Large Scale Video Representation Learning via Relational Graph Clustering

Hyodong Lee[*]
MIT
hyo@mit.edu

Joonseok Lee
Google Research
joonseok@google.com

Joe Yue-Hei Ng
Google Research
yhng@google.com

Paul Natsev
Google Research
natsev@google.com

Abstract

Representation learning is widely applied for various tasks on multimedia data, e.g., retrieval and search. One approach for learning useful representation is by utilizing the relationships or similarities between examples. In this work, we explore two promising scalable representation learning approaches on video domain. With hierarchical graph clusters built upon video-to-video similarities, we propose: 1) smart negative sampling strategy that significantly boosts training efficiency with triplet loss, and 2) a pseudo-classification approach using the clusters as pseudo-labels. The embeddings trained with the proposed methods are competitive on multiple video understanding tasks, including related video retrieval and video annotation. Both of these proposed methods are highly scalable, as verified by experiments on large-scale datasets.

1. Introduction

A tremendous amount of video data are uploaded on the web everyday. Such web video data has become one of the important elements of numerous online products. The goal of video representation learning is to compactly encode the semantic information in the videos to a lower dimensional space. The resulting embeddings are useful for video annotation, search, and recommendation problems, and thus is a core technology area for many online products. However, learning video representations is challenging due to the large data volume and their multi-modality, especially at million- or billion-scale. In addition, most of the web videos are unlabeled or inaccurately annotated, making the representation learning even more challenging.

One useful approach for video representation learning is to leverage the video-to-video relationships to learn an embedding function which projects videos onto a low-dimensional feature space. Given sparsely observed pairwise similarity scores, one can construct a *relational graph* whose nodes are videos and edges represent the relationship or similarity scores between video pairs. Such a graph contains rich information to learn embeddings that preserve

[*]This work was done during an internship at Google Research.

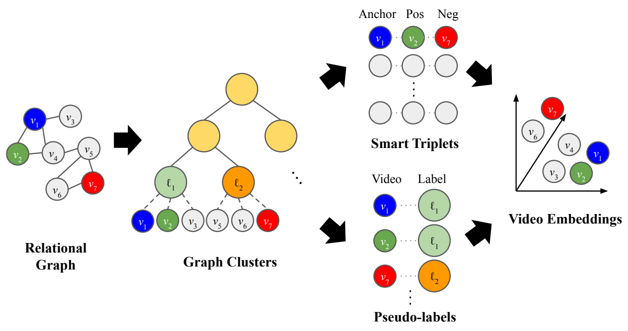

Figure 1. Overview of the proposed methods. Given a relational graph that defines the pairwise relationships, the videos are clustered based on the similarities using a hierarchical graph clustering. Then, the clusters and their hierarchy are used to construct informative training samples for two different learning approaches: *smart triplets* and classification with *pseudo-labels*.

the video-to-video similarity by putting the related videos closer to each other. The pairwise similarity of videos can be obtained from various sources, e.g., whether the videos appear in the same web page, without manual labeling.

Collaborative Deep Metric Learning (CDML) [25], for example, is a recently developed deep metric learning technique that learns such video embeddings from a relational graph. It embeds the content (audio-visual) features into a new feature space preserving the video-to-video relationships defined in the relational graph, proved to be useful for video annotation and recommendation, especially for cold-start cases.

However, CDML [25] has some limitations. Inspired by [37], the model uses triplet loss, which encourages the anchor-positive distances to be smaller than anchor-negative distances. They sample the anchor-positive pairs from related videos defined in the graph, and randomly sample negatives among all videos. Most of the randomly sampled negatives are too far from the anchor and thus are not informative for training after initial stage. As a result, semi-hard negative mining [37] is required for good performance, where the negatives are re-sampled among the videos within each mini-batch such that each negative is farther than the positive but not too far from the corresponding anchor. Even with the semi-hard negative mining, however, training is inefficient as the negative candidates in each mini-batch are random samples, regardless of their relationships with anchor-positive pairs. This requires models to be trained with a large batch size in order to ensure learning from informative triplets, which makes it difficult to train larger models with limited memory, especially with accelerated hardwares like GPU and TPU.

In this work, we explore a couple of promising approaches for scalable video metric learning, employing a relational graph to learn representations that preserves relationships between videos (Figure 1). Starting from a relational graph, where each node is a video and each edge represents the similarity between two videos, we first learn the overall structure of video-to-video relationships via hierarchical clustering. The first approach (*smart triplets*) is to generate training triplets similar to CDML [25] in a more sophisticated way, guaranteeing negatives with proper difficulty level to the corresponding anchor-positive pairs. This enables controlling difficulty level of triplets, and thus significantly reduces training inefficiency of previous work. The second approach (*pseudo-labels*) is to train a classification model with the cluster membership as the target pseudo-label, which can be scaled easily with recent advances in training framework for classification models (e.g., [13]). We emphasize that the proposed methods are applicable to any representation learning scenario where the pairwise relationships or similarity scores are defined.

The major contributions of this work are as follows:

1. We propose novel methods to learn video representation from video-to-video similarities defined in a relational graph. Along with hierarchical clustering on the relational graph, we propose a smart negative sampling strategy that significantly boosts training efficiency with triplet loss. Also, we propose a pseudo-classification approach that treats the clusters as pseudo-labels.
2. We verify that both of the proposed methods are highly scalable on large-scale datasets with hundreds of millions of videos.
3. The embeddings we train with the proposed methods are competitive on multiple video understanding tasks, outperforming state-of-the-art models on related video retrieval and video classification.

