Screencast Tutorial Video Understanding

Kunpeng Li[1], Chen Fang[2], Zhaowen Wang[2], Seokhwan Kim[2], Hailin Jin[2] and Yun Fu[1]
[1]Northeastern University, [2]Adobe Research

Abstract

Screencast tutorials are videos created by people to teach how to use software applications or demonstrate procedures for accomplishing tasks. It is very popular for both novice and experienced users to learn new skills, compared to other tutorial media such as text, because of the visual guidance and the ease of understanding. In this paper, we propose visual understanding of screencast tutorials as a new research problem to the computer vision community. We collect a new dataset of Adobe Photoshop video tutorials and annotate it with both low-level and high-level semantic labels. We introduce a bottom-up pipeline to understand Photoshop video tutorials. We leverage state-of-the-art object detection algorithms with domain specific visual cues to detect important events in a video tutorial and segment it into clips according to the detected events. We propose a visual cue reasoning algorithm for two high-level tasks: video retrieval and video captioning. We conduct extensive evaluations of the proposed pipeline. Experimental results show that it is effective in terms of understanding video tutorials. We believe our work will serves as a starting point for future research on this important application domain of video understanding.

1. Introduction

Video is the dominant modality which people use to consume content and share experiences, thanks to ubiquitous personal computing devices, mobile cameras, video sharing websites, and social media. Among all the different types of video, screencast tutorial video is of special interests. It is a video screen capture created and edited by a human educator for the purpose of teaching the use of a software feature or demonstrating a procedure for solving a problem. In some cases, screen capture videos may be augmented with audio narrations and captions to make it easy for people to understand. Screencast tutorial video is becoming increasingly popular for both novice and experienced users to learn new skills and therefore for people to create and share. For instance, Youtube, one of the many video sharing sites, offers over 45 million videos related to Adobe Photoshop and over 12 million videos related to Microsoft Word. Algorithms for understanding screencast

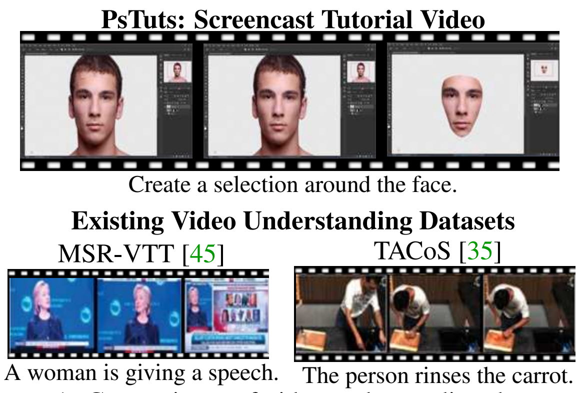

PsTuts: Screencast Tutorial Video
Create a selection around the face.
Existing Video Understanding Datasets
MSR-VTT [45]
TACoS [35]
A woman is giving a speech.
The person rinses the carrot.

Figure 1: Comparisons of video understanding data examples. Our PsTuts dataset (top) focuses on screencast tutorial video with significantly different aspects and challenges from existing datasets (bottom) for general video understanding problems [4, 35, 42, 45].

tutorials can enable many interesting applications ranging from video retrieval and recommendation based on content analysis to video synthesis for new tutorial content. Furthermore, they can help extract expert knowledge embedded in screencast tutorials and build artificial intelligence systems to perform complex tasks.

In this work, we consider the problem of understanding screencast tutorial video. Our ultimate goal is that given a video tutorial which typically ranges between 5 to 20 minutes long and may or may not contain a audio narration or a caption, we want to understand the actions the user performed, the specific tools and workflow she used, and the eventual goal she accomplished. We are interested in tasks such as dividing a video into semantic segments (temporal video segmentation), summarizing a video into a text description (video captioning) and retrieving related videos from a database given a text query (video retrieval). To the best of our knowledge, this is the first time the screencast tutorial video understanding problem is considered in the Computer Vision community. To facilitate research activities in this area, we collect a large-scale screencast tutorial dataset on Adobe Photoshop and will share the dataset with the community. We focus on Adobe Photoshop for two reasons. First, Adobe Photoshop is one of the most popular software. It is used not just for image editing, but for all kinds of graphic designs, even web and mobile designs. Second, it is the software with rich and diverse video tutorial content. As we discuss in Section 7, algorithms for understanding Photoshop tutorials can be extended to other applications and it is an important first step toward understanding general screencast tutorial video.

Understanding screencast tutorial video is a highly challenging task. For a modern software application such as Adobe Photoshop, there are virtually an unlimited number of ways in which a user can interact with the application. For instance, a user can invoke a command (action) through the application menu, the context menu, the tool bar, or buttons in various panels and windows. Each interaction has a very different visual pattern. One may think of tracking the mouse cursor and using the location information to assist the analysis, but it turns out that mouse cursor tracking is not a solved problem due to the small object size and fast movement (easily tens of pixels in adjacent frames). There are also "global" appearance changes such as canvas movement and opening of another application that may challenge an algorithm. Finally, there are potential mouse and keyboard effects which are purely "ornament" but can easily degrade the performance of a machine learning algorithm.

