Ranking Domain-specific Highlights by Analyzing Edited Videos

Min Sun, Ali Farhadi, and Steve Seitz

University of Washington

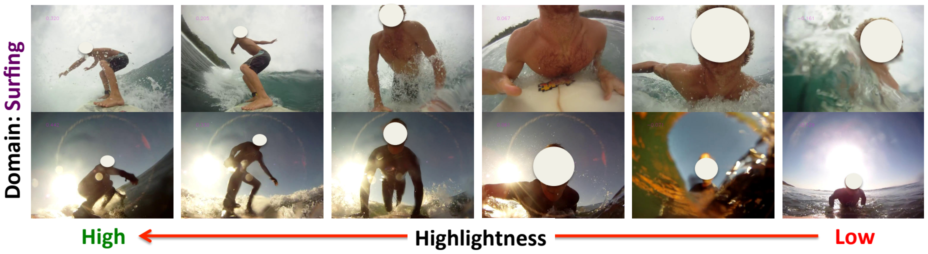

Fig. 1: Retrieving domain-specific highlights (e.g., for surfing) from unconstrained personal videos is an important step toward automatic video editing. Our system automatically learns how to rank the "highlightness" of every moment (a short 2 seconds clip) in a raw video by analyzing edited videos on Youtube. Here we show ranking results of our system on two raw videos (click to watch on Youtube link1, link2) captured by GoPro cameras, where each clip is represented by a frame sampled from the clip.

Abstract. We present a fully automatic system for ranking domain-specific highlights in unconstrained personal videos by analyzing online edited videos. A novel latent linear ranking model is proposed to handle noisy training data harvested online. Specifically, given a search query (domain) such as "surfing", our system mines the Youtube database to find pairs of raw and corresponding edited videos. Leveraging the assumption that edited video is more likely to contain highlights than the trimmed parts of the raw video, we obtain pair-wise ranking constraints to train our model. The learning task is challenging due to the amount of noise and variation in the mined data. Hence, a latent loss function is incorporated to robustly deal with the noise. We efficiently learn the latent model on a large number of videos (about 700 minutes in all) using a novel EM-like self-paced model selection procedure. Our latent ranking model outperforms its classification counterpart, a motion analysis baseline [15], and a fully-supervised ranking system that requires labels from Amazon Mechanical Turk. Finally, we show that impressive highlights can be retrieved without additional human supervision for domains like skating, surfing, skiing, gymnastics, parkour, and dog activity in unconstrained personal videos.

Keywords: video highlight detection, latent ranking

1 Introduction

We increasingly capture large amounts of video data, a trend that is likely to accelerate with new devices like Google Glass. On YouTube alone, 100 hours of video are uploaded every minute. Most video content, however, is not fun to watch; the best videos have usually been carefully and manually *edited* to feature the highlights and trim out the boring segments.

Wouldn't it be great if computers could do the editing for us? I.e., we'd provide raw video footage, and out would pop a high quality edited video. Indeed, both Google and Facebook recently released products that seek to achieve similar goals. Google's Auto-Awesome movie feature generates a video summary from all the footage of an event (complete with filters and background music!). However, Auto-Awesome works best when the event videos are short and contain only highlights. It is not clear how it can handle raw personal videos typically a few minutes long. Facebook's new *Look Back* feature provides similar functionality, but focused on photos and your most popular posts instead of videos. These applications motivate the importance of research in automatic video editing.

As a step towards this goal, we address the problem of retrieving domain-specific *highlights* in raw videos (see Fig. 1). While prior research has explored the highlight selection problem in limited domains, e.g., [27,20,16,26,25,10,4,23], most methods require large amounts of human-crafted training data. Since the definition of a highlight is highly dependent on the domain of interest (e.g., blowing out the candles on a birthday cake, a ski jump, raising glasses in a toast), it's not clear that these techniques are scalable to handle video contents from all domains.

Instead, we ask a crucial question: can we learn to detect highlights by analyzing how users (with domain knowledge) edit videos? There is a wealth of edited video content on YouTube, along with the raw source material. This content captures highlights spanning a vast range of different activities and actions. Furthermore, we show that it's possible to identify the mapping of raw source material to edited highlights, leading to a wealth of training data. In this work, we introduce (1) a novel system to automatically harvest domain-specific information from Youtube, and (2) a novel latent ranking model which is trained with the harvested noisy data (see Fig. 2(a)). Leveraging the assumption that edited video is more likely to contain highlights than the trimmed parts of the raw video, we formulate the highlight selection problem as a pair-wise ranking problem between short video clips in the raw video. (see rank constraints in Fig. 2(b)). We introduce latent variables into the ranking model to accommodate variation of highlight selection across different users. For instance, user "A" might select a very long duration clip as a highlight, whereas user "B" prefers shorter clips (see Fig. 2(c)). We use a novel EM-like self-paced model selection procedure to learn the latent ranking model.

Our approach has several advantages. First, our latent ranking model consistently outperforms its classification counterpart (see Fig. 4). Second, it can be efficiently trained on a large number of videos by taking advantage of a newly developed solver [11] for linear ranking SVM. Third, our latent model nicely takes care of the noise in our automatically harvested training data (see Fig. 5). Finally, we demonstrate results using automatically harvested YouTube data that rival those obtained from a fully supervised ranking approach trained on specially constructed training sets commissioned on Amazon Mechanical Turk (see Fig. 6). Hence, we demonstrate state-of-the-art performance, while achieving much greater scalability (avoiding the need to manually construct new annotated datasets).