2. Related Work

Video Representation Learning. Various video-related tasks have been proposed for efficient video representation learning either with or without supervision. [21, 59] exploited supervised learning via video captioning. There have been plenty of un/self-supervised methods proposed, such as context-based self-supervised learning [31, 24, 18, 2], audio-visual cross-modal learning [41], and reconstruction/generation-based learning [46, 30].

Metric Learning. Metric learning embeds the examples into a space where similar or related ones are encoded to be closer to each other. Metric learning algorithms has been useful for various applications, such as face recognition [37, 53, 4], image retrieval [34, 54, 40, 51], fine-grained object recognition [58, 6, 52, 40], and related video retrieval [7, 20, 25]. There are distance-based [37, 34, 40, 45, 51, 5], classification-based [53, 28, 47], and clustering-based [33, 22, 44] approaches. In this work, we focus on distance-based and classification-based learning.

Distance Metric Learning. Distance metric learning learns embeddings using distance-based loss functions. It has become more popular in metric learning after FaceNet [37], where triplet loss was used for face recognition. The margin-based loss [54] and batch hard loss [9] are variants of triplet loss. Modified loss functions with distance metrics were proposed, including triangular angle [49] and histogram [45]. Recently, the distance-based loss functions have been further extended to utilize more possible pairs of examples within each of mini-batch than merely triplets, by considering quadruplets [5], *n*-tuples [40, 34, 51, 55], or all possible pairs within mini-batch [45].

Negative Mining. The importance of sampling scheme has been highlighted in metric learning. The idea of using more informative negatives was introduced in FaceNet [37], where they proposed the online semi-hard negative mining. Several following works proposed more negative mining and training loss schemes. Wu et al. provided theoretical analysis of different sampling strategies and proposed a distributed sampling scheme [54]. Huang et al. used an additional PDDM unit which tries to capture the local structure of different classes and dynamically mines samples based on the local similarity [12]. Yuan et al. ensembled cascaded models to combine different levels of hard triplets [56].

Recent work suggested to sample negatives using the class (label) information of the examples. Proxy-NCA [32] pre-defined a subset of "proxies" that presumably represents each class, and sampled the negatives from the different classes according to the proxies. Ge proposed to dynamically sample triplets from a hierarchical tree capturing the global structure of relationships between classes (labels) [8]. Most previous works [37, 40, 34, 51] utilize the class (label) information for better sampling, which is inapplicable when label is unavailable. Inspired by previous works, we adapt the negative "class" mining idea by applying hierarchical clustering on the relational graph and treat the "clusters" as labels. Wang et al. also utilized graph clustering to sample triplets [50], yet their approach focuses on extracting visual invariance among image patches.

Classification-based Metric Learning. Classification models have been introduced to metric learning [53, 28, 47] as an alternative of ranking losses. Qian et al. proposed to improve the softmax loss for more efficient metric learning [36], by maintaining multiple centers for each class.

Wang et al. [48] and Zhai et al. [57] applied metric learning with classification loss to face recognition and image classification, respectively. Iscen et al. utilized label propagation method to generate pseudo labels [14]. Again, most previous works required discrete labels associated with the examples, while our proposed pseudo-classification approach does not require any labeled data other than the relational graph. This is an application of pseudo-labeling, which has been widely used for semi-supervised learning [23, 38], where the network predictions were used as pseudo-labels.

3. Methods

In this section, we first formally define the video metric learning problem and describe the preprocessing step - graph clustering on the relational graph based on pairwise video similarities. Then, we present our proposed methods: 1) smart negative sampling for ranking losses (*smart triplet*), and 2) classification based on pseudo-labels from graph clustering (*pseudo-classification*).

3.1. Problem Setting

Metric learning, in general, is a task of learning a function $f(\mathbf{x}_1, \mathbf{x}_2)$ that returns a similarity (or distance) between the two input examples. As we focus on video metric learning, each $\mathbf{x}$ is a representation of a video, which can be raw pixels ($\mathbf{x} \in \mathbb{R}^{m \times n \times f}$, where each frame is $m \times n$ dimensional and the video contains f frames), or some compact representation of audio-visual features from a pre-trained video model ($\mathbf{x} \in \mathbb{R}^{d}$, where d is the dimensionality of the feature). We assume that this feature representation $\mathbf{x}$ is available for all videos.

We are also given as a source for training a *relational graph* $G = (V, E)$, where each node $\mathbf{x} \in V$ is a video and each edge $(\mathbf{x}_1, \mathbf{x}_2) \in E$ represents how similar the videos $\mathbf{x}_1$ and $\mathbf{x}_2$ are. The edge weights can be either binary or real numbers. Usually, the edges are sparsely observed, as it is impractical to log or compute similarity between all possible pairs of videos on the web. Note that the similarity measure in this relational graph can be arbitrary. It may be collected from human raters. We can also use implicit feedback from users; for instance, how frequently they are co-watched, co-searched, co-clicked, belong to same channels, share texts in the title, and so on. Regardless of the source, the proposed method can be applied as long as they are represented as the form described above.

Given these, our task is learning an embedding $\mathbf{z} \in \mathbb{R}^{k}$, typically where $k \ll d$, such that $\mathbf{z}_1^\top \mathbf{z}_2 \approx (\mathbf{x}_1, \mathbf{x}_2) \in E$. In other words, we would like to locate video embeddings close to each other if they are similar (as defined in the relational graph), and far away otherwise.

