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.
ships 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 an adversarial training framework for classification models, which 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].

**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 $f(x_1, x_2)$ that returns a similarity (or distance) between the two input examples. As we focus on video metric learning, each $x$ is a representation of a video, which can be raw pixels ($x \in \mathbb{R}^{m \times n \times f}$, 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 ($x \in \mathbb{R}^d$, where $d$ is the dimensionality of the feature). We assume that this feature representation $x$ is available for all videos.

We are also given as a source for training a relational graph $G = (V, E)$, where each node $x \in V$ is a video and each edge $(x_1, x_2) \in E$ represents how similar the videos $x_1$ and $x_2$ 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 $z \in \mathbb{R}^k$, typically where $k \ll d$, such that $z_1^T z_2 \approx (x_1, x_2) \in E$. 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.

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

![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.](image)
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:

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

where $x_i^a$, $x_i^p$, and $x_i^n$ 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 $\alpha$ 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
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 over-fit 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 TPUsv3 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:

$$\cos(f(v_1), f(v_2)) = f(v_1)^T f(v_2)$$

where $v_i$ 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

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

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 comparing the area under the precision-recall curve:

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

### 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.

### 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

![Table 1. Performance on related video retrieval task](image1.png)

<table>
<thead>
<tr>
<th>Method</th>
<th>Training set</th>
<th>MAP</th>
<th>NDCG@60</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDML [25]</td>
<td>150M</td>
<td>2.85%</td>
<td>6.05%</td>
</tr>
<tr>
<td></td>
<td>458M</td>
<td>3.34%</td>
<td>7.07%</td>
</tr>
<tr>
<td>GCML (ours)</td>
<td>45M</td>
<td>3.57%</td>
<td>7.19%</td>
</tr>
<tr>
<td></td>
<td>420M</td>
<td>3.74%</td>
<td>7.66%</td>
</tr>
<tr>
<td>CLC (ours)</td>
<td>405M</td>
<td>3.41%</td>
<td>6.99%</td>
</tr>
</tbody>
</table>

![Table 2. Related video retrieval performance with different difficulty levels of sampled smart negatives.](image2.png)

<table>
<thead>
<tr>
<th>Level of Difficulty</th>
<th>MAP</th>
<th>NDCG@60</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCML (Easy; 45M)</td>
<td>3.37%</td>
<td>6.64%</td>
</tr>
<tr>
<td>GCML (Medium; 45M)</td>
<td>3.43%</td>
<td>6.91%</td>
</tr>
<tr>
<td>GCML (Hard; 45M)</td>
<td>3.57%</td>
<td>7.19%</td>
</tr>
<tr>
<td>CDML [25] (150M)</td>
<td>2.85%</td>
<td>6.05%</td>
</tr>
</tbody>
</table>

![Figure 4. Related video retrieval performance with different batch sizes.](image3.png)

![Figure 5. Usage ratio of the assigned negatives. Our smart negatives are used more frequently compared to random negatives.](image4.png)
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:

$$
margin = \| f(x^a) - f(x^n) \|_2^2 - \| f(x^p) - f(x^n) \|_2^2
$$

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.

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

<table>
<thead>
<tr>
<th>Sampled classes</th>
<th>MAP</th>
<th>NDCG@60</th>
</tr>
</thead>
<tbody>
<tr>
<td>10,000</td>
<td>3.21%</td>
<td>6.53%</td>
</tr>
<tr>
<td>50,000</td>
<td>3.31%</td>
<td>6.60%</td>
</tr>
<tr>
<td>200,000</td>
<td>3.42%</td>
<td>6.86%</td>
</tr>
<tr>
<td>400,000</td>
<td>3.42%</td>
<td>6.84%</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Methods</th>
<th>Embedding only</th>
<th>w/ Audio-visual</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GAP</td>
<td>MAP</td>
</tr>
<tr>
<td>2018 Challenge (YouTube-8M features)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KANU [19]</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>YT8M-T [43]</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PhoneixLin [27]</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Samsung AI [35]</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Next top GB [39]</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Audio-visual features (Section 4.1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CDML [25] (150M)</td>
<td>86.6%</td>
<td>51.7%</td>
</tr>
<tr>
<td>GCML (45M)</td>
<td>86.9%</td>
<td>52.5%</td>
</tr>
<tr>
<td>CDML [25] (458M)</td>
<td>88.4%</td>
<td>55.5%</td>
</tr>
<tr>
<td>GCML (420M)</td>
<td>88.3%</td>
<td>55.5%</td>
</tr>
<tr>
<td>CLC</td>
<td>87.9%</td>
<td>54.6%</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Methods</th>
<th>Embedding only</th>
<th>w/ Audiovisual</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hit@1</td>
<td>Hit@5</td>
</tr>
<tr>
<td>Audio-visual only</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CDML [25] (150M)</td>
<td>67.0%</td>
<td>86.4%</td>
</tr>
<tr>
<td>GCML (45M)</td>
<td>68.7%</td>
<td>87.5%</td>
</tr>
<tr>
<td>CDML [25] (458M)</td>
<td>71.0%</td>
<td>89.1%</td>
</tr>
<tr>
<td>GCML (420M)</td>
<td>71.7%</td>
<td>89.5%</td>
</tr>
<tr>
<td>CLC</td>
<td>71.3%</td>
<td>89.4%</td>
</tr>
</tbody>
</table>

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 multi-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.

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


