Collaborative Deep Metric Learning for Video Understanding

Joonseok Lee
Google AI Perception
Mountain View, CA
joonseok@google.com

Sami Abu-El-Haija
Google AI Perception
Mountain View, CA
haija@google.com

Balakrishnan Varadarajan
Google AI Perception
Mountain View, CA
balakrishnanv@google.com

Apostol (Paul) Natsev
Google AI Perception
Mountain View, CA
natsev@google.com

ABSTRACT

The goal of video understanding is to develop algorithms that enable machines understand videos at the level of human experts. Researchers have tackled various domains including video classification, search, personalized recommendation, and more. However, there is a research gap in combining these domains in one unified learning framework. Towards that, we propose a deep network that embeds videos using their audio-visual content, onto a metric space which preserves video-to-video relationships. Then, we use the trained embedding network to tackle various domains including video classification and recommendation, showing significant improvements over state-of-the-art baselines. The proposed approach is highly scalable to deploy on large-scale video sharing platforms like YouTube.

CCS CONCEPTS

• **Computing methodologies** → **Visual content-based indexing and retrieval**; *Neural networks*; • **Information systems** → Recommender systems;

KEYWORDS

Video understanding, metric learning, triplet learning, recommendation, classification, collaborative filtering

ACM Reference Format:
Joonseok Lee, Sami Abu-El-Haija, Balakrishnan Varadarajan, and Apostol (Paul) Natsev. 2018. Collaborative Deep Metric Learning for Video Understanding. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3219819.3219856

1 INTRODUCTION

Recent advancements in computer vision enabled machines to surpass human accuracy in object recognition in still images [14], powered by large amounts of manually labeled data. We human beings, however, naturally understand our surroundings with less supervision through stereovision and watching moving objects. In order for machines to understand the world as we do, supervised image understanding may not suffice as still images lose information about actor intent and how objects interact. Video understanding, a rapidly developing area of research, aims at understanding videos at the level of a human expert and is essential to make drastic progress in visual recognition.

There are many applications in video understanding. One application is *video annotation*, where the goal is to identify key individual objects (e.g., car) or higher-level concepts (e.g., soccer game) given a video. As humans, we quickly recognize familiar objects, such as cars, children, or buildings. We are able to recognize further details such as the kind of the car, or identify the main actors. Usually the set of candidate labels are finite, and in this case video annotation problem can be considered as multi-label *video classification* problem, assigning an appropriate set of labels to the given video. Another application is *video recommendation*. In online video sharing platforms like YouTube, there is "Recommended" section, which is personalized based on the user profile, watch history, and site-wide behaviors such as searching or commenting. *Video search* is another application. Traditionally, video has been searched based only on the human-labeled tags or meta-data (e.g., titles or descriptions). With video understanding, we can retrieve videos that are relevant to the query purely from the content. *Related video retrieval* problem is a variant of video search or recommendation, where the query is a video and the response is a list of videos. For example, YouTube shows a set of related videos on the right side while you are watching a video. This recommendation may depend not only on the inferred user taste, but also on the video currently being watched. These examples are shown in Figure 1. In addition to these applications, there are many more applications where video content understanding can be applied, including event detection, video question answering (VQA), motion recognition, and video summarization.

Video understanding research, however, faces some challenges. First, video files are large and often prohibitive to download and store. Second, training machine learning models on videos is computationally intensive as the video files must be decoded during training. Third, collecting human labels on videos is costly as raters are required to watch the video, which can take minutes to hours per video. Fortunately, recent release of large-scale public dataset [1, 39] and specialized hardware [23] alleviate some of these challenges.

KDD '18, August 19–23, 2018, London, United Kingdom

ACM ISBN 978-1-4503-5552-0/18/08.
https://doi.org/10.1145/3219819.3219856

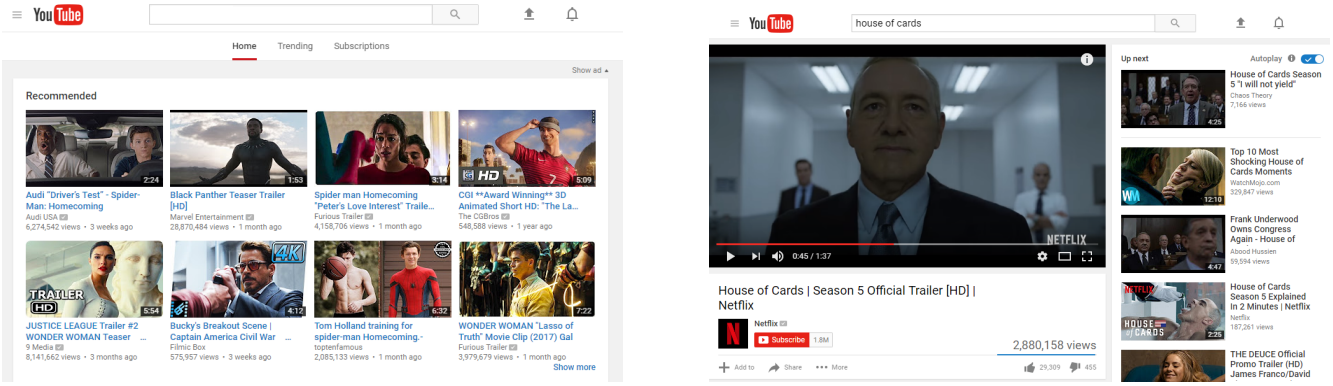


Figure 1: Examples of Video Understanding Problems. (*Left*): Video recommendations for a user who recently watched trailers of "Spider-Man: Homecoming (2017)" and "Wonder Woman (2017)". (*Right*): Related video retrieval problem, where the query is a trailer for "House of Cards Season 5 (2017)".

We propose to tackle these various video understanding problems by metric learning [4]. Concretely, we are given a graph of videos, where a pair of videos have an edge in the graph if they are related. Our goal is to learn an "embedding" function which can project a video onto a low-dimensional space, where related videos are close in the embedding space, while unrelated ones are far apart. Our goal is to learn an embedding function that can generalize across different video understanding tasks.

Our proposed method uses two sources of information: raw video content and user behavior. From the first source, we use state-of-the-art deep neural networks to extract image and audio features. From the second source, we construct a graph on videos where an edge exists between two videos if they are co-watched by many users. We train our embedding function to preserve the structure of this graph. This idea is conceptually aligned with Collaborative Filtering (CF) [15], where many users (implicitly) *collaborate* each other to *filter* relevant items. We wish to generalize beyond the training set, where a video or its edges are not observed during training.

We foresee many applications with our method, some of which are demonstrated in this paper. For instance, our method can be used for *cold-start* recommendation, where one wishes to infer user interests in newly uploaded videos. Our method can also be used for video annotation, which could assist search and discovery. Improving performance on these tasks can have positive impact on user engagement, as such methods can help users navigate YouTube's large content[1]. We show quantitative and qualitative results on video recommendation and annotation, showing that our metric space improves performance over our baselines.