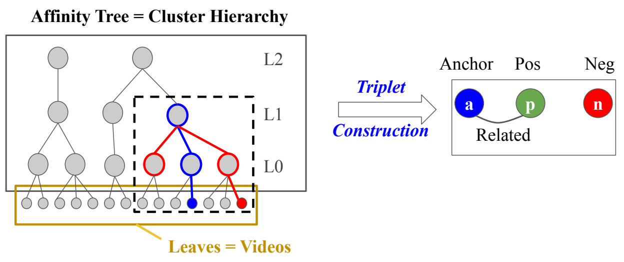

Figure 2. Smart triplets construction. Left panel represents an output tree of affinity clustering on a relational graph. The leaves represent the videos and the upper level nodes correspond to the video clusters found at each iteration of clustering, i.e., L0 nodes from the first iteration. The colored nodes and lines represent the negative sampling procedure using the cluster hierarchy. Here, we illustrate how to sample negatives from the L1 clusters. An anchor (blue leaf node) is randomly sampled, and a positive is chosen based on the original relational graph. A negative (red leaf) is sampled from the leaves that share the common parent at the desired level (the rightmost node at L1, outlined with blue color). The box with dashed lines represents the subtree whose leaves are the pool for negative sampling. The right panel shows the final triplet constructed.

3.2. Graph Clustering

We preprocess the relational graph using a clustering algorithm, which enables an efficient data sampling. We note that one can use any clustering algorithm that is applicable to a graph. We use affinity clustering [3], a hierarchical graph clustering tool that clusters the nearest neighbors using minimum spanning tree (MST) approach. Although other types of clustering are also applicable, we choose "hierarchical" clustering as we can control difficulty of triplets to generate with multiple levels. Using the edge scores, the algorithm clusters each node together with its closest neighbors in bottom-up fashion. The output is a collection of trees, each of which represents a cluster. It iterates the clustering process for several steps, and outputs an affinity tree whose levels correspond to the hierarchy of the clusters. Affinity clustering is fast, simple, and scalable, and thus is useful for processing a large-scale relational graph.

The left panel in Figure 2 shows an example output of affinity clustering on a relational graph, with 3 iterations. The leaves represent videos and intermediate nodes at each level correspond to the clusters of the lower-level nodes. The clusters of the lowest level contain most similar videos. The higher the level is, the less similar videos within the cluster are. Sampling videos from the higher-level clusters decreases the average similarities between the sampled pairs.

3.3. Graph Clustering Metric Learning (GCML)

Smart Negative Sampling. Once we have the clustering results, the triplets are sampled using the relational graph and the clusters. For each triplet, the *anchor* is randomly selected among all videos, and the *positive* is chosen among neighbors of the anchor on the relational graph. Instead of random sampling as in [25], we sample the *negative* using the cluster hierarchy in the affinity tree. For each anchor, we choose its sibling clusters (that share the same parent) at a desired level and sample the negatives from those clusters, as illustrated in Figure 2. This ensures that the sampled negatives are not too far from the anchor, and thus more informative for model training. As described in the previous section, we can choose the clusters at different levels in order to adjust the difficulty level of the sampled negatives. As an extreme, if we sample negatives from the lowest-level clusters and the cluster size is small, the sampled negatives will most likely be as similar as the sampled positives and thus are hardest negatives.

Training with Triplets. We follow the standard procedure for triplet loss optimization [25, 37]. The goal is to put a pair of relevant videos closer to each other in the embedding space while keeping the less related pairs farther. The embedding network optimizes the triplet loss, such that the distance between *anchor* and *positive* is smaller than the distance between *anchor* and *negative*:

$$\min \sum_{i=1}^{N} \left[\|f(\mathbf{x}_i^a) - f(\mathbf{x}_i^p)\|^2 - \|f(\mathbf{x}_i^a) - f(\mathbf{x}_i^n)\|^2 + \alpha \right]_+$$

where $\mathbf{x}_i^a$, $\mathbf{x}_i^p$, and $\mathbf{x}_i^n$ are the input feature vectors of anchor, positive, and negative of the i-th training triplet among N, respectively, f is the embedding network, and α is margin parameter for a hinge loss.

Similar to [25, 37], we use online semi-hard negative mining throughout the training. At each iteration, the negatives are re-sampled among the videos within each mini-batch as the closest ones that are farther than the positives from the corresponding anchors. If the initially sampled smart negatives are useful, they will be consistently chosen as the semi-hard negatives during training.

3.4. Cluster Labels Classification (CLC)

An alternative way to utilize the graph clusters is to treat them as labels for classification. We propose to use the cluster to which each video belongs as its label and train a classification model, e.g., cross entropy loss with softmax function. In this case, the top layer output of the embedding network will be fed into an n-way softmax classifier that will compute a distribution over the n cluster IDs. During training, the network is tuned to predict the cluster ID where the video belongs to. At test time, the classification layer is discarded and only the embedding towers are used.

Classifying large number of classes is expensive and time consuming, as we could have millions of clusters from the graph. Instead of computing the full softmax classification, we use sampled softmax [15] to reduce computational

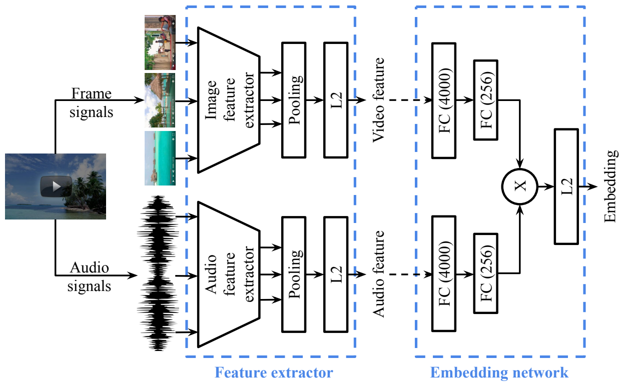

Figure 3. Network architecture of the embedding network.

costs. Specifically, a subset of classes are sampled in each step to compute the softmax loss during training.