2 Related Work

Our work is highly related to video summarization. There is a large literature on video summarization (see review [1]), including techniques based on keyframes [13,3,17,15]. In the following, we focus on subjects most relevant to our work.

2.1 Content-aware video summarization

Many methods have been recently proposed for summarizing a video with a known type of content. Given an ego-centric video, we know hands, objects, and faces are important cues. [14,12] propose to summarize a video according to discovered interesting objects and faces. Similarly, given a video uploaded to ecommerce websites for selling cars and trucks, [8] propose to use web-image priors (i.e., canonical viewpoints of cars and trucks online) to select frames to summarize the video. Our method is another step in this direction. However, our proposed system is not restricted to handling only ego-centric videos, where cues from hands and objects are easier to extract, nor does it rely on discovered canonical viewpoints which are shown to have generalization issues in other domains such as cooking [8]. Whereas, our method is feature independent and we directly harvest information about how users select domain-specific highlights to create their own edited videos.

2.2 Sports video analysis

Highlight detection in broadcast sport videos has attracted several researchers [27,20,16,26,25,10,4,23] due to the popularity of such videos. Compared to other video types, such as personal videos, broadcast sports videos have well defined structure and rules. A long sports game often can be divided into parts and only a few of these parts contain certain well defined highlights. For example, common highlights are the score event in soccer games, the hit event in baseball games, and the “bucket” event in basketball games. Due to the well defined structure, specifically designed mid-level and high-level audio-visual features, such as player trajectories, crowds, audience cheering, goal or score events, etc., are used in many methods. One exception is [23] which uses easy-to-extract low-level visual features. Most of the methods treat highlight detection as a binary classification task, where each part of a training video is labeled as true highlight or not. We argue that for unconstrained personal videos, it is ambiguous for humans to label highlights as binary labels. This also imposes extra burden for annotators. Hence, it is important for a method to scale-up by naturally harvesting crowd-sourced information online.

2.3 Crowd-sourced highlight discovery

Recently, many researchers have demonstrated the ability to discover highlights from crowd-sourced data such as Twitter. Olsen et al. show that user interaction

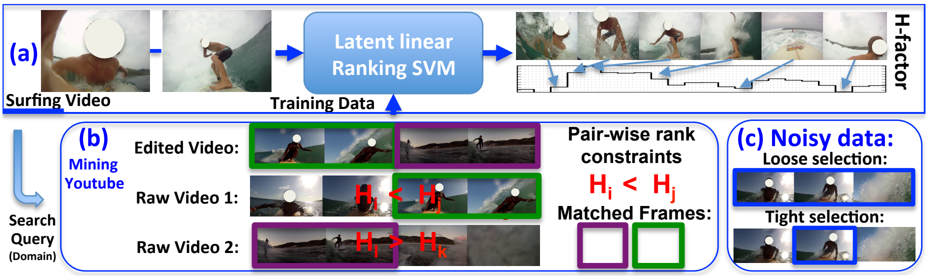

Fig. 2: System overview: Given "surfing" videos, we train a latent linear ranking SVM to predict the h-factors fully automatically (Panel (a)). Our system automatically harvests training data online by mining raw and edited videos in Youtube related to "surfing". The raw and edited pair of videos give us pair-wise rank constraints as shown in panel (b). Note that the harvested data is noisy due to the variation of highlights selected by the users on Youtube (Panel (c)).

data from an interactive TV application can be mined to detect events [18]. Hannon et al. use Twitter data in PASSEV [5], combining summaries of tweet frequency and user-specified search for terms in tweets to generate a highlight reel. The main difference between these approaches and others in computer vision is that crowd-sourced data (users' annotations but not the the video data) are always required as an input. Hence, these methods cannot work well on videos with no or very few associated crowd-sourced data. On the contrary, our method harvests both crowd-sourced and video data from Youtube for training our latent ranking model. After training, our model can be applied to rank highlights in any video.

2.4 Learning to rank

Learning to rank is an important learning technique in recent years, because of its application to search engines and online advertisement. In computer vision, researchers have adopted the ranking approach mainly for the image retrieval task [6,21]. Recently, Parikh and Grauman have introduced the concept of relative attributes [19] which opens up new opportunities to learn human-nameable visual attributes using a ranking function. In this work, we propose one of the first ranking models for domain-specific video highlight retrieval. Our domains include popular actions such as skating, surfing, etc.

3 Video Highlights

A video highlight is a moment (a very short video clip) of major or special interest in a video. More formally, a highlightness measure h (later referred to as "h-factor") for every moment in a video can be defined such that moments with high h are typically selected as highlights. Hence, the goal of retrieving highlights is equivalent to learning a function $f(x)$ to predict the h-factor (i.e., $h = f(x)$) given features x extracted from a moment in the video (see Fig. 1).

One straight forward way to learn $f(x)$ is to treat it as a supervised regression problem. However, labeling the h-factor consistently across many videos is hard for humans. On the contrary, it is much easier for humans to rank pairs of moments based on their highlightness. Hence, we propose to formulate the video highlights problem as a pair-wise ranking problem described next.