To summarize, we propose Collaborative Deep Metric Learning (CDML) approach which utilizes both video content and CF information. Specifically, our deep embedding of audio-visual content defines a metric space that is trained to reconstruct the CF information. Since the audio-visual content is available at upload time, our model can do inference right-away, and can then assist practical tasks of video recommendation and annotation. Qualitative analysis suggests that our embedding function generalizes beyond simple visual and audial similarity, and can capture higher-level semantic relationships. Our main contributions are:

- We train a mapping from video content to CF signals, capturing high-level semantic relationship between videos. This content-aware embedding can be used towards solving the cold-start problem.
- We verify that our embedding generalizes and can be transferred to various problems, including personalized video recommendation and video annotation.
- The proposed method is scalable and can be deployed on large-scale video sharing platform like YouTube. We describe engineering concerns and solutions regarding scalability issues, and evaluate on large-scale datasets.

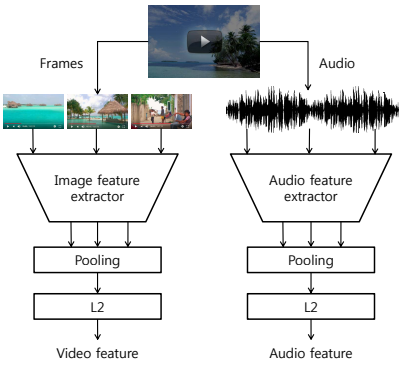
We start by reviewing related work in literature in Section 2. Then, we introduce our proposed approach with details in Section 3. In Section 4, we demonstrate performance of our system in various video understanding problems, both on public dataset and with real system. We conclude with our contributions and future work in Section 5.

2 RELATED WORK

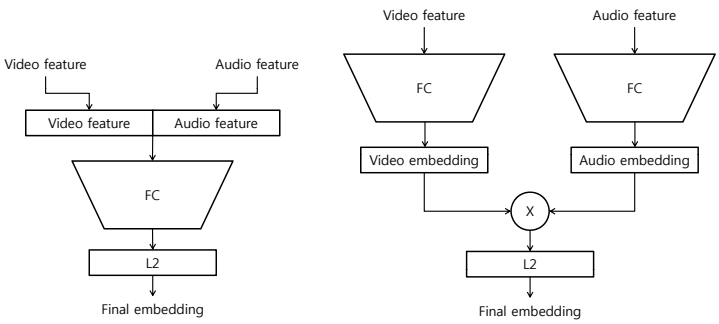
There are many related work modeling pair-wise similarities of items in a learned metric space. It is firstly introduced for nearest neighbor classification in early papers [13, 44], followed by a seminal paper [56] for clustering. It has been applied to many areas, including text retrieval [26], recommendation [32], and bioinformatics [57]. See the survey by Bellet et al. [4] for traditional methods and other applications. Recently, "deep metric learning" [6] is widely used in computer vision, such as image classification [20, 38] and face recognition [8].

Deep neural networks are also popular on other video understanding tasks. Content-based video classification (or video annotation) takes advantage of convolutional neural networks (CNN) for image (frame) understanding [24] as well as recurrent neural networks (RNN) for temporal aspect of videos. [46, 49, 55, 58, 60] Fernando et al. [12] introduced rank-pooling for end-to-end

[1]YouTube receives 300 hours of video uploads every minute, http://youtube-trends.blogspot.com/2015/05/y-is-for-you-10yearsofyoutube_26.html



(a) **Feature extraction**. From sampled frames and low-level audio signals, we extract video and audio features with pre-trained models.



(b) Two possible variants of **embedding network** f. Video and audio features are combined, either from the beginning (early fusion; *left*) or with element-wise multiplication after two separate towers for each of them (late fusion; *right*).

Figure 2: The network architecture for Collaborative Deep Feature Embedding

video classification. Deep learning also has been applied to video search [21, 59].

Collaborative filtering [15] has been widely used in recommender systems, commonly with matrix factorization [27–30, 42]. There is a large wealth of works that employ editorial content-based information, such as movie genres, description and user-generated tags, traditionally known as content-based filtering. [9, 33, 36, 37, 47, 48, 62] We do not use editorial information, with the exception of movie titles, which we only use to collect YouTube trailers for movies.

Our approach is also related to semi-supervised node classification, where we predict labels for all unlabeled nodes, given a partially-labeled graph with its node features. Broadly speaking, it can be divided into two classes. The first one learns a model of labels from features, and uses the graph to "regularize" the model [3, 41], and the second propagates latent node representations through edges and the aggregated information at each node is used to predict its label [10, 25], but they require the graph during inference. We are more similar to the first, in that we do not need a graph during inference. However, we differ from them because the graph (i.e., Collaborative Filtering information) is our optimization target, rather than a regularizer. Most similarly, Van den Oord et al. [51] trains a deep network to predict song ratings from the sound wave. This is similar to ours, except that we use a triplet learning objective and apply our method on videos with audio-visual information.

3 PROPOSED METHOD

In this section, we detail our video metric learning method. We start by describing how we extract video and audio features, followed by our triplet-based neural network embedding model.

3.1 Video and Audio Features

Since videos are large in size, we follow [1] by extracting visual and audio features from videos using pre-trained models. Feature extraction step is illustrated in Figure 2(a).

In particular, we extract image features at 1-frame-per-second using an Inception-v3 network [50], trained on the JFT dataset [19] which contains 100 million labeled images. We fetch the ReLu activation of the last hidden layer, before the classification layer. Afterwards, we apply PCA (and whitening) to reduce feature dimensions to 1,500 for storage and computational reasons. These frame-level features are aggregated into video-level by average pooling. There may be more sophisticated methods to combine frame-level features, but we leave it as a future work.

We extract audio features using a VGG-inspired acoustic model with a modified version of ResNet-50 [18]. Specifically, the audio is divided into non-overlapping 960 ms frames, and then decomposed with a short-time Fourier transform with 25 ms windows for every 10 ms, producing 64 mel-spaced frequency spectrogram. We feed 100 feature frames (corresponding to 1 second) into the ResNet [17], followed by average pooling to aggregate them into video level.

3.2 Collaborative Deep Metric Learning

The visual and audial features we extract above compactly represent the video content, but do not contain information about relationship between video pairs. We train an embedding function f over the content features to predict the *collaborative* signals.

3.2.1 Training Objective. Given a streaming session, where a user watches videos $\{v_1, v_2, v_3 \ldots\}$ on YouTube, we call v_k and v_{k+1} are *co-watched*. We aggregate co-watches from multiple users to construct a graph on videos, where the edges are weighted by the co-watch frequency. In practice, we may drop edges below some threshold. We would like co-watched videos to appear close in the embedding space.