The benefit of classification is that we do not need to sample hard negative examples within the mini-batch, which is replaced by sampling negative cluster labels. This removes the dependency on batch size as in triplet loss and it is relatively easier to scale to large number of negative sampled classes. On the other hand, the performance of the cluster classification models is highly dependent on the clustering quality. If the cluster size is too large, the model may not learn fine-grained distinction between samples; and if the cluster size is too small, the classifier may easily overfit to a few examples in the cluster, and thus can not learn useful embeddings.

4. Experiments

We conduct extensive experiments on real-world large-scale datasets to evaluate our methods on two video understanding tasks: *related video retrieval* and *video annotation*.

4.1. Network Architecture

Audio-visual Features. End-to-end video representation learning from raw signals is computationally prohibitive. To efficiently train a model on hundreds of millions of videos, we use audio-visual features extracted using pre-trained models, similarly to [25], illustrated in the left (Feature extractor) part of Figure 3.

To extract visual features, we first sub-sample the video frames at a rate of 1 frame per second. We then extract the ReLU outputs of the last hidden layer of Inception-v2 network [42], pre-trained on the JFT dataset [11], followed by PCA and whitening for dimension reduction into 1500. We apply average pooling over the processed features of all frames to generate the video-level features.

For audio features, we use an acoustic model with a modified version of ResNet-50 [10]. The audio signals are segmented into non-overlapping 960 ms window, and then decomposed with a short-time Fourier transform with 25 ms windows for every 10 ms. This results in a spectrogram integrated into 64 mel-spaced frequency bins. We extract features by fetching 100 of such segments into the ResNet, and aggregate them into video-level features by average pooling.

Embedding Network. Following [25], our embedding networks consist of two fully-connected hidden layers with dimension of 4000 and 256. We use two separate towers, each of which takes visual and audio features, respectively. The outputs of the two towers are aggregated by element-wise multiplications, followed by L_2 normalization. The network architecture is shown in Figure 3. The embedding network is trained to optimize triplet loss (Section 3.3) or cross entropy loss (Section 3.4).

4.2. Experimental Settings

We compare our two proposed models, Graph Clustering Metric Learning (*GCML*; Section 3.3) and Cluster Labels Classification (*CLC*; Section 3.4), against the state-of-the-art video metric learning method CDML [25]. CDML is equivalent to our first approach, but randomly choosing the negatives instead of the smart triplet generation.

Training Dataset. We construct the relational graph of videos from user signals; specifically, an edge in the graph exists if a pair of videos are frequently co-watched by multiple users. The videos are randomly split into training and test partition with a 7:3 ratio. For the triplet loss models (CDML and GCML), we sample up to 350M publicly available videos and generate up to 460M triplets from the 70% training partition of them.

We apply 3 iterations of affinity clustering on the relational graph (See Figure 2). We use L0 clusters to sample triplets for GCML models, except for triplet difficulty level experiments (Table 2). For the CLC model, we use the 14M L0 clusters, each selecting up to 50 videos to avoid class imbalance, resulting in about 405M videos in total.

Hyperparameters. The CDML and GCML models are trained using semi-hard negative mining [37, 25] within a mini-batch of 9600 triplets, where each negative is reassigned as the hardest among the ones that are as close as or farther than the positive to the anchor. We apply triplet loss margin of 0.5. The initial learning rate is set to 0.1, with decay by 0.98 for every 300,000 steps throughout 1M training steps, using asynchronous-SGD with 150 worker replicas on CPUs with RMSProp optimizer. The CLC model is trained with initial learning rate of 300, learning rate decay of 0.5 for every 200,000 steps with batch size 512 on TPUv3 with 32 cores with SGD optimizer. At each iteration, 200,000 classes are sampled for classification.

4.3. Related Video Retrieval

We first evaluate the performance of models on related video retrieval, which is highly relevant to video-to-video recommendations. In this task, related videos are identified among candidates for a given query video q. The relatedness between the query and the candidates is defined by the relational graph.

Based on our embedding network f, we extract the embeddings for the query and candidate videos. For each query, we compute the similarity between the query video and each of the candidate videos, and rank them by the estimated similarity. The similarity between a pair is defined as the cosine similarity between the embeddings of the two:

$$\cos(f(\mathbf{v}_1), f(\mathbf{v}_2)) = f(\mathbf{v}_1)^\top f(\mathbf{v}_2) \tag{1}$$

where $\mathbf{v}_i$ represents the content features of the video i. Since we normalize the embeddings, the cosine similarity is equivalent to the dot product. When evaluating the retrieval performance, we use top k candidate videos with largest cosine similarity to the query.

4.3.1 Evaluation Protocol

For evaluation, we use videos that are not seen during the training for queries and candidates. As proposed in [25], this setting simulates cold-start video recommendation, where the queries and retrievals are drawn from fresh videos. Note that this setting provides fair comparison scheme that guarantees the test videos are unseen by all models trained on differently sampled videos.

The evaluation set is composed of 1.3M query videos and the database set is constructed as the union of the related videos in the query set. Each query video has on average 11 related videos in the database set, which has about 15M unique videos in total.

Evaluation Metrics. The retrieval performance is evaluated using two different ranking metrics:

1) Normalized Discounted Cumulative Gain (NDCG) measures the ranking quality using the relevancy of each retrieved item and its order in the list. Discounted cumulative gain for top-k retrieved items (DCG@k) is

$$DCG@k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{log_2(i+1)}, \tag{2}$$

where i is the position of each item in the list and $rel_i \in \{0, 1\}$ is the binary indicator of the relevancy of the i-th item to the query. NDCG is computed as DCG normalized by the maximum possible DCG through position k, which happens when the list of top-k retrieved items matches the correct order of relevancy to the query. We use $k = 60$ in our experiments.