3.1 Pair-wise ranking

To establish notation, we use i as a unique index for all moments in a set of videos. Each moment is associated to a tuple (y, q, x), where $y \in R$ is the relative h-factor, $q \in Q$ is the index of the video containing the moment, Q is a set of videos, and x is the features extracted from the moment. The set of ranking constraints corresponding to the q^{th} video is defined as,

$$P_q \equiv \{(i,j)|q_i = q_j = q, y_i > y_j\} , \tag{1}$$

where $q_i = q_j = q$ ensures that only moments from the q^{th} video are compared to each other according to the relative h-factor y. Note that y only needs to be a relative value to express the order within each video. The full set of pair-wise ranking constraints in the dataset is defined as,

$$P \equiv \cup_{q\in Q} P_q = \{(i,j)|q_i = q_j, y_i > y_j\} , \tag{2}$$

where $q_i = q_j$ ensures that only moments from the same video are enforced with the ranking constraints.

Our goal is to learn a function $f(x)$ such that

$$f(x_i) > f(x_j),\ \forall (i,j) \in P , \tag{3}$$

which means none of the pair-wise ranking constraints is violated. We adopt the L2 regularization and L2 loss linear ranking SVM formulation to learn $f(x; w) = w^T x$ as follows,

$$\min_w \tfrac{1}{2} w^T w + \lambda \textstyle\sum_{(i,j)\in P} \max(0, 1 - w^T(x_i - x_j))^2 , \tag{4}$$

where w is the linear model parameter and $\lambda > 0$ is a regularization parameter. Ideally, the optimal model parameters can be learned since the optimization problem is convex. However, the large number of pair-wise constraints ($|P|$) becomes the main difficulty in training the ranking SVM efficiently. We solve the problem efficiently by taking advantage of a newly proposed fast truncated newton solver which employs order-statistic trees to avoid evaluating all the pairs [11]. Note that other non-linear ranking methods can also be used. However, they are typically impractical to train on a large-scale dataset such as our videos dataset (about 700 minutes long in all).

3.2 Ranking from edited videos

Asking humans to order moments within each video is feasible but not scalable. Since the definition of a highlight is highly dependent on the domain of interest, human annotators without domain knowledge will find the task ambiguous and tedious. We argue that we can naturally harvest such information from edited videos. Assume we have raw videos which are used to create the edited videos, and users make a (somewhat) informed decision to select the moments in the raw video to include in the edited video. We now get the following ranking constraints,

$$y_i > y_j, \;\; i \in E_q, j \in R_q \setminus E_q \;, \tag{5}$$

where q is the index of the raw video, E_q is the set of moments in the raw video which are included in the edited video, and R_q is the set of moments in the raw video. Eq. 5 states that the relative h-factors of moments included in the edited video (E_q) is higher than the rest of the moments in the raw video ($R_q \setminus E_q$) (see Fig. 2(b)).

Given many pairs of raw videos and their edited versions, the linear ranking SVM becomes

$$\min_w \tfrac{1}{2}\|w\|^2 + \lambda \textstyle\sum_{q \in Q} L(q; w) \;, \tag{6}$$

$$L(q; w) = \textstyle\sum_{i \in E_q} \sum_{j \in R_q \setminus E_q} \max(0, 1 - w^T(x_i - x_j))^2 \tag{7}$$

where Q is a set of raw videos, and $L(q; w)$ is the loss of violating ranking constraints in the q^{th} raw video. As a result, we can even extract ranking constraints for training data where users edit their videos by simply trimming it. The main advantage of this approach is that a wealth of such data exists in the digital world. We describe in Sec. 3.5 how we can harvest such data online automatically.

3.3 Handling noisy data

Not all the users online are "experts". The start and end of a specific highlight can vary significantly depending on the users. For example, user "A" might select a loose highlight with a few minutes included in its edited version. In contrast, user "B" might select a tight highlight with a few seconds (see Fig 2(c)). Especially for the loose highlight, constraints expressed in Eq. 5 might not always be true. In order to address this problem, we propose a latent loss as follows,

$$L_{z_q}(q; w) = \textstyle\sum_{j \in R_q \setminus E_q} \max(0, 1 - w^T(x_{z_q} - x_j))^2 \;, \tag{8}$$

$$z_q = \arg\max_{i \in E_q} f(x_i; w) \;, \tag{9}$$

where z_q is the best highlighted moment in E_q. Note that the latent loss relaxes the constraints in Eq. 5 to

$$y_{z_q} > y_j, \;\; j \in R_q \setminus E_q \;, \tag{10}$$

where only the the best highlighted moment z_q in E_q must have higher relative h-factor than the rest of the moment ($R_q \setminus E_q$) in the q^{th} raw video.

Latent linear ranking SVM A latent linear ranking SVM incorporating the latent loss is defined as,

$$\min_{w,Z} \tfrac{1}{2}\|w\|^2 + \lambda \left(\textstyle\sum_{q \in Q_T} L(q; w) + \sum_{q \in Q_L} L_{z_q}(q; w) \right) \;, \tag{11}$$

where Q_T is a set of raw videos with tightly selected highlights, Q_L is a set of raw videos with loosely selected highlights, $\{Q_T, Q_L\}$ is a partition of Q (i.e., $Q_T \cap Q_L = \emptyset$, $Q_T \cup Q_L = Q$), and $Z = \{z_q\}_{q \in Q_L}$ is a set of latent best highlighted moments in Q_L.

EM-like approach Given Q_T and Q_L, we solve the model parameters w and the set of latent best highlighted moments Z iteratively using a EM-like approach.