In order to do so, we optimize a ranking triplet loss [43], where a training data point is defined as a triplet of three videos: *anchor*, *positive*, and *negative*. A triplet is constructed such that the *anchor* is relevant to the *positive*, but not to the *negative*, or the *anchor* video is more relevant to the *positive* than to the *negative*. During training, we update the parameters of the embedding network such that the *anchor* is closer to the *positive* than to the *negative* in the embedding space. We select co-watched pair as the *anchor* and *positive*, and randomly sample a *negative*. We train our embedding

function f by solving

$$\min_{\theta} \sum_{i=1}^{n} \mathcal{L}(f_\theta(\mathbf{x}_i^a), f_\theta(\mathbf{x}_i^p), f_\theta(\mathbf{x}_i^n)), \quad (1)$$

where $\mathbf{x}_i^a$, $\mathbf{x}_i^p$, and $\mathbf{x}_i^n$ correspond to the feature vector of *anchor*, *positive*, and *negative* video of the i-th training data point, respectively, $f_\theta(\cdot)$ is the embedding function (e.g, a neural network) parameterized by θ, and $\mathcal{L}$ is a loss function penalizing if $f_\theta(\mathbf{x}_i^a)$ is closer to $f_\theta(\mathbf{x}_i^n)$ than to $f_\theta(\mathbf{x}_i^p)$. (Henceforth, we may omit θ to denote f for simplicity.) Some widely-used loss functions include hinge loss $\mathcal{L}_{\texttt{hinge}}(\mathbf{x},\mathbf{y},\mathbf{z}) = [\|\mathbf{x}-\mathbf{y}\|_2^2 - \|\mathbf{x}-\mathbf{z}\|_2^2 + \alpha]_+$, log loss $\mathcal{L}_{\texttt{log}}(\mathbf{x},\mathbf{y},\mathbf{z}) = \log(1 + \exp\{\|\mathbf{x}-\mathbf{y}\|_2^2 - \|\mathbf{x}-\mathbf{z}\|_2^2 + \alpha\})$, and exponential loss $\mathcal{L}_{\texttt{exp}}(\mathbf{x},\mathbf{y},\mathbf{z}) = \exp\{\|\mathbf{x}-\mathbf{y}\|_2^2 - \|\mathbf{x}-\mathbf{z}\|_2^2 + \alpha\}$, where α is a margin parameter. We use the hinge loss with $\alpha = 0$.

3.2.2 Embedding Network. We feed the extracted features in Section 3.1 into the embedding network f. We propose two possible forms in Figure 2(b): *early fusion* and *late fusion*. The first concatenates the two modalities (vision and audio) at the input, and the second combines them after separate fully-connected layers. There can be other choices, for example, mid-level fusion, but we only experiment with the forementioned two. In both cases, we apply L_2 normalization at the output.

The parameters of embedding network f are trained to minimize the objective in Eq. (1). Since the output of f is L_2 normalized, i.e., $\|f(\mathbf{x}))\|_2 = 1$, the pairwise negative distance $-\|f(\mathbf{x}_1) - f(\mathbf{x}_2)\|$ provides the same ordering as the dot-product $f(\mathbf{x}_1)^\top f(\mathbf{x}_2)$ or cosine similarity $\cos(f(\mathbf{x}_1), f(\mathbf{x}_2))$.

4 EXPERIMENTS

In this section, we extensively evaluate the proposed method and generated embeddings in various video understanding tasks: 1) related video retrieval, 2) personalized video recommendation, and 3) video annotation/classification.

4.1 Related Videos Retrieval

Related video retrieval problem is the main task we train on. Given a query video q with its content features $\mathbf{x}_q$, we retrieve related videos in a candidate set. This *relatedness* is defined as how much the two videos are co-watched by users. Using our embedding network f, we embed the candidates videos and take the top k with largest cosine similarity to the query video. Formally, the score between a candidate c and query video q is given by

$$\cos(f(\mathbf{x}_q), f(\mathbf{x}_c)) = f(\mathbf{x}_q)^\top f(\mathbf{x}_c). \quad (2)$$

4.1.1 Experimental Settings. To train the model, we extract video and audio features from 278M YouTube videos with 1,000 or more views. We randomly split the videos into training and eval partition with 7:3 ratio, and create the training triplets based from the co-watch graph with videos in the training partition only. After training, we evaluate with two different cold-start cases: 1) where we retrieve fresh (eval) videos for a query from train partition (*train-to-eval, T2E*), and 2) where we retrieve established videos for a query with fresh video (*eval-to-train, E2T*). We evaluate end-to-end retrieval performance in two widely-used ranking metrics:

Input Features	NDCG		MAP	
	T2E	E2T	T2E	E2T
Video Only	7.22%	10.38%	2.27%	2.68%
Video + Audio	**8.48%**	**12.04%**	**2.70%**	**3.20%**

Table 1: Related video retrieval performance of our proposed model trained with video features only vs. video + audio features. We clearly see that the audio signal helps to improve end-to-end performance.

1) Normalized Discounted Cumulative Gain (**NDCG**) considers the order of retrieved items in the list. DCG@k is defined as:

$$DCG@k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{\log_2(i+1)}, \quad (3)$$

where i is the position in the list and $rel_i \in \{0, 1\}$ indicates whether the i-th item is relevant to the user or not. NDCG is the ratio of DCG to the maximum possible DCG for that query video. This maximum occurs when the recommended items are presented in decreasing order of user preference. We use $k = 10$ for our experiment.

2) Mean Average Precision **(MAP)** is the area under precision-recall curve, and is given by:

$$MAP = \frac{1}{|Q|} \sum_{q \in Q} \int_0^1 P_q(r) dr \approx \frac{1}{|Q|} \sum_{q \in Q} \sum_{k=1}^{n} P_q(k) \Delta r(k), \quad (4)$$

where r ranges over all possible recall levels and $P_q(r)$ is the precision at recall level r for the query q. In practice, the integral is replaced with a finite sum over every position k in the ranked list of n videos.

For all experiments, we apply semi-hard negative mining [43] within a large mini-batch of 7,200 triplets. We choose this size based on empirical evidence, observing inferior performance with smaller but almost no improvement with larger size than this. Intuitively, it is important to mine hard examples as we train, since original triplets created with random negatives become too easy after some iterations. We assign to each triplet the hardest (mostly violating anchor-positive-negative relationship) negative within the mini-batch to facilitate training. For the triplet loss gap parameter, we cross-validate over $\{0.1, 0.3, 0.5\}$, and search learning rate among $\{0.001, 0.005, 0.01, 0.03, 0.05, 0.07, 0.1\}$. We apply learning rate decay of $\{0.2\%, 0.4\%, 0.6\%\}$ for every 100,000 or 150,000 training steps.