2) Mean Average Precision (MAP) measures the accuracy of ranking by computing the area under the precision-recall curve:

$$MAP = \frac{1}{|Q|} \sum_{q \in Q} \int_0^1 P_q(r)dr \approx \frac{1}{|Q|} \sum_{q \in Q} \sum_{i=1}^{k} P_q(i)\Delta r(i)$$

Method	Training set	MAP	NDCG@60
CDML [25]	150M	2.85%	6.05%
	458M	3.34%	7.07%
GCML (ours)	45M	3.57%	7.19%
	420M	**3.74%**	**7.66%**
CLC (ours)	405M	3.41%	6.99%

Table 1. Performance on related video retrieval task

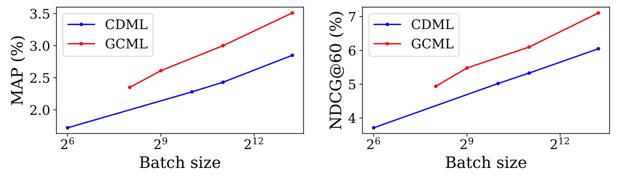

Figure 4. Related video retrieval performance with different batch sizes.

where $P_q(r)$ is precision for the query q as a function of recall r. This integral is approximated to be a finite sum over all possible positions in the list of 60 retrieved videos.

Both NDCG and MAP measure the relevance of ranked items to the query, but NDCG puts more emphasis on the higher ranks by exponentially decaying the weights, while MAP does quadratically.

4.3.2 Results and Discussion

We first compare the performance of GCML and CDML models. Table 1 compares MAP and NDCG@60 scores for different models. Using the same batch size of 9600 and the similar number of training triplets (400M+), GCML models outperform CDML models. We note that the GCML model trained with less triplets (45M) outperform the CDML baseline trained with more triplets (150M–458M), producing up to 25% MAP and 18% NDCG relative improvement. This concludes that the smart negatives are more informative and efficient than the random negatives. In addition, using more training triplets does not benefit GCML model (6.5%/7.9% increase in MAP/NDCG from 45M to 420M) as much as CDML (17.2%/16.8% increase in MAP/NDCG from 150M to 458M), indicating that GCML models are already efficiently utilizing much information even with small training data. Our second model, CLC is competitive to CDML models trained on similar size of training set, while not being dependent on the training batch size.

Batch size. As described in Section 1, the online semi-hard negative mining inherently causes the dependence of performance on the number of negative candidates within the mini-batch. We thus compare the performance of our GCML models trained with various batch sizes against CDML (Figure 4). We observe that both GCML and CDML yield better performance when trained with larger batch size. Yet, GCML consistently outperforms CDML for all batch sizes.

Level of Difficulty	MAP	NDCG@60
GCML (Easy; 45M)	3.37%	6.64%
GCML (Medium; 45M)	3.43%	6.91%
GCML (Hard; 45M)	**3.57%**	**7.19%**
CDML [25] (150M)	2.85%	6.05%

Table 2. Related video retrieval performance with different difficulty levels of sampled smart negatives.

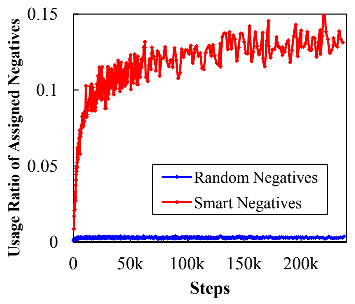

Figure 5. Usage ratio of the assigned negatives. Our smart negatives are used more frequently compared to random negatives.

Difficulty of Negatives. To understand how the cluster quality affects the performance, we train and evaluate the GCML models trained with triplets sampled from different difficulty levels. For this experiment, we use lighter setting (batch size of 2048 and training on smaller training set in Table 1) on TPU for speed. The *easy*, *medium*, and *hard* levels correspond to the smart negatives sampled from L2, L1, and L0 clusters (see Figure 2), respectively. As the later round of affinity clustering yields larger clusters, the negatives sampled from *easier* clusters are less similar to the anchors than the negatives sampled from *harder* ones.

As shown in Table 2, the *harder* the triplets are, the better the MAP and NDCG scores are. We also note that, regardless of the difficulty level of training triplets, all the GCML models outperform the CDML baseline.

4.3.3 Analysis of Negative Sampling

We further analyze how the proposed smart negative sampling benefits the video metric learning.

Usage Ratio of Sampled Negatives. To understand how the sampled negatives are used, we measure the ratio of those assigned negatives actually being used for training. As the models are trained using online semi-hard negative mining, a new semi-hard negative is chosen among all the candidates within the mini-batch for each anchor-positive pair at every step. The semi-hard negative mining, by definition, chooses the hardest among the negatives that are farther than the positive from the anchor. Therefore, the more the negatives originally assigned to the anchor-positive pairs are chosen as semi-hard, the more useful those assigned negatives are for learning.

Figure 5 shows the ratio of the assigned negatives being

Method	Margin	
	No Online Mining (Assigned Neg.)	w/ Online Mining (Semi-hard Neg.)
CDML [25]	0.62 $\sim$ 0.76	0.01 $\sim$ 0.02
GCML	-0.08 $\sim$ 0.05	0.015 $\sim$ 0.02

Table 3. Margin of assigned (random or smart) negatives vs. semi-hard negatives. Each range corresponds to +/- one standard deviation from the mean.

chosen as semi-hard throughout the training. We observe that the usage ratio is much larger with GCML than with CDML, suggesting the effectiveness of our cluster-based negative sampling. The usage ratio for CDML is close to $1/batch\,size$, meaning that randomly assigned initial negatives are rarely used throughout training.