1. We set $Q = Q_T$ and obtain our initial w by solving Eq. 6.
2. Given the initial w, we estimate our initial Z by solving Eq. 9 (E-step).
3. Then, we obtain w by solving Eq. 11 while fixing Z (M-step).
4. Given w, we estimate Z by solving Eq. 9 (E-step).
5. Go to step 3 if the estimated latent variables Z have changed; otherwise, stop the procedure.

In our experiments, the EM-like approach stops typically within five iterations.

3.4 Self-paced model selection

Identifying the loosely selected videos Q_L for our latent ranking SVM model is important. On the one hand, SVM is known to be sensitive to inconsistent labels, since they are very likely to become strong distractors (supporting vectors) which affect the learned model significantly. On the other hand, we essentially throw away most of the potentially good highlighted moments in E_q by moving a video q from Q_T to Q_L. Hence, it is important to automatically find a good trade-off between Q_T and Q_L.

We propose a self-paced model selection procedure to evaluate K partitions of Q guided by pair-wise accuracy of every video $A = \{a_q\}_q$ as follows.

1. We set $Q_T = Q$ and obtain w by solving Eq. 11.
2. Given the model w, we evaluate the pair-wise accuracy A on the training videos.
3. Given A, we order the videos from low to high pair-wise accuracy, and evenly split the videos into K mutually exclusive sets.
4. Starting from the set with the lowest accuracy, we remove one set at a time from Q_T to Q_L[1].
5. For each partition of Q_T and Q_L, we solve Eq. 11 to obtain a new model w and new pair-wise accuracy A on the training videos.
6. Finally, we select the model with the highest mean pair-wise accuracy $mean(A)$.

The pair-wise accuracy a_q for the q^{th} video is defined as,

$$a_q = \frac{\sum_{(i,j)\in P_q} \mathbf{1}(w^T x_i > w^T x_j)}{|P_q|} \ , \tag{12}$$

where $\mathbf{1}(\cdot)$ is a indicator function which is one if a pair-wise constraint is satisfied in our learned model ($w^T x$), P_q (defined in Eq. 1) is the set of pair-wise constraints for the q^{th} video, and $|P_q|$ is the number of pairs. The pair-wise accuracy is a normalized value between 0 and 1. Hence, it is comparable for different pairs of Q_T and Q_L. Our results in Fig. 5 demonstrate that our novel latent linear ranking model can be effectively learned to achieve superior accuracy by our EM-like self-paced model selection procedure.

[1] Note that there are videos with highlights consisting of only one or two moments (a few seconds). We always keep these videos in Q_T

3.5 Harvesting Youtube videos

Nowadays, many videos are shared online through websites such as Youtube. We propose to mine these videos online to harvest a large amount of data. In particular, we query the Youtube database with popular search queries to retrieve relevant videos. Given the retrieved videos, we can use [7] to efficiently identify duplicated frames between every pair of videos. Once a set of consecutively duplicated frames are matched for a pair of videos (q_1, q_2), we define these matched frames as the selected highlighted moments in E. Then, we identify either q_1 or q_2 as the raw video and treat the moments in the raw video as R. Note that [7] is robust, but it can miss matches when the video includes after effects. Hence, our dataset is built with high precision.

Theoretically, what we have proposed is feasible even for a large scale database such as Youtube. In particular, Google is already using a similar procedure to find videos with copyright violations. However, this is a daunting task for individual researchers like us, since we need to retrieve a large number of videos to find enough matched pairs of edited and raw videos. Fortunately, Youtube has an online editor called "Youtube video editor", which keeps track of the information of raw and edited pairs of videos. Hence, we use the Youtube API to query videos generated by "Youtube video editor" to retrieve a smaller set of raw and edited pairs of videos. Then, we use [7] to efficiently identify the duplicated frames as the highlighted moments. Inevitably, our current data is limited by the availability of videos generated by "Youtube video editor". Nevertheless, our current system works amazingly well for common action and animal related domains such as "skating", "dog", "parkour", etc. We describe in Sec. 4.3 about the selected domains and the data statistics. We also believe that a much larger dataset can be easily built by accessing Google's internal infrastructure.

3.6 Comparison to binary highlight classification

Recall that previous highlight selection methods [27,20,16,26,25,10,4,23] formulate their problem as a binary highlight classification problem. Although we argue computing highlights is intrinsically a ranking problem, we show that our problem can also be considered as a binary classification problem by setting

$$\{y_i = +1; i \in E_q, q \in Q\} \;\; and \;\; \{y_j = -1; j \in R_q \setminus E_q, q \in Q\} \ , \tag{13}$$

where $y \in \{-1, +1\}$ becomes a binary label. Next we discuss the advantages and disadvantages of a binary classification problem.

The binary classification problem is highly related to a pair-wise ranking problem which ignores the video index and expresses the following constraints,

$$P \equiv \{(i, j); y_i > y_j\} \ . \tag{14}$$

In this case, moments are also compared to each other across videos. However, training a classification model is much more efficient than training a ranking model. This is because the huge number (i.e., quadratic to the number of moments) of pair-wise violation losses (Eq. 7) are replaced by losses with respect to a separation hyperplane as defined below,

$$L^C(q; w) = \textstyle\sum_{i; q_i = q} \max(0, 1 - y_i(w^T x_i)) \ . \tag{15}$$

The number of losses in Eq. 15 is linear to the number of moments. The advantages of a classification model is (1) it implicitly incorporates more pair-wise constraints in Eq. 14 than constraints in Eq. 2, and (2) it can be solved more efficiently using many existing methods. Nevertheless, the newly added pair-wise constraints which compare moments from two different videos (i.e., $\{(i,j); q_i \neq q_j, y_i > y_j\}$) could be harmful. For example, the highlight in video q_1 might be less interesting than many non-highlighted moments in video q_2.

