuning pretrained model on separate datasets.

**Results on separate datasets.** We separately show the comparisons on YouCook2, MSR-VTT and ActivityNet in Table 5, 6 and 7. For a fair comparison with previous works, we use the same or similar features as listed in the tables. As we can see, our method outperforms all previous work across all datasets. These results validates its effectiveness to learn video-text alignment. Note that previous works either use a variety of loss functions [32, 27] or a collection of multiple features [30, 14]. In contrast, we achieve the best performance using a simpler contrastive learning pipeline with smaller model size. This supports our earlier claim on the efficiency. Comparing the numbers in Table 2, Table 5 and Table 6, we can find our model achieves better performance with the same video features when using deeper video encoder (2 layers v.s. 1 layer).
**5.2. Other video-related tasks**

Following [34, 59, 32], we evaluate action step localization performance on CrossTask dataset [60]. It covers 18 tasks and each video contains multiple video segments annotated with action steps and natural language descriptions. Similar to [34, 59, 32], we use our model to compute the similarity between each frame and the action step descriptions, which results in a score matrix. Using the official algorithm provided by [60], we can find the optimal frame-wise order of action steps for a video. By comparing it with the ground-truth annotations, we compute the recall for each task and then do the average. According to the results in Table 9, our model achieves the best performance compared with previous works. This indicates that our model can learn good video-language representations.

We further evaluate our pretrained model on action segmentation task on COIN dataset, following [33, 59]. Unlike the above task, action segmentation does not rely on texts, and thus can be used to evaluate the learned video representation. As shown in Table 9, our method significantly outperforms MIL-NCE and ActBert, and achieves comparable performance to UniVL. This indicates that our model is also a good video representation learner.

**6. Conclusion**

In this paper, we introduced **TACo**, a simple yet effective contrastive learning method for learning video-text alignment. It is aimed at addressing two existing issues in current contrastive learning pipelines: missing fine-grained alignment and inefficient sampling for multi-modal fusion. Without introducing any extra parameters, our method achieved promising results on three text-video retrieval benchmarks under various evaluation protocols. We further demonstrated the learned representations can be effectively transferred to other tasks such as action step localization and segmentation. Based on all these encouraging results, we believe **TACo** is a good alternative to conventional contrastive learning pipeline.

---

Table 8: A complete comparison of **TACo** under zero-shot and finetuning evaluation protocols. Note that the zero-shot and upper part of finetuned performance for MSR-VTT is on *split2*, while the bottom is on *split1* for fair comparison.

<table>
<thead>
<tr>
<th>Method</th>
<th>Video</th>
<th>YouCook2</th>
<th>MSR-VTT</th>
<th>ActivityNet</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Zero-shot</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TJVE [34]</td>
<td>R-152+I-101</td>
<td>6.1 17.3</td>
<td>24.8 46.0</td>
<td>7.5 21.2</td>
</tr>
<tr>
<td>ActBERT [59]</td>
<td>O-101+R(2+d)</td>
<td>9.6 26.7</td>
<td>38.0 19.0</td>
<td>8.6 23.4</td>
</tr>
<tr>
<td>MIL-NCE [33]</td>
<td>S3D-HM</td>
<td>15.1 38.0</td>
<td>51.2 10.0</td>
<td>9.9 24.0</td>
</tr>
<tr>
<td><strong>TACo (Ours)</strong></td>
<td>S3D-HM</td>
<td><strong>19.9</strong></td>
<td><strong>43.2</strong></td>
<td><strong>35.7</strong></td>
</tr>
<tr>
<td><strong>Finetuned</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TJVE [34]</td>
<td>R-152+I3D-X101</td>
<td>8.2 24.5</td>
<td>35.3 24.0</td>
<td>14.9</td>
</tr>
<tr>
<td>UniVL [33]</td>
<td>S3D-HM</td>
<td>28.9 57.6</td>
<td>70.0 4.0</td>
<td>21.2</td>
</tr>
<tr>
<td><strong>TACo (Ours)</strong></td>
<td>S3D-HM</td>
<td><strong>29.6</strong></td>
<td><strong>59.7</strong></td>
<td><strong>72.7</strong></td>
</tr>
<tr>
<td>MMT [14]</td>
<td>Collaborative Experts</td>
<td>– – – –</td>
<td>– – – –</td>
<td>26.6</td>
</tr>
<tr>
<td><strong>TACo (Ours)</strong></td>
<td>R-152+S3D-HM</td>
<td>27.3</td>
<td>56.5</td>
<td>68.8</td>
</tr>
</tbody>
</table>

Table 9: Action step localization on CrossTask (avg. recall) and action segmentation on COIN (acc.).

<table>
<thead>
<tr>
<th>Method</th>
<th>CrossTask</th>
<th>COIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActBert [59]</td>
<td>41.4</td>
<td>57.0</td>
</tr>
<tr>
<td>MIL-NCE [33]</td>
<td>40.5</td>
<td>61.0</td>
</tr>
<tr>
<td>TVJE [34]</td>
<td>33.6</td>
<td>– –</td>
</tr>
<tr>
<td>UniVL [33]</td>
<td>42.0</td>
<td>70.0</td>
</tr>
<tr>
<td><strong>TACo (Ours)</strong></td>
<td>42.5</td>
<td>68.4</td>
</tr>
</tbody>
</table>

Zero-shot and finetuned performance. In Table 8, we show the comparisons across different models pretrained on Howto100M. In the upper part of the table, we compare the zero-shot performance on YouCook2 and MSR-VTT. We do not evaluate on ActivityNet since it has different number of input video tokens compared with the pretrained model and thus is not directly compatible to the pretrained model. As we can see, **TACo** outperforms previous works significantly on YouCook2 and slightly on MSR-VTT. Since YouCook2 has closer domain gap to Howto100M than MSR-VTT, the improvement brought by large-scale pretraining is more significant. However, on MSR-VTT, our model still outperforms MIL-NCE [33] which uses the same video features. In Fig. 2, we show the zero-shot performance on YouCook2 and MSR-VTT when pretraining our models with different contrastive losses as listed in Table 3. Accordingly, it shows our proposed contrastive losses gradually improve the performance, and combining all techniques achieves the best performance. Based on the pretrained model, we further finetune it on specific datasets. In our experiments, we use two feature S3D-HM and R-152+S3D-HM, to compare with the methods with the same/similar settings. As we can see, our model using S3D-HM outperforms UniVL [32] using the same feature but more video encoder layers (6). Different from zero-shot results, we observe more improvement on MSR-VTT than YouCook2 after finetuning. This implies that finetuning on specific datasets can compensate the domain gap to the pretraining datasets. To compare with the methods using features extracted from collaborative experts [14], we enrich our video representation by adding 2D R-152 feature, which achieves better performance on MSR-VTT, and better Recall@1 and Median Rank on ActivityNet. Note that this combination hurts the performance on YouCook2, and we witnessed a similar trend for models without pretraining in Table 2. Finally, comparing with the results without pretraining in Table 5, 6 and 7, we clearly find large-scale pretraining and finetuning brings substantial improvements consistently.
References


[50] Youngjae Yu, Jongseok Kim, and Gunhee Kim. A joint sequence fusion model for video question answering and re-