To address these challenges, in this paper we propose a two-stage bottom-up pipeline to understand screencast video. In the first stage, i.e., the low-level stage, we temporally segment a video into short clips based on the occurrences of a set of low-level operations, which are often correlated to important state changes in a software. In the case of Photoshop (and a lot of other software as well), we consider three types of operations: select a tool, operate a pop-up window, and operate a pop-up panel. To capture these events, we build computer vision modules to detect and recognize relevant visual cues. A video is segmented at the boundaries of these operations. In the second stage, i.e., the high-level stage, the goal is to learn a correlation between low-level signals and high-level intentions. To this end, we design a scalable data collection and annotation pipeline to crowdsource natural language descriptions from Photoshop experts. The data we collect supports a range of research explorations in this problem. As shown in Figure 1, the content and text description within our dataset are very different from existing common video datasets [4, 35, 42, 45], which further demonstrates the uniqueness of our target problem. To better encode these unique content, we further propose a visual cue reasoning model that considers the correlations between different visual cues and video frames. It can be embedded into existing retrieval and video captioning methods so that they are more applicable for screencast tutorials.

Main contributions of our paper are as follows: 1. We propose visual understanding of screencast tutorial video as a new research problem for the computer vision community to explore. It is a challenging video understanding problem but of significant practical importance. 2. We collect a new screencast tutorial video dataset on Adobe Photoshop. The dataset contains both low-level and high-level human annotations that support a range of research explorations in this area. 3. We propose a novel pipeline to understand Photoshop tutorial video that combines low-level temporal video segmentation and high-level text-to-video retrieval and video captioning. 4. We propose a visual cue reasoning model for text-to-video retrieval and video captioning tasks on this new dataset, which proves that the whole pipeline takes a meaningful step towards understanding screencast tutorials.

2. Related Work

Video understanding is one of the most important topics in Computer Vision. We make no attempt to review all the related works in this paper. Instead, we reference the works related to our methods in the individual sections and focus our discussion on video datasets and tutorial understanding. Large-scale labeled video datasets are one of the main driving forces behind recent developments in video understanding. It is still an active research area where new datasets are being introduced. HMDB [23] and UCF [39] are among the early video datasets on understanding general human actions and they serve the community well. They are followed by Sports-1M [18] and ActivityNet [4] and more recently by Kinetics [19] and Moments [31] with many more videos and more action classes. These datasets are collected starting from a list of labels and searching on video sites for related videos. A different approach is taken in [13] where videos are collected before being analyzed and annotated. Our data collection methodology follows the latter approach and focuses on screencast tutorials. Tutorial or instruction video understanding is not a new problem in the domain of cooking, where several related datasets have been collected including [2, 9, 30, 35, 37, 40, 41]. Our work differs from all aforementioned ones in that it focuses on screencast tutorials which are very different from videos filmed by cameras (see Figure 1 for examples). They pose unique challenges and at the same time offer new research opportunities.

Existing works in screencast tutorials are mainly in the HCI literature [7, 14–16, 20, 33] and the multimedia community [11, 24, 47]. [24, 47] focus on the system design and cover research in capture systems, content analysis and content delivery, so that lecture videos can be effectively accessed by users in an e-learning system. [20] proposes an interactive video player that effectively displays step descriptions and intermediate thumbnails in the video timeline. [15] extracts video clips from live-streamed software tutorials with software log data and recommends clips based on user behavior. These HCI works in screencast tutorials focus on enhancing the learning experience (system and interface design) and require key content information from

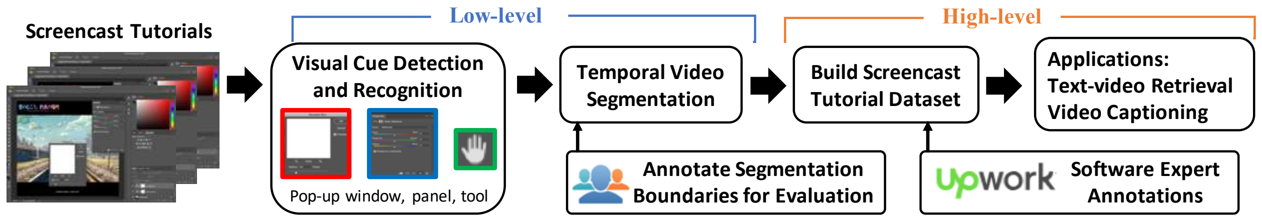

Screencast Tutorials
Low-level
High-level
Visual Cue Detection and Recognition
Pop-up window, panel, tool
Temporal Video Segmentation
Build Screencast Tutorial Dataset
Applications: Text-video Retrieval Video Captioning
Annotate Segmentation Boundaries for Evaluation
Upwork
Software Expert Annotations

Figure 2: Overview of the proposed two-stage screencast tutorial understanding pipeline. The first low-level stage aims at segmenting a video into short clips, each of which corresponds to an atomic software operation. The high-level semantics of each clip is further analyzed in the second stage for downstream use cases including video clip retrieval and video captioning using learned models based on labeled data collected through crowdsourcing.

external data (e.g. software log data). This significantly limits the applicability to the vast amount of existing screencast tutorials, such as those on Youtube. In comparison, our work focuses on visual understanding of tutorial video and does not require external data.