We can also define a latent linear classification SVM to handle the noisy data by replacing $L(q;w)$ with $L^C(q;w)$ and $L_{z_q}(q;w)$ with $L^C_{z_q}(q;w)$ in Eq. 11, where $L^C_{z_q}(q;w)$ is defined as,

$$L^C_{z_q}(q;w) = \max(0, 1 - y_{z_q}(w^T x_{z_q}))^2 + \textstyle\sum_{j \in R_q \setminus E_q} max(0, 1 - y_j(w^T x_j))^2 \ , \\ z_q = \arg\max_{i \in E_q} f(x_i; w) \ . \qquad (16)$$

The latent linear binary classification SVM model can also be learned using our EM-like (Sec. 3.3) self-paced model selection (Sec. 3.4) procedure. In Fig. 4, we demonstrate that our proposed latent ranking model is consistently better than the latent classification model. This proves that the additional constraints in the latent classification model are indeed harmful.

4 Experiments

We conduct experiments on a newly created Youtube highlight dataset harvested by our system automatically. For analysis and evaluation purposes, we have labeled the dataset using Amazon Mechanical Turk. In the following sections, we first give details of our implementation such as feature representation, parameter setting, etc. Then, we report quantitative and qualitative results on our novel Youtube highlight dataset.

4.1 Implementation details

We describe our representation and training parameters in detail.

Moment definition We first define each moment as a 100 frames clip evenly sampled across each raw video. The start and end frames of each moment is then aligned to nearby shot boundaries within 50 frames away.

Feature representation Given these moments, we extract the state-of-the-art dense trajectory motion feature [24] which is best for action classification (other scene, objects, and audio features can also be included in our framework). Then, the dimension of the dense trajectory features is reduced by half using PCA. The dense trajectory features within each moment are mapped to a learned Gaussian mixture codebook to generate a fisher vector with fix dimension (26000). The Gaussian mixture model with 200 mixture components is learned using the training raw videos for each domain.

Highlight definition During training, a moment is considered as a highlight (in E) if at least 70% of its frames are matched in the edited video. A moment is not considered as a highlight (in $R \setminus E$) if at most 30% of its frames are matched in the edited video. All the remaining moments are not included in training.

Fig. 3: Statistics of our data harvested from Youtube.

	skating	gymnastics	surfing	dog	parkour	skiing
# Training videos	37	46	81	49	43	98
# Testing videos	37	47	82	50	43	99
Total # videos	74	93	163	99	86	197
Total seconds	9003	8294	15348	9000	11698	32457
% relevant videos	64.86%	77.42%	55.83%	49.49%	65.12%	49.23%

Model training All the regularization parameters λ in both the ranking and classification models are selected from 10 logarithmically spaced values from 0.001 to 10 to maximize the average pair-wise accuracy on the training data. We use the liblinear package [2] to train both models in their primal form with the same stopping criteria: maximum 10000 iterations, and $\epsilon = 0.0001$. For self-paced model selection, we set $K = 4$.

4.2 Evaluation details

For evaluating our ranking results, ideally we would like to obtain a h-factor order of all moments in every video. However, it is time consuming and ambiguous to ask general people to order all the moments since often it is hard to compare the h-factors of two random moments. Hence, we found that it is more effective to ask multiple people to select a single highlight (i.e., a segment of consecutive moments) within each video. We use Amazon Mechanical Turk to collect these ground truth annotations. For each video, we collect a less than five seconds highlight from each turker, where there are five turkers assigned to each video. Since we are crowd-sourcing these annotations from a large number of turkers, our annotations will inevitably be noisy. Hence, we only keep the moments selected more than 2 times[2] as ground truth highlights for evaluation. As a result, our problem becomes a highlight detection task.

Highlight detection Within each video, the best method should first detect the ground truth highlighted moments rather than other moments. We calculate the average precision of highlight detection for each testing video and report the mean average precision (mAP) summarizing the performance of all videos. Note that unlike object detection which accumulates all the detections from images to calculate the average precision, highlight detection treats each video separately since a highlighted moment in one video is not necessary more interesting than a non-highlighted moment in another video.

4.3 Youtube highlight dataset

By harvesting freely available data from Youtube as described in Sec. 3.5, we have collected data for "skating", "gymnastics", "dog", "parkour", "surfing", and "skiing". These 6 domains are selected because about 20% of the retrieved raw videos in these domains are publicly downloadable. For each domain, there is about 100 videos with various length. The total accumulated time is 1430 minutes, which is at the similar scale as the state-of-the-art large scale action

[2] For parkour and skiing, we found the turkers' annotation is noiser; hence, we only keep the moments selected more than 3 times.