4.1.2 Results and Discussion. At first, we compare performance of our model trained only with visual signal vs. with both visual and audio features. Video-only model is equivalent to the left (video) tower of the late fusion embedding network in Figure 2(b). Table 1 shows NDCG and MAP scores for both models. In both types of cold-start scenarios, we see that the audio signal clearly helps to improve end-to-end retrieval performance over the video-only model.

Another interesting parameter is the size of output embedding vector. There is general trade-off between capacity and cost; the larger the size, the more expressive the feature can be, potentially leading to better performance. For this, however, we pay the cost for longer training time with more computational resources. If we are about to serve the features online, the storage size may be also an

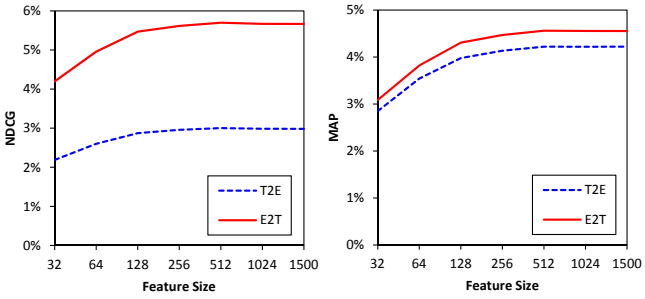


Figure 3: Related video retrieval performance with various output feature size. We choose the 256-dimensional embedding, which performs almost close to the larger ones.

Architecture (Fusion)	NDCG		MAP	
	T2E	E2T	T2E	E2T
2560-256 (Early)	8.48%	12.04%	2.70%	3.20%
4000-256 (Early)	9.51%	13.13%	3.01%	3.49%
4000-256 (Late)	**9.91%**	**13.46%**	**3.13%**	**3.58%**
4000-512-256 (Early)	9.29%	12.64%	2.90%	3.33%
4000-1024-256 (Early)	9.46%	12.80%	2.95%	3.38%
4000-1024-256 (Late)	9.82%	13.05%	3.07%	3.44%

Table 2: Related video retrieval performance with different network architectures. We observe wider layer helps more than deeper models. Late fusion generally outperforms early fusion.

important concern, preventing large feature size despite its better performance. In Figure 3, we see that increasing the feature size above 512 gives little gain. On the other hand, the performance dramatically drops when the size is smaller than 128. We chose 256 dimensions as a good compromise between performance and computational cost.

We also compare several different neural network architectures, with shallower vs. deeper models, different number of nodes in hidden layers, and early vs. late fusion. Table 2 summarizes end-to-end performance with different architectures we tried. Interestingly, we see little advantage with deeper models. It is probably because the input features have already gone through deep models (Inception and ResNet), and this embedding function may not be that complex to take advantage of deeper models. Another observation is that increased width of the first hidden layer actually helps to improve the performance. Lastly, late fusion tends to always perform better than early fusion. This might be because one signal is almost ignored with early fusion if it is not strong enough to survive against the other. From this empirical evidence, we choose 4000-256 architecture with late fusion.

4.1.3 Demonstration. We demonstrate qualitative performance of our proposed method on YouTube-8M dataset [1], which consists of about 8 million videos of 450K hours. To illustrate cold-start situation, we choose all query videos from the eval partition (approximately 30%), thus no query video has been shown to the model during training. All videos in the dataset other than the chosen query are considered as candidates. For each query video, we retrieve the most relevant top-k (here, $k = 4$) videos to the query using Eq. (2). We do not use meta-data other than audio-visual signals.

Figure 4 illustrates some examples of the result. From the top, 1) the query video is about Sergio Busquets, a Spanish soccer player. All retrieved videos are about soccer, and in the last two thumbnails we see the same uniform from F.C. Barcelona. This shows our proposed method effectively finds out similarity even in fine details. 2) The second query is a video game (Candy Crush). All retrieved videos show the same game. 3) Next query is about a hamster. We can see hamsters in all retrieved videos, but each hamster is in different color and shape. We can infer that our embedding captures not just visual but semantic similarity. 4) In the next seed video, a girl demonstrates how to make her hair braided ('tresse' in French, seen in the title). The retrieved videos look semantically very relevant, as they are also showing how to braid hair. Interestingly, the title of the first and third recommended videos are in Russian and Korean, respectively. This is natural since we take visual and audio signal only, not the title or other meta-data like language. 5) The last row shows that for a concert video it retrieves other concerts with similar background and music. Overall, visual and audial features are powerful to retrieve relevant videos from the corpus, but in practice may be used in conjunction with other sources like meta-data to compensate its blind spots, such as language.

4.2 Video Recommendation

Personalized video recommendation is the task to rank items for a particular user in the order of her preference. It is similar to related video retrieval, except that here the query (context) is a user instead of a single video. We model a user with her recent watch history, i.e., a sequence of videos. Computing similarity with a candidate video now gives a sequence of scores, one for each recently watched video. We would like to aggregate these scores into one. One possible aggregation is the *arithmetic mean*, where the ranking function returns videos that maximize the score:

$$\max_{v \in V-Q} \frac{1}{|Q|} \sum_{q \in Q} \cos(f(\mathbf{x}_q), f(\mathbf{x}_v)), \tag{5}$$

where Q is the set of videos the user has recently watched, and V is the set of candidate videos. Another aggregation is *max*, where the score of a candidate video is the maximum cosine similarity across videos in Q. Specifically,

$$\max_{v \in V-Q} \max_{q \in Q} \cos(f(\mathbf{x}_q), f(\mathbf{x}_v)). \tag{6}$$

4.2.1 Scalability. In real services like YouTube, $|V|$ can be millions or billions, but these recommendations need to be computed in micro-seconds. Video embeddings may be cached for quick retrieval, so it is important to reduce the feature size to save storage. For this, we quantize the embedding values. Specifically, we choose 2^k representative values to minimize the expected squared distance between original and quantized features, representing each dimension with k bits. For faster similarity computation, we pre-compute and cache all possible combinations of multiplications in a lookup table, minimizing expensive floating-point operation at serving time.

Figure 4: Demonstration of related video retrieval with YouTube-8M Dataset. The left-most column is the query video, and other videos in the same row are top 4 most relevant videos found by our model. We show YouTube thumbnail, title, and relevance score we computed (in red italic).

Output Dim	Quantization	Size (byte)	T2E	E2T
512	original	2048	2.99%	5.68%
256	original	1024	2.96%	5.61%
32	original	128	2.19%	4.20%
128	8 bit/dim	128	2.87%	5.47%
256	4 bit/dim	128	**2.94%**	**5.58%**
512	2 bit/dim	128	2.87%	5.47%
128	4 bit/dim	64	**2.84%**	**5.40%**
256	2 bit/dim	64	2.75%	5.23%
128	2 bit/dim	32	2.48%	4.74%

Table 3: Comparison in NDCG for various feature size with quantization. We observe that 4 bits per value are enough to almost preserve end-to-end recommendation performance.