We also note that the usage ratio of the smart negatives increases as training proceeds. This indicates that the model starts learning from easier negatives at earlier stage, and gradually moves towards harder negatives and thus uses the assigned smart negatives more.

Margin Analysis. The effectiveness of smart negatives can also be found in the analysis of the triplet relationships. We quantify the difficulty level of a negative for each anchor-positive pair as the *margin*:

$$margin = \|f(\mathbf{x}^a) - f(\mathbf{x}^n)\|^2 - \|f(\mathbf{x}^a) - f(\mathbf{x}^p)\|^2 \quad (3)$$

where a, p, n represent anchor, positive, and negative, respectively. The *margin* measures how farther the anchor is from the negative compared from the positive. Closer negatives from the anchor are generally harder to distinguish, and thus allow the model to learn more minuscule partitions. We compare the margin values for the negatives originally assigned for the anchor-positive pairs and the negatives actually chosen as semi-hard. If the margin for the assigned negative is within the range of the margin for semi-hard negatives, it suggests that the assigned negatives are at the right level of difficulty and thus are useful candidates even if they were not actually chosen.

Table 3 summarizes the margin values of the assigned and semi-hard negatives for GCML and CDML. We observe that the margin values of smart negatives are within the range of semi-hard negatives. On the other hand, most of the random negatives of CDML are far from the semi-hard negatives, indicating that the random negatives are too easy and thus not being chosen as semi-hard. This result is consistent with the findings in usage ratio analysis.

Discussion. The two analyses above as a whole suggest that the smart negatives allow an efficient learning via providing more informative triplets. We find that the smart negatives are the right candidates and actually chosen as semi-hard throughout the training. Combined with the better performance of the GCML, we conclude that using informative training samples is important for learning good representations.

Samples per cluster	MAP	NDCG@60
20	3.42%	**6.86%**
50	**3.43%**	6.80%
100	3.41%	6.83%
1,000	3.24%	6.58%

Table 4. Different number of sampled videos per cluster in CLC.

Interestingly, we observe that GCML does not perform well without online semi-hard negative mining, even with the more informative triplets. One possible explanation is that it is also important to learn from easier negatives at the early stage of training. Figure 5 suggests that the smart negatives are too hard for a model at the beginning, and chosen only after it has been trained for few iterations. This observation points to a possible improvement via curriculum learning. We can mimic the online semi-hard negative mining by training the models with different difficulty levels of triplets, i.e., starting from easy to hard ones. If we can adjust the right levels of triplets, we will be able to train better models even without online mining.

4.3.4 Sampling in Cluster Classification

We limit the maximum number of sampled videos per cluster. As the clusters obtained from affinity clustering may be highly imbalanced in size, the number of samples per cluster is an important hyper-parameter to tune. If this is too large, the training set would bias towards larger clusters as they contain more examples. This could potentially hurt the embedding quality as it focuses on coarse-grained relationship between videos during training. If too few videos are sampled per cluster, the training set size would become very small and may fail to learn good representations.

We train CLC models on L1 cluster labels with different number of sampled videos per cluster (Table 4). Sampling more videos per cluster does not translate to better retrieval performance, despite the increase of the total number of videos in the training set. When the number of videos per cluster is too large (e.g. 1,000), the retrieval performance drops significantly, suggesting the model cannot learn fine-grained similarities of videos from the large clusters.

We also explore how the number of sampled classes affects the embedding quality, as shown in Table 5. Sampling more classes would produce better approximation of the original softmax classification, at a higher computational cost during training. We found that the retrieval performance saturates at about 200,000 sampled classes.

4.4. Video Annotation

The goal of video annotation task is to predict one or more labels for a given video. We evaluate how useful the proposed embeddings are on two public video classification datasets: YouTube-8M [1] and Sports-1M [17].

Sampled classes	MAP	NDCG@60
10,000	3.21%	6.53%
50,000	3.31%	6.60%
200,000	**3.42%**	**6.86%**
400,000	**3.42%**	6.84%

Table 5. Different number of sampled class for training cluster labeling model. Sampling 200,000 labels is enough for good performance.

Methods	Embedding only		w/ Audio-visual	
	GAP	MAP	GAP	MAP
2018 Challenge (YouTube-8M features)				
KANU [19]	–	–	88.5%	58.3%
YT8M-T [43]	–	–	88.7%	58.8%
PhoneixLin [27]	–	–	88.7%	59.7%
Samsung AI [35]	–	–	88.7%	58.4%
Next top GB [39]	–	–	89.0%	59.6%
Audio-visual features (Section 4.1)				
Audio-visual only	–	–	89.0%	55.5%
CDML [25] (150M)	86.6%	51.7%	89.0%	55.7%
GCML (45M)	86.9%	52.5%	89.1%	56.0%
CDML [25] (458M)	**88.4%**	**55.5%**	**89.4%**	57.0%
GCML (420M)	88.3%	**55.5%**	**89.4%**	**57.1%**
CLC	87.9%	54.6%	89.2%	56.5%

Table 6. Classification performance on YouTube-8M. The numbers in parenthesis are the number of triplets used for training.

Annotation Model. We train a video classification model on top of the embeddings, and evaluate the performance of the annotation model on a held-out set. The annotation model consists of a 4096-dimensional fully connected layer followed by a muti-label classifier. We use a Mixture of Experts (MoE) model [16] with 5 mixtures as the classifier. The model is trained with ADAM optimizer with initial learning rate of 0.005 and batch size 512. We use a half-period cosine learning rate decay schedule [29] to decrease the learning rate gradually to 0 at 100,000 steps.

We train the annotation models with two different input types: the embeddings from the proposed models (e.g., GCML) with and without the audio-visual features extracted as described in 4.1.