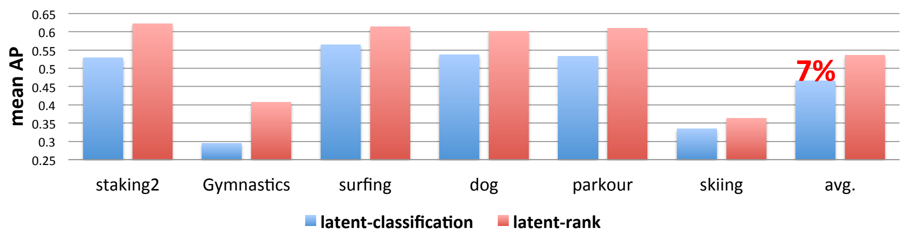

Fig. 4: Performance comparison between our latent ranking model and its classification counterpart. Latent ranking model is consistently better than latent classification model with an average improvement of $\sim 7\%$ in mAP.

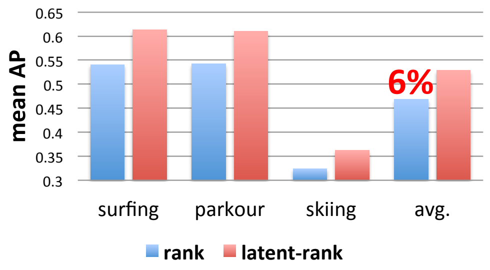

Fig. 5: Performance comparison between latent and non-latent ranking models. Our self-paced model selection procedure selects non-latent models for surfing, parkour, and skiing. In these 3 queries, our latent ranking model is consistently better with an average improvement of $\sim 6\%$ in mAP.

recognition dataset [9] (1600 minutes). Then, we split the data in half for training and testing. Our new dataset is very challenging since (1) it contains a variety of videos captured by portable devices, (2) the start and end of a specific highlight can vary significantly depending on the behavior of the users, and (3) there are irrelevant videos included in the dataset. For example, videos of interviews and slideshows of images are considered as irrelevant videos. Given all these challenges, our fully automatic system searches for the best latent configuration to limit the effects of the noise without human intervention. For testing, we evaluate on videos where turkers reach consensus on highlighted moments. The statistics of our collected data for each domain is shown in Fig. 3. The dataset and codes are available (see technical report [22] for details).

Our fully automatic system performs well on the novel Youtube highlight dataset with a mean average precision of 53.6% across six domains, which is significantly better than a motion analysis baseline (46%) [15]. We compare our system with other sophisticated methods below.

Latent ranking v.s. Classification In Fig. 4, we compare the mean average precision for every domain between our latent ranking model (Sec. 3.3) and a latent classification model (Sec. 3.6), where both models are trained using noisy data harvested from Youtube. Our latent ranking model consistently outperforms the the latent classification model in all domains with an average improvement of $\sim 7\%$ in mean average precision.

Latent v.s. Non-latent ranking As described in Sec. 3.4, there is a trade-off between how many videos should be considered as loosely selected videos Q_L and

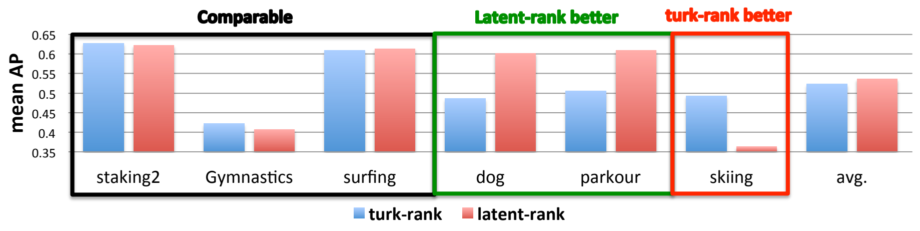

Fig. 6: Performance comparison between a fully supervised ranking model trained with turkers' annotations (referred to as "turk-ranking") and our latent ranking model trained with harvested data. Our model is very competitive compared to "turk-ranking", and suprisingly outperforms "turk-ranking" significantly on parkour and dog.

how many videos should be considered as tightly selected videos Q_T. Our self-paced model selection procedure selects latent models for surfing, parkour, and skiing[3]. We show in Fig. 5 that the latent model is consistently more accurate (larger mean average precision) then the non-latent model (Sec. 3.1), when both trained with noisy data harvested from Youtube. On average, the mean average precision of our latent ranking model is $\sim 6\%$ more than the non-latent ranking model.

Explicitly crowd-sourcing v.s. Naturally harvesting data In order to analyze the effect of our model trained using noisy data harvested online, we also trained a fully supervised ranking model (Sec. 3.1) with annotations from turkers. Note that the data is supposed to be less noisy since turkers have identified irrelevant videos and they are forced to annotate a highlight less than five seconds per video. However, as we have argued before, labeling highlights is really a time consuming and ambiguous task. Moreover, we suspect that labeling highlights in raw personal videos is not a well defined task for turkers unrelated to the events in the videos. The comparison between the fully supervised ranking model trained with raw turkers' annotations and our proposed latent ranking model is shown in Fig. 6. Intuitively, the fully supervised ranking model should perform much better. However, in some domains such as dog and parkour, our latent ranking model is even superior to the fully supervised ranking model trained with turkers' annotations. This proves that our fully automatic system is very competitive compared to the approach requiring annotations from a crowd-sourcing platform. Moreover, our system is not only more scalable, but also slightly more accurate on average.