We empirically test the performance we can retain with different levels of quantization. We quantize each dimension into 2, 4, and 8 bits (from 4 bytes float), allowing 4, 16, and 256 representative values, respectively. Table 3 compares end-to-end performance with various quantization options. The top two rows show the best performance we achieved without quantization. When we reduce the output size to 128 bytes (16x from the best), we compare 4 options: {32 floats, 128D * 8 bits, 256D * 4 bits, 512D * 2 bits}. We observe that 4 bits per dimension are enough to preserve the original performance, as NDCGs dropped just 0.02% and 0.03% for each scenario. 2 bits per dimension seem not enough to preserve information, when we compare it against performance with original 512D (the top row). When we reduce further to 64 bytes, we still achieve reasonable performance with 4 bits/dim quantization with 128D. In conclusion, we can reasonably represent a video with just 128 or 64 bytes, which is compact enough to deploy in large-scale platform like YouTube.

Another computational bottleneck of this system is solving Eq. (5) and (6). A naive implementation of this may iterate over all possible pairs of q and v, leading to quadratic time complexity $O(|Q||V|)$. Since $\|f(\mathbf{x})\|_2 = 1$, the cosine becomes a dot-product and average aggregation in Eq. (5) can be done in linear time, taking

advantage of distributive property of inner-product as:

$$\max_{v \in V-Q} \frac{1}{|Q|} \sum_{q \in Q} f(\mathbf{x}_q)^\top f(\mathbf{x}_v) = \max_{v \in V-Q} \left(\frac{1}{|Q|} \sum_{q \in Q} f(\mathbf{x}_q) \right)^\top f(\mathbf{x}_v)$$

Averaging vectors in Q does not rely on v and can be pre-computed once and reused, achieving linear time complexity of $O(|V| + |Q|)$.

4.2.2 Evaluation with MovieLens Trailers. We evaluate our method on MovieLens[2] with amendment [2], one of the most widely-used public dataset for recommendations. As it does not come with video and audio data of movies, we collected movie trailers available on YouTube as a proxy. To obtain the trailer YouTube video IDs for the movies in MovieLens, we programatically searched on google.com with "<MovieTitle> (<year>) trailer" for all movies in MovieLens, and took the first YouTube result, limiting to the top 5 results. Usually, the YouTube result is the third result, after IMDB and Wikipedia. In some cases, our search query did not return a YouTube trailer. We collected 25,141 (94.0%) trailers for the 26,744 unique movies that are rated in MovieLens 20M. Movies without trailers were excluded from experiments.

First, we evaluate the aggregation methods discussed in Section 4.2.1. For each user, we split ratings into training and test partitions by 5:5 ratio. Users with less than 10 test ratings were excluded from the experiment. For each user, we define the set of preferred movies as those in training partition with a rating higher than some threshold: 1) fixed threshold with $\{3, 4, 5\}$ stars or higher; 2) the user's own mean rating; and 3) no threshold (simulating applications with watch history given without explicit ratings). We then rank candidate videos in the test partition by similarity to the preferred movie set, computed as dot-product in our embedding space. We aggregate the scores by average and max aggregation proposed in Eq. (5) - (6). For this experiment, we used actual ground truth rating for rel_i in Eq. (3) to make the setting comparable to [28]. Table 4 summarizes performance on MovieLens 100K in NDCG@10 and MAP. Below is our observations and discussions:

- With higher threshold, the performance gets better in general. This makes sense, since with higher threshold we only take videos that we are sure the user liked, as long as we have enough ratings from the user. We see some performance drop when we do not filter out videos with low ratings, but in applications only with implicit feedback (e.g, clicks, views) this may be the only option.
- Max aggregation performs generally better than average aggregations. In other words, it performs better to recommend the most similar item to one of the user's favorites, probably because many users have more than one preferred types of movies. We also tried some variations of these aggregations, such as average of top $k = 2, 3, 5$, and variance, but these did not outperform max and average aggregations.

4.2.3 Comparison with Existing Methods. We follow the standard experimental setup in literature [11, 28, 52, 54], using the same dataset (MovieLens 100K). For each user, N randomly chosen ratings, with $N = 10, 20, 50$, are used for training. All other items are used for evaluation. Users with less than $N + 10$ ratings are ignored to guarantee at least 10 items for testing.

[2]https://grouplens.org/datasets/movielens/

Aggregation	Watch History Threshold	NDCG	MAP
Max	Higher than user mean	0.6242	0.7186
Max	5 stars only	**0.6282**	0.7155
Max	4 stars or higher	0.6251	**0.7196**
Max	3 stars or higher	0.6149	0.7141
Max	All rated movies	0.6071	0.7078
Average	Higher than user mean	0.6022	0.7073
Average	5 stars only	0.6133	0.7120
Average	4 stars or higher	0.6018	0.7065
Average	3 stars or higher	0.5954	0.7013
Average	All rated movies	0.5900	0.6966

Table 4: Recommendation performance with different score aggregations. Max aggregation with higher threshold tends to perform better.

Model	Average Precision			NDCG@10		
	$N = 10$	$N = 20$	$N = 50$	$N = 10$	$N = 20$	$N = 50$
CofiRank [54]	0.6632	0.6825	0.6915	0.6502	0.6629	0.6912
SVD	0.7000	0.7454	0.7510	0.5594	0.6514	0.6750
NMF [27]	0.7107	0.7234	0.7328	0.6285	0.6336	0.5956
BPMF [42]	0.6376	0.6517	0.6915	0.5636	0.5513	0.6746
LLORMA [30]	0.7341	0.7424	0.7490	0.6712	0.6682	0.6932
LCR [28]	0.7406	0.7503	0.7626	0.7152	0.6908	0.7039
CDML (ours)	0.7030	0.7215	0.7442	0.5725	0.6213	0.6941

Table 5: Recommendation performance compared with existing methods. Other than SVD and CDML, we use the reported values in [28], which were obtained using the same protocol as reported here.

Table 5 compares recommendation performance of the proposed method against several existing methods, in NDCG and MAP. We observe that the proposed approach performs comparable to most of the baselines, including CofiRank [54] and Bayesian PMF [42]. Although it does not outperform the state-of-the-art, Local Collaborative Ranking [28], it is remarkable for the proposed method to accomplish comparable performance to most existing models, considering that 1) the embedding was not optimized for this task but trained for related video retrieval task, and 2) it does not extensively exploits the rating matrix but mostly based on the contents.