YouTube-8M. YouTube-8M [1] provides a video classification dataset of 6.1M+ YouTube videos and video-level labels from 3,862 knowledge graph entities. In this experiment, we train the annotation model on the full YouTube-8M (ver. 2018) training set and evaluate the classification performance on the full validation set. We follow the standard evaluation protocol and report the Global Average Precision (GAP) and Mean Average Precision (MAP).

Table 6 shows the GAP and MAP scores on YouTube-8M classification task. When training on small datasets (45–150M), GCML outperforms CDML consistently on both GAP and MAP, similarly to the related video retrieval results. When training on large (400M+) datasets, however, GCML and CDML perform on par. When used in conjunction with audio-visual features, both CDML and GCML improve over the audio-visual-only baseline.

Methods	Embedding only		w/ Audiovisual	
	Hit@1	Hit@5	Hit@1	Hit@5
Audio-visual only	-	-	70.0%	86.3%
CDML [25] (150M)	67.0%	86.4%	70.8%	87.7%
GCML (45M)	68.7%	87.5%	71.4%	88.3%
CDML [25] (458M)	71.0%	89.1%	72.2%	88.7%
GCML (420M)	**71.7%**	**89.5%**	**72.6%**	**89.2%**
CLC	71.3%	89.4%	72.2%	88.9%

Table 7. Classification performance on Sports-1M. The numbers in parenthesis are the number of triplets used for training.

We also compare the proposed methods with the top performers in the 2nd YouTube-8M challenge [26]. The proposed methods yield better results when concatenated with the audio-visual features. The GCML and CDML models yield the highest GAP scores among all including the top performers in [26]. The proposed methods show relatively lower MAP scores compared to the top performers in [26], probably because training not only on the video-level features but also on frame-level features is critical for the MAP scores. Note that our method is not directly comparable to the challenge results due to the difference in training data and features.

Sports-1M. Sports-1M dataset [17] consists of about 1M YouTube videos with 487 classes of sports. We train the annotation model on the full training set and evaluate on the validation set, and report Hit@1 and Hit@5 in this dataset as proposed in [17]. As shown in Table 7, our GCML features outperform CDML, both with and without audio-visual features.

5. Conclusion

In this work, we propose two novel methods that learn video representations from pairwise similarity structure via graph clustering: Graph Clustering Metric Learning (GCML), using ranking loss with smart triplet mining, and Cluster Labels Classification (CLC), modeling it as a pseudo-classification problem with cluster labels. We observe that both methods outperform the baselines in video retrieval and annotation tasks. With more detailed analysis, we observe that the proposed mining strategy allows the models to learn from more informative training examples.

As a future work, we may improve the performance by adjusting the difficulty level of smart negatives at the right level with more thorough parameter search for clustering. Curriculum learning may be another option for further improvement, such that the model can learn from easier examples first, then move toward harder examples. We could also investigate how the clustering algorithms and the number of clusters affect the representation learning quality in CLC.

References

[1] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. YouTube-8M: A large-scale video classification benchmark. *arXiv:1609.08675*, 2016. 7, 8

[2] Unaiza Ahsan, Rishi Madhok, and Irfan Essa. Video jigsaw: Unsupervised learning of spatiotemporal context for video action recognition. In *WACV*, 2019. 2

[3] MohammadHossein Bateni, Soheil Behnezhad, Mahsa Derakhshan, MohammadTaghi Hajiaghayi, Raimondas Kiveris, Silvio Lattanzi, and Vahab Mirrokni. Affinity clustering: Hierarchical clustering at scale. In *Advances in Neural Information Processing Systems (NIPS)*, 2017. 3

[4] Qiong Cao, Yiming Ying, and Peng Li. Similarity metric learning for face recognition. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2013. 2

[5] Weihua Chen, Xiaotang Chen, Jianguo Zhang, and Kaiqi Huang. Beyond triplet loss: a deep quadruplet network for person re-identification. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[6] Yin Cui, Feng Zhou, Yuanqing Lin, and Serge Belongie. Fine-grained categorization and dataset bootstrapping using deep metric learning with humans in the loop. In *Proc. of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2016. 2

[7] Yun Fu, Zhu Li, Thomas S Huang, and Aggelos K Katsaggelos. Locally adaptive subspace and similarity metric learning for visual data clustering and retrieval. *Computer Vision and Image Understanding*, 110(3):390–402, 2008. 2

[8] Weifeng Ge. Deep metric learning with hierarchical triplet loss. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2018. 2

[9] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *arXiv:1703.07737*, 2017. 2

[10] Shawn Hershey, Sourish Chaudhuri, Daniel PW Ellis, Jort F Gemmeke, Aren Jansen, R Channing Moore, Manoj Plakal, Devin Platt, Rif A Saurous, Bryan Seybold, et al. CNN architectures for large-scale audio classification. In *Proc. of the IEEE international conference on acoustics, speech and signal processing (ICASSP)*, 2017. 4

[11] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv:1503.02531*, 2015. 4

[12] Chen Huang, Chen Change Loy, and Xiaoou Tang. Local similarity-aware deep feature embedding. In *Advances in neural information processing systems (NIPS)*, 2016. 2

[13] Seong Jae Hwang, Joonseok Lee, Balakrishnan Varadarajan, Ariel Gordon, Zheng Xu, and Apostol Paul Natsev. Large-scale training framework for video annotation. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019. 2

[14] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. Label propagation for deep semi-supervised learning. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3

[15] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. On using very large target vocabulary for neural machine translation. *arXiv:1412.2007*, 2014. 4

[16] Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994. 8

[17] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proc. of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 7, 8