Qualitative results We show a set of moments ranking from high to low according to our predicted "h-factors" for different domains in Fig. 7 (see supplementary materials for more results). Note that our data is challenging and realistic, since it includes videos captured by widely used cameras such as GoPro and cellphone cameras. Detailed visualization of our predicted h-factors and raw turkers' annotations, overlaid with manually sampled raw frames for each do-

[3] For skating, gymnastics, and dog, our self-paced model selection procedure selects non-latent models (i.e., $Q_L = \emptyset$)

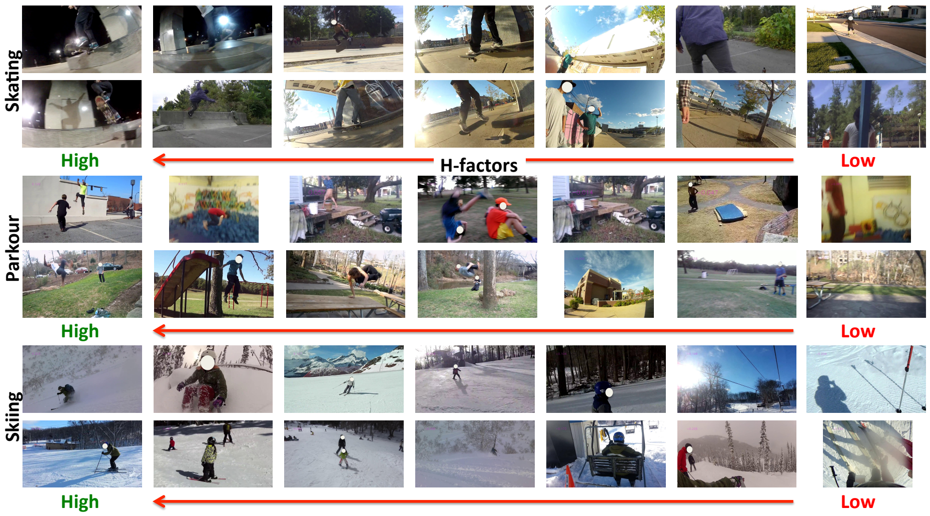

Fig. 7: Examples of moments ranking from high (left) to low (right) according to our predicted "h-factors" for skating, parkour, and skiing (see technical report [22] for more examples). We show one sampled frame from each moment to represent it.

main is shown in Fig. 8. Note that our predicted h-factors follow the raw turkers' annotations nicely.

5 Conclusion

As a step towards automatic video editing, we introduce a fully automatic system for ranking video highlights in unconstrained personal videos by analyzing online edited videos. Our system and the proposed novel latent ranking model are shown to be superior to its binary classification counterpart, a motion analysis baseline [15], a non-latent ranking model without handling noise in the harvested data, and a fully supervised ranking system requiring annotations from Amazon Mechanical Turk. We believe our system paves the way toward learning a large number of domain-specific highlights since more and more users' behavioral data can be harvested online. In the future, we would like to use a bank of domain-specific highlight rankers to describe every moment in a raw video. Given this high-level understanding of moments, we hope to automatically generate an edited video.

Acknowledgement We thank Microsoft, Google, Intel, NSF IIS-1218683, ONR N00014-13-1-0720, and ONR MURI N00014-10-1-0934 for supporting this research.