Considering this result, we hypothesize that the proposed embedding may perform relatively stronger for colder-start cases, while other CF-based models may outperform with sufficient feedback from users. To test this hypothesis, we conduct further experiments with smaller N, as small as 2. Specifically, we compare performance of the proposed CDML model vs. SVD, one of the most widely used CF-based baselines. We use rank 5 for SVD, as it is enough capacity for this extremely sparse matrix with small N. Figure 5 shows that the proposed method outperforms the CF counterpart with less than 14 (NDCG) or 10 (MAP) ratings per user, confirming that the content-aware model is powerful when we know less about the user.

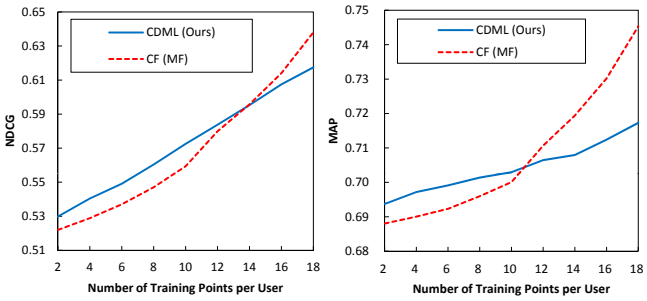


Figure 5: Cold-start recommendation performance with different number of training data points per user, in NDCG (*left*) and MAP (*right*). We see that our CDML is relatively stronger for colder start cases.

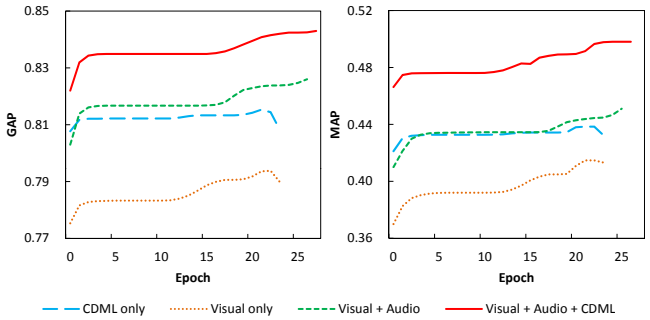


Figure 6: YouTube-8M video classification training curve comparing different features. Adding CDML features improves classification accuracy, by bringing the complimentary user behavior information to the content information.

4.3 Video Annotation

We consider a video annotation task, where the goal is to predict one or more labels for a video, following the setup of [1]. Considering only a finite set of labels, we can model this as a multi-labeled classification problem. Formally, given a training set $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1,\ldots,N}$ over d classes, where $\mathbf{x}_i$ is the feature vector for i-th video, and $\mathbf{y}_i \in \{0,1\}^d$ is a binary vector with 1 on corresponding classes and 0 everywhere else, the task is estimating $\mathbf{y}$ on unseen $\mathbf{x}$.

4.3.1 Annotation Model. We use a fully connected layer (with L_2 norm) followed by a Mixture of Experts (MoE) model [22] for our multi-label classifier. Specifically, a video feature $\mathbf{x}$ is as a concatenation of input audio-visual features. We PCA each feature into 256D by

$$\mathbf{z} = \frac{\mathbf{A}(\mathbf{x} - \mu)}{\|\mathbf{A}(\mathbf{x} - \mu)\|_2} \tag{7}$$

where A is block PCA matrix shared across all labels and μ is the mean of $\mathbf{x}$. Given $\mathbf{z}$, MoE model estimates the probability $p(e|\mathbf{z})$ for an entity e exist in the video as a weighted average over experts $h \in \mathcal{H}_e$. For each expert h, we use a binary logistic regression classifier, $p(e|\mathbf{z}, h) = \sigma(\mathbf{w}_h^\top \mathbf{z})$. Overall, the classifier is given by

$$p(e|\mathbf{z}) = \sum_{h \in \mathcal{H}_e} p(h|\mathbf{z}) p(e|\mathbf{z}, h) = \sum_{h \in \mathcal{H}_e} p(h|\mathbf{z}) \sigma(\mathbf{w}_h^\top \mathbf{z}), \tag{8}$$

where $p(h|\mathbf{z})$ is a softmax over $|\mathcal{H}_e| + 1$ with a dummy state indicating non-existence of the entity. To train this model, We minimize log-loss $\mathcal{L}(p, g) = -g \log p - (1-g)\log(1-p)$ over the entire training data, where $g \in \{0,1\}$ is the ground truth and p is $p(e|\mathbf{z})$.

In all of our experiments, we first train the MoE model with $\{5, 10, 30, 100\}$ mixtures using Adam optimizer with a learning rate of 0.008 and a mini-batch size of 512. Then, the block PCA matrix and the MoE are alternatively fine-tuned using a full batch optimization technique named RPROP [40]. Given the output $p(e|\mathbf{z})$ for all classes e, we take the top k labels for the video.

4.3.2 Experimental Settings. We evaluate in two widely-used metrics:

(1) Global Average Precision (**GAP**) for the top k predicted labels is given by

$$GAP = \sum_{i=1}^{k|Q|} P(i) \Delta R(i), \tag{9}$$

where Q is the set of query videos, $P(i)$ is the precision, and $R(i)$ is the recall. This is the metric used in the first YouTube-8M video classification challenge, so we use the same $k = 20$ for comparison.

(2) Mean Average Precision (**MAP**): this is same as Eq. (4), where Q is the set of query videos.

GAP computes overall accuracy of the predicted labels on the entire dataset, while MAP evaluates each video individually and takes average of the scores. If the labels are uniformly distributed over all videos, GAP and MAP will be equal. As most dataset have long-tailed distribution over labels (so does YouTube-8M dataset), MAP penalizes more on errors with videos corresponding to less frequent labels.

We use the input video and audio features extracted from pre-trained feature extractors in Figure 2(a) as our baseline. Comparing against these features reveals the direct improvement with co-watch information. For fair comparison, we took PCA over these input features to have the same dimensionality (256) with our embedding. In addition, we also compare against the top-performers at the 1st YouTube-8M video classification challenge.

4.3.3 YouTube-8M Results and Discussion. YouTube-8M [1] is the largest video classification benchmark dataset as of this writing. It consists of 8 millions of videos with labels over 4,716 classes. Each video has 3.4 labels on average, representing the main themes of the video.