[18] Dahun Kim, Donghyeon Cho, and In So Kweon. Self-supervised video representation learning with space-time cubic puzzles. In *Pro. of the AAAI Conference on Artificial Intelligence*, 2019. 2

[19] Eun-Sol Kim, Kyoung-Woon On, Jongseok Kim, Yu-Jung Heo, Seong-Ho Choi, Hyun-Dong Lee, and Byoung-Tak Zhang. Temporal attention mechanism with conditional inference for large-scale multi-label video classification. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2018. 8

[20] Giorgos Kordopatis-Zilos, Symeon Papadopoulos, Ioannis Patras, and Yiannis Kompatsiaris. Near-duplicate video retrieval with deep metric learning. In *Proc. of the IEEE International Conference on Computer Vision (CVPR)*, 2017. 2

[21] Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. Dense-captioning events in videos. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 2

[22] Marc T Law, Raquel Urtasun, and Richard S Zemel. Deep spectral clustering learning. In *Proc. of the International Conference on Machine Learning (ICML)*, 2017. 2

[23] Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *ICML Workshop on Challenges in Representation Learning*, volume 3, page 2, 2013. 3

[24] Hsin-Ying Lee, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Unsupervised representation learning by sorting sequences. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 2

[25] Joonseok Lee, Sami Abu-El-Haija, Balakrishnan Varadarajan, and Apostol Paul Natsev. Collaborative deep metric learning for video understanding. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018. 1, 2, 4, 5, 6, 7, 8

[26] Joonseok Lee, Walter Reade, Rahul Sukthankar, George Toderici, et al. The 2nd Youtube-8M large-scale video understanding challenge. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2018. 8

[27] Rongcheng Lin, Jing Xiao, and Jianping Fan. NextVLAD: An efficient neural network to aggregate frame-level features for large-scale video classification. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2018. 8

[28] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *Proc. of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2017. 2

[29] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv:1608.03983*, 2016. 8

[30] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv:1511.05440*, 2015. 2

[31] Ishan Misra, C Lawrence Zitnick, and Martial Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2016. 2

[32] Yair Movshovitz-Attias, Alexander Toshev, Thomas K Leung, Sergey Ioffe, and Saurabh Singh. No fuss distance metric learning using proxies. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 2

[33] Hyun Oh Song, Stefanie Jegelka, Vivek Rathod, and Kevin Murphy. Deep metric learning via facility location. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[34] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[35] Pavel Ostyakov, Elizaveta Logacheva, Roman Suvorov, Vladimir Aliev, Gleb Sterkin, Oleg Khomenko, and Sergey I Nikolenko. Label denoising with large ensembles of heterogeneous neural networks. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2018. 8

[36] Qi Qian, Lei Shang, Baigui Sun, Juhua Hu, Hao Li, and Rong Jin. Softtriple loss: Deep metric learning without triplet sampling. *arXiv:1909.05235*, 2019. 2

[37] Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A unified embedding for face recognition and clustering. In *Proc. of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2015. 1, 2, 4, 5

[38] Weiwei Shi, Yihong Gong, Chris Ding, Zhiheng MaXiaoyu Tao, and Nanning Zheng. Transductive semi-supervised deep learning using min-max features. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2018. 3

[39] Miha Skalic and David Austin. Building a size constrained predictive model for video classification. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2018. 8

[40] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems (NIPS)*, 2016. 2

[41] Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. Videobert: A joint model for video and language representation learning. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2019. 2

[42] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 4

[43] Yongyi Tang, Xing Zhang, Lin Ma, Jingwen Wang, Shaoxiang Chen, and Yu-Gang Jiang. Non-local NetVLAD encoding for video classification. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2018. 8

[44] Makarand Tapaswi, Marc T Law, and Sanja Fidler. Video face clustering with unknown number of clusters. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2019. 2

[45] Evgeniya Ustinova and Victor Lempitsky. Learning deep embeddings with histogram loss. In *Advances in Neural Information Processing Systems*, 2016. 2

[46] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *Advances in Neural Information Processing Systems (NIPS)*, 2016. 2

[47] Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Liu. Additive margin softmax for face verification. *IEEE Signal Processing Letters*, 25(7):926–930, 2018. 2

[48] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 3

[49] Jian Wang, Feng Zhou, Shilei Wen, Xiao Liu, and Yuanqing Lin. Deep metric learning with angular loss. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 2

[50] Xiaolong Wang, Kaiming He, and Abhinav Gupta. Transitive invariance for self-supervised visual representation learning. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 2

[51] Xinshao Wang, Yang Hua, Elyor Kodirov, Guosheng Hu, Romain Garnier, and Neil M Robertson. Ranked list loss for deep metric learning. *arXiv:1903.03238*, 2019. 2

[52] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009. 2

[53] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In *Proc. of the European conference on computer vision (ECCV)*, 2016. 2

[54] Chao-Yuan Wu, R Manmatha, Alexander J Smola, and Philipp Krahenbuhl. Sampling matters in deep embedding learning. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 2

[55] Baosheng Yu and Dacheng Tao. Deep metric learning with tuplet margin loss. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2019. 2

[56] Yuhui Yuan, Kuiyuan Yang, and Chao Zhang. Hard-aware deeply cascaded embedding. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 2

[57] Andrew Zhai, Hao-Yu Wu, and US San Francisco. Classification is a strong baseline for deep metric learning. In *Proc. of the British Machine Vision Conference (BMVC)*, 2019. 3

[58] Xiaofan Zhang, Feng Zhou, Yuanqing Lin, and Shaoting Zhang. Embedding label structures for fine-grained feature representation. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[59] Luowei Zhou, Yingbo Zhou, Jason J Corso, Richard Socher, and Caiming Xiong. End-to-end dense video captioning with masked transformer. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2