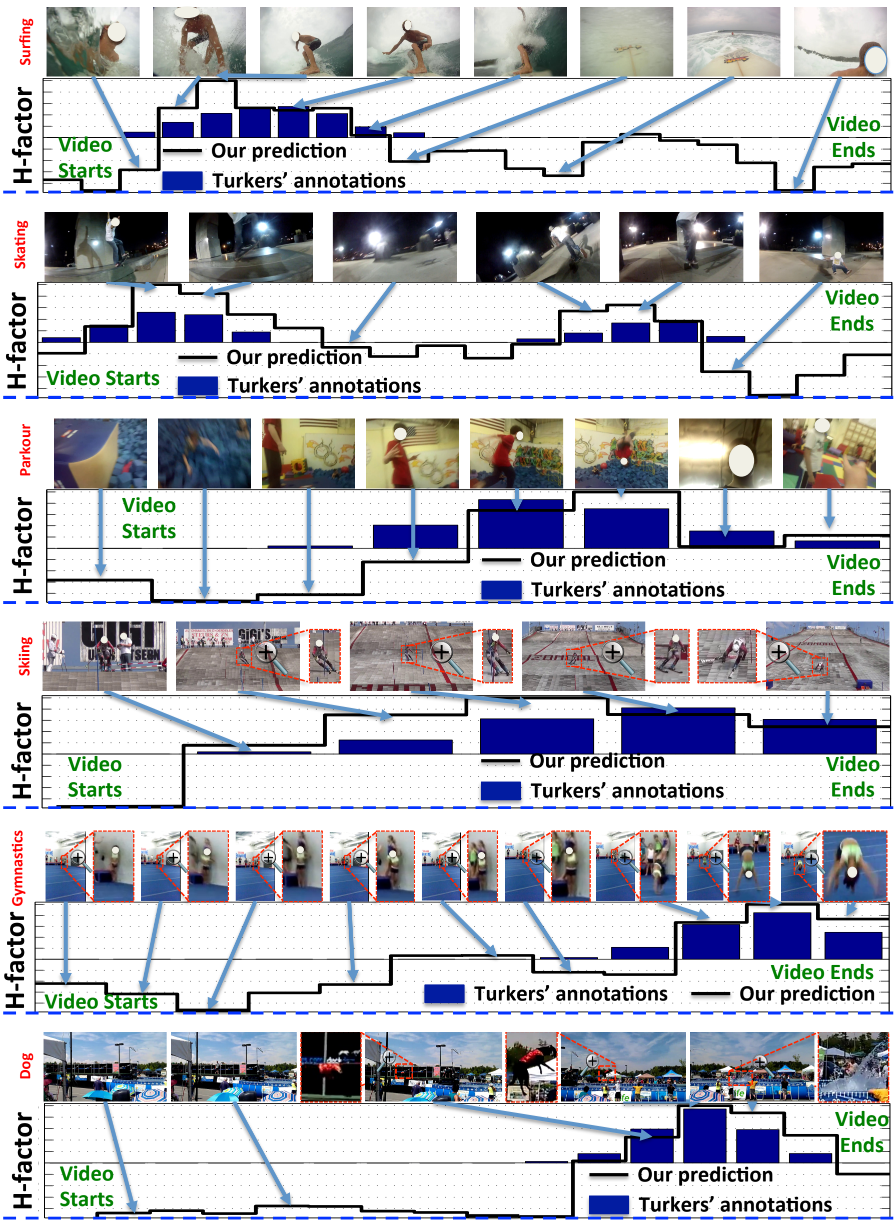

Fig. 8: Visualization of our predicted h-factors (black lines) and raw turkers' annotations (blue bars), overlaid with manually sampled raw frames. For gymnastics, dog, and skiing, the main characters in some frames are too small; hence, we show the manually cropped zoom-in version of the frames. Click to watch videos on Youtube: surfing link, skating link, parkour link, skiing link, Gymnastics link, dog link.

References

1. Borgo, R., Chen, M., Daubney, B., Grundy, E., Heidemann, G., Hoferlin, B., Hoferlin, M., Janicke, H., Weiskopf, D., Xie, X.: A survey on video-based graphics and video visualization. In: EUROGRAPHICS (2011)
2. Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: LIBLINEAR: A library for large linear classification. Journal of Machine Learning Research 9, 1871–1874 (2008)
3. Gong, Y., Liu, X.: Video summarization using singular value decomposition. In: CVPR (2000)
4. Hanjalic, A.: Adaptive extraction of highlights from a sport video based on excitement modeling. EEE Transactions on Multimedia (2005)
5. Hannon, J., McCarthy, K., Lynch, J., Smyth, B.: Personalized and automatic social summarization of events in video. In: IUI (2011)
6. Hu, Y., Li, M., Yu, N.: Multiple-instance ranking: Learning to rank images for image retrieval. In: CVPR (2008)
7. Jacobs, C.E., Finkelstein, A., Salesin, D.H.: Fast multiresolution image querying. In: SIGGRAPH (1995)
8. Khosla, A., Hamid, R., Lin, C.J., Sundaresan, N.: Large-scale video summarization using web-image priors. In: CVPR (2013)
9. Khurram Soomro, A.R.Z., Shah, M.: Ucf101: A dataset of 101 human action classes from videos in the wild. In: CRCV-TR (2013)
10. Kolekar, M., Sengupta, S.: Event-importance based customized and automatic cricket highlight generation. In: ICME (2006)
11. Lee, C.P., Lin, C.J.: Large-scale linear ranksvm. In: Neural Computation (2013)
12. Lee, Y.J., Ghosh, J., Grauman, K.: Discovering important people and objects for egocentric video summarization. In: CVPR (2012)
13. Liu, D., Hua, G., Chen, T.: A hierarchical visual model for video object summarization. TPAMI (2010)
14. Lu, Z., Grauman, K.: Story-driven summarization for egocentric video. In: CVPR (2013)
15. Mendi, E., Clemente, H.B., Bayrak, C.: Sports video summarization based on motion analysis. Computers and Electrical Engineering 39(3), 790 – 796 (2013)
16. Nepal, S., Srinivasan, U., Reynolds, G.: Automatic detection of goal segments in basketball videos. In: ACM Multimedia (2001)
17. Ngo, C., Ma, Y., Zhan, H.: Video summarization and scene detection by graph modeling. In: CSVT (2005)
18. Olsen, D.R., Moon, B.: Video summarization based on user interaction. In: EuroITV (2011)
19. Parikh, D., Grauman, K.: Relative attributes. In: ICCV (2011)
20. Rui, Y., Gupta, A., Acero, A.: Automatically extracting highlights for tv baseball programs. In: ACM Multimedia (2000)
21. Siddiquie, B., Feris, R., Davis, L.: Image ranking and retrieval based on multi-attribute queries. In: CVPR (2011)
22. Sun, M., Farhadi, A., Seitz, S.: Technical report of ranking domain-specific highlights. `http://homes.cs.washington.edu/~sunmin/projects/at-a-glace/`
23. Tang, H., Kwatra, V., Sargin, M., Gargi, U.: Detecting highlights in sports videos: Cricket as a test case. In: ICME (2011)
24. Wang, H., Kläser, A., Schmid, C., Liu, C.L.: Action Recognition by Dense Trajectories. In: CVPR (2011)

25. Xiong, Z., Radhakrishnan, R., Divakaran, A., Huang, T.: Highlights extraction from sports video based on an audio-visual marker detection framework. In: ICME (2005)
26. J. Wang andC. Xu, E.C., Tian, Q.: Sports highlight detection from keyword sequences using hmm. In: ICME (2004)
27. Yow, D., Yeo, B., Yeung, M., Liu, B.: Analysis and presentation of soccer highlights from digital video. In: ACCV (1995)