We first compare the performance of several features and their combinations with the same MoE model. Figure 6 presents how GAP and MAP scores improve with MoE training. It is interesting to note that the model with all three features (visual + audio + CDML) outperforms the baselines with visual + audio, by 2.6% (GAP) and 3.1% (MAP). Considering that the CDML feature itself is also trained on top of the same visual and audio features, the user behavior information that CDML brings into is complimentary to

Rank	Team Name	Video-level features only		Frame-level features used	
		Single-model	Ensembled	Single-model	Ensembled
1	WILLOW [34]	—	—	0.8300	0.8469
2	monkeytyping [53]	0.8106	0.8225	0.8179	0.8458
3	offline [31]	0.8082	—	0.8275	0.8454
4	FDT [7]	—	—	0.8178	0.8419
5	You8M [45]	—	0.8308	—	0.8418
6	Rankyou [61]	0.8141	—	0.8246	0.8408
7	Yeti [5]	—	—	0.8254	0.8396
8	SNUVL X SKT [35]	—	—	0.8200	0.8389
9	Lanzan Ramen	—	—	—	0.8372
10	Samaritan [63]	—	—	0.8139	0.8366
Video + Audio		0.8247	—	—	—
Video + Audio + CDML		**0.8430**	—	—	—

Table 6: Comparison with top performers at the First YouTube-8M video classification challenge (CVPR 2017). All scores are in GAP, Eq. (9).

the content information. It has been known that watch behavior is informative for video recommendation, but we verify that it actually improves classification accuracy as well.

At CVPR 2017, the creators of YouTube-8M hosted a challenge on large-scale video classification[3]. Table 6 lists the top 10 performers with their highest GAP scores. For comparison, we also add a line for another baseline "video + audio", where we simply concatenate video and audio features we extracted, without CDML. All the top 10 teams applied ensemble methods, which boost the performance by 1-2%, so we listed the best performance with and without ensembles. Also, some participants provided performance on models only utilizing video-level features (not using frame-level features with temporal models). As our models are video-level only, it is fair to compare against the video-level models. As shown in Table 6, our proposed embedding performs better than any other top 10 teams dealing with video-level features only, regardless of ensembling. Even if we take frame-level models into account, it still performs better than any single model in the table. It turns out that only the top 3 ensembled models slightly perform better than ours. (Note that this comparison is mainly to show how it performs compared to the state-of-the-art in video annotation models. It may not be directly comparable as we use additional data for training.)

4.3.4 MovieLens-20M Results and Discussion. MovieLens-20M [16] comes with genome tags for 27,278 movies. We exclude infrequent tags with less than 200 movies associated, leaving 187 most frequent tag classes as ground truth labels. Table 7 shows that GAP improves when we concatenate our features with the audio-visual features.

Features	GAP
Video only	0.2717
Audio only	0.2429
Video + Audio	0.2972
Video + Audio + CDML	**0.3115**

Table 7: Video annotation result on MovieLens-20M. CDML features improve classification accuracy over other baselines.

5 SUMMARY AND FUTURE WORK

We propose to model collaborative signals over videos using their audio-visual content. Specifically, we train a deep network that takes as input video features and outputs an embedding that preserves pair-wise video-to-video collaborative similarity. Then, we use the trained embedding network for various large-scale video understanding tasks, including related video retrieval, personalized video recommendation, and multi-label video annotation. We show that our trained embeddings improve baselines on all, even though our embedding was not directly trained for these tasks.

Our work sets a baseline in learning features that generalize across video tasks. The work can be extended along different directions, including incorporation of modern neural architectures for extracting input features like LSTMs, modern metric learning techniques, e.g., via lifted structured embedding, or modern graph learning methods such as propagating information through the edges via Graph Convolutional Networks. We hope that other researchers find our work inspiring, towards training a universal video network that generalizes across many challenging video understanding tasks.

[3] https://www.kaggle.com/c/youtube8m

REFERENCES

[1] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan. 2016. YouTube-8M: A large-scale video classification benchmark. *arXiv:1609.08675* (2016).

[2] S. Abu-El-Haija, J. Lee, M. Harper, and J. Konstan. 2018. MovieLens YouTube Trailers. (2018).

[3] M. Belkin, P. Niyogi, and V. Sindhwani. 2006. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. (2006).

[4] A. Bellet, A. Habrard, and M. Sebban. 2013. A survey on metric learning for feature vectors and structured data. *arXiv:1306.6709* (2013).

[5] M. Bober-Irizar, S. Husain, E. J. Ong, and M. Bober. 2017. Cultivating DNN Diversity for Large Scale Video Labelling. *arXiv:1707.04272* (2017).

[6] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah. 1994. Signature verification using a" siamese" time delay neural network. *Advances in Neural*

Information Processing Systems (1994).

[7] S. Chen, X. Wang, Y. Tang, X. Chen, Z. Wu, and Y. G. Jiang. 2017. Aggregating frame-level features for large-scale video classification. *arXiv:1707.00803* (2017).

[8] S. Chopra, R. Hadsell, and Y. LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition* (2005).

[9] L. de Campos, J. Fernandez-Luna, J. Huete, and M. Rueda-Morales. 2010. International Journal of Approximate Reasoning. *International Journal of Approximate Reasoning* (2010).

[10] D. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams. 2015. Convolutional Networks on Graphs for Learning Molecular Fingerprints. In *Advances in Neural Information Processing Systems*.

[11] C. Fan, Y. Lan, J. Guo, Z. Lin, and X. Cheng. 2013. Collaborative factorization for recommender systems. In *Proc. of the International ACM SIGIR Conference on Information Retrieval*.

[12] B. Fernando and S. Gould. 2016. Learning end-to-end video classification with rank-pooling. In *Proc. of the International Conference on Machine Learning*.

[13] J. Friedman et al. 1994. *Flexible metric nearest neighbor classification.* Technical Report. Technical report, Dept. of Statistics, Stanford University.

[14] R. Geirhos, D. H. J. Janssen, H. H. SchÃijtt, J. Rauber, M. Bethge, and F. A. Wichmann. 2017. Comparing deep neural networks against humans: object recognition when the signal gets weaker. *arXiv:170606969* (2017).

[15] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. 1992. Using Collaborative Filtering to Weave an Information Tapestry. 35, 12 (1992), 61–70.

[16] F. M. Harper and J. A Konstan. 2016. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 5, 4 (2016), 19.

[17] K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep residual learning for image recognition. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*.

[18] S. Hershey, S. Chaudhuri, D. PW Ellis, J. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. Saurous, B. Seybold, et al. 2017. CNN architectures for large-scale audio classification. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

[19] G. Hinton, O. Vinyals, and J. Dean. 2015. Distilling the knowledge in a neural network. *arXiv:1503.02531* (2015).

[20] Song H.O., Xiang Y., and Jegelka S. 2016. Deep Metric Learning via Lifted Structured Feature Embedding. (2016).

[21] L. Jiang, Y. Kalantidis, L. Cao, S. Farfade, J. Tang, and A. Hauptmann. 2017. Delving deep into personal photo and video search. In *Proc. of the ACM International Conference on Web Search and Data Mining (WSDM)*.

[22] M. I Jordan and R. A Jacobs. 1994. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation* 6, 2 (1994), 181–214.

[23] N. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. A. Bhatia, et al. 2017. In-datacenter performance analysis of a tensor processing unit. In *Proc. of the Annual International Symposium on Computer Architecture*.

[24] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. 2014. Large-scale video classification with convolutional neural networks. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*.

[25] T. Kipf and M. Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proc. of the International Conference on Learning Representations*.

[26] G. Lebanon. 2006. Metric learning for text documents. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28, 4 (2006), 497–508.

[27] D. Lee and S. Seung. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature* 401, 6755 (1999), 788.

[28] J. Lee, S. Bengio, S. Kim, G. Lebanon, and Y. Singer. 2014. Local collaborative ranking. In *Proc. of the International Conference on World Wide Web*.

[29] J. Lee, S. Kim, G. Lebanon, and Y. Singer. 2013. Local low-rank matrix approximation. In *Proc. of the International Conference on Machine Learning*.

[30] J. Lee, S. Kim, G. Lebanon, Y. Singer, and S. Bengio. 2016. LLORMA: Local low-rank matrix approximation. *The Journal of Machine Learning Research* 17, 1 (2016), 442–465.

[31] F. Li, C. Gan, X. Liu, Y. Bian, X. Long, Y. Li, Z. Li, J. Zhou, and S. Wen. 2017. Temporal modeling approaches for large-scale youtube-8m video understanding. *arXiv:1707.04555* (2017).

[32] B. McFee, L. Barrington, and G. Lanckriet. 2012. Learning content similarity for music recommendation. *IEEE Transactions on Audio, Speech, and Language Processing* 20, 8 (2012), 2207–2218.

[33] P. Melville, R. J. Mooney, and R. Nagarajan. 2001. Content-Boosted Collaborative Filtering. In *Proc. of the SIGIR Workshop on Recommender Systems*.

[34] A. Miech, I. Laptev, and J. Sivic. 2017. Learnable pooling with Context Gating for video classification. *arXiv:1706.06905* (2017).

[35] S. Na, Y. Yu, S. Lee, J. Kim, and G. Kim. 2017. Encoding Video and Label Priors for Multi-label Video Classification on YouTube-8M dataset. *arXiv:1706.07960* (2017).

[36] D. Y. Pavlov and D. M. Pennock. 2002. A maximum entropy approach to collaborative filtering in dynamic, sparse, high-dimensional domains. In *Advances in Neural Information Processing Systems*.

[37] D. M. Pennock, E. Horvitz, S. Lawrence, and C. L. Giles. 2000. Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach. In *Proc. of the Conference on Uncertainty in Articifial Intelligence (UAI)*.

[38] Q. Qian, R. Jin, S. Zhu, and Y. Lin. 2015. Fine-grained visual categorization via multi-stage metric learning. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition* (2015).

[39] E. Real, J. Shlens, S. Mazzocchi, X. Pan, and V. Vanhoucke. 2017. Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[40] M. Riedmiller and H. Braun. 1992. RPROP-A fast adaptive learning algorithm. In *Proc. of ISCIS VII*.

[41] R. Rothe, R. Timofte, and L. V. Gool. 2016. Some like it hot - visual guidance for preference prediction. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*.

[42] R. Salakhutdinov and A. Mnih. 2008. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proc. of the International Conference on Machine Learning*. 880–887.

[43] F. Schroff, D. Kalenichenko, and J. Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*.

[44] R. Short and K. Fukunaga. 1981. The optimal distance measure for nearest neighbor classification. *IEEE Transactions on Information Theory* 27, 5 (1981), 622–627.

[45] M. Skalic, M. Pekalski, and X. Pan. 2017. Deep Learning Methods for Efficient Large Scale Video Labeling. *arXiv:1706.04572* (2017).

[46] N. Srivastava, E. Mansimov, and R. Salakhudinov. 2015. Unsupervised learning of video representations using lstms. In *Proc. of the International Conference on Machine Learning*.

[47] F. Strub, J. Mary, and R. Gaudel. 2016. Hybrid Collaborative Filtering with Neural Networks. *CoRR* abs/1603.00806 (2016).

[48] X. Su, R. Greiner, T. M. Khoshgoftaar, and X. Zhu. 2007. Hybrid collaborative filtering algorithms using a mixture of experts. In *Proc. of the IEEE/WIC/ACM International Conference on Web Intelligence*.

[49] Y. Sun, Z. Wu, X. Wang, H. Arai, T. Kinebuchi, and Y. G. Jiang. 2016. Exploiting Objects with LSTMs for Video Categorization. In *Proc. of the ACM Internacional Conference on Multimedia*.

[50] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. 2015. Going deeper with convolutions. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*.

[51] A. van den Oord, S. Dieleman, and B. Schrauwen. 2013. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems*.

[52] M. Volkovs and R. Zemel. 2012. Collaborative ranking with 17 parameters. In *Advances in Neural Information Processing Systems*. 2294–2302.

[53] H. Wang, T. Zhang, and J. Wu. 2017. The Monkeytyping Solution to the YouTube-8M Video Understanding Challenge. *arXiv:1706.05150* (2017).

[54] M. Weimer, A. Karatzoglou, Q. Le, and A. J. Smola. 2008. Cofi rank-maximum margin matrix factorization for collaborative ranking. In *Advances in neural information processing systems*. 1593–1600.

[55] Z. Wu, X. Wang, Y. G. Jiang, H. Ye, and X. Xue. 2015. Modeling spatial-temporal clues in a hybrid deep learning framework for video classification. In *Proc. of the ACM International Conference on Multimedia*.

[56] E. Xing, M. Jordan, S. Russell, and A. Ng. 2003. Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems*.

[57] H. Xiong and X. W. Chen. 2006. Kernel-based distance metric learning for microarray data classification. *BMC bioinformatics* 7, 1 (2006), 299.

[58] X. Yang, P. Molchanov, and J. Kautz. 2016. Multilayer and multimodal fusion of deep neural networks for video classification. In *Proc. of the ACM International Conference on Multimedia*.

[59] S. Yu, L. Jiang, Z. Xu, Y. Yang, and A. Hauptmann. 2015. Content-based video search over 1 million videos with 1 core in 1 second. In *Proc. of the ACM on International Conference on Multimedia Retrieval*.

[60] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. 2015. Beyond Short Snippets: Deep Networks for Video Classification. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*.

[61] L. Zhu, Y. Liu, and Y. Yang. 2017. UTS submission to Google YouTube-8M Challenge 2017. *arXiv:1707.04143* (2017).

[62] C.-N. Ziegler, G. Lausen, and L. Schmidt-Thie. 2004. Taxonomy-driven computation of product recommendations. In *Proc. of the thirteenth ACM international conference on Information and knowledge management*.

[63] H. Zou, Kun Xu, and J. Li. 2017. The YouTube-8M Kaggle Competition: Challenges and Methods. *arXiv:1706.09274* (2017).