3. A Bottom-Up Understanding Pipeline

Screencast tutorials often consist of user actions with various low-level operations in a software, such as using different tools, windows, panels, etc. These low-level operations together form a complex workflow for accomplishing a high-level goal. For a professional software like Adobe Photoshop, there are many low-level operations as well as high-level tasks, which makes parsing and understanding challenging. We propose a two-stage bottom-up pipeline to deal with screencast tutorials, as shown in Figure 2.

Low-level Stage. We start by defining a set of low-level operations that are informative for inferring high-level tasks, which usually indicate important state changes when using a software. As shown in Figure 2, in the case of Photoshop, the low-level operations fall into three major categories: selecting a tool, operating on a pop-up window, and operating on a pop-up panel. We build computer vision modules to detect and recognize corresponding visual cues to capture these low-level events.

With the detected low-level events, we further temporally partition a video into segments, where the boundary of each segment aligns with the occurrence of an event. Note that, the design of our low-level pipeline can be generalized to other software with a customized definition of visual cues. The details of this level-low understanding stage are described in Section 5.

High-level Stage. Given segments from the previous stage, our ultimate goal is to understand the high-level task being performed in each segment as well as the entire video. This work focuses on understanding the semantics in segmented video clips, which aligns with the setting of most existing video understanding work. In order to achieve this, we collect a large-scale tutorial video dataset with high-quality annotations from design experts, and train various models that can automatically parse new videos. For each video segment, we ask a professional Photoshop user to annotate the corresponding task/intention. We design the annotation task to be scalable and cost-effective, so that a large amount of human labels can be collected from crowdsourcing platforms such as Upwork [1]. With the collected dataset, we build machine learning models for two major video understanding applications: text-to-video retrieval and video captioning. The data collection and model training are detailed in Sections 4 and 5, respectively.

4. The Photoshop Tutorial Video Dataset

In this section, we describe the details of our data collection and annotation procedures for Photoshop tutorial videos. The data to collect should support the study of both low-level and high-level tasks as shown in Figure 2. Our dataset construction is designed to be general enough to be applied on other software with complex workflow.

Obtain high-quality tutorial videos. We start with collecting a large set of tutorial videos from YouTube by searching queries containing keywords such as "Photoshop" and "tutorial." This results in an extremely large number of diverse videos with varying quality. In order to obtain a high-quality dataset with a manageable scale, we only keep videos that are uploaded after the year of 2012, longer than 3 minutes, featuring HD resolution (1280$\times$720 or higher), and watched at least 1000 times. This filtering step removes the majority of irrelevant videos, but some hard negatives still remain. For instance, a recording of a Photoshop lecture where a teacher gives a verbal lesson without showing the software screen. In order to deal with this challenge and obtain tutorial videos with screen shots of the target software as the main contents, We design a semi-automatic filtering approach by leveraging the visual cue detection and recognition modules described in Section 5. These modules are trained on a small set of human-annotated screencast videos, and can reliably indicate the presence of a Photoshop screen in each frame of any unlabeled video. We eliminate all the videos with a small percentage of frames detected to contain these visual cues. Additionally, we remove videos with very few tool or action changes.

Low-level annotation for temporal segmentation. Annotations for temporal segmentation boundaries are needed to verify the proposed temporal segmentation method in

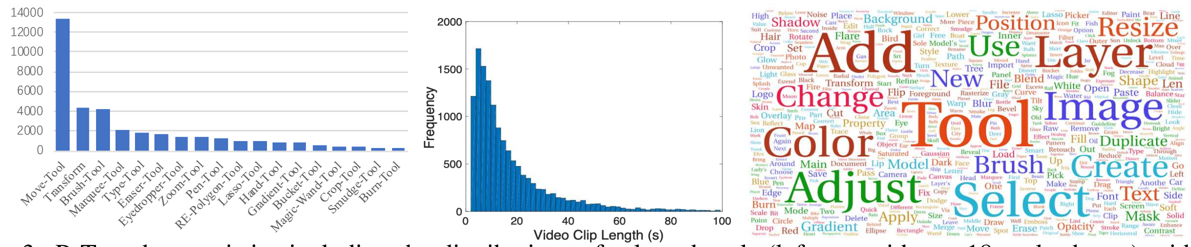


Figure 3: PsTuts data statistics including the distributions of selected tools (left one with top 18 tools shown), video clip lengths (middle) and word frequencies (right). The high frequency words are software-specific.

the low-level processing stage. The ground truth annotations indicate the frames with low-level visual cue changes: changing of selected tool, opening and closing of pop-up windows and panels. However, it is too expensive to exclusively annotate all the segmentation boundaries which requires careful inspection of each frame. Again, we use a semi-automated approach leveraging the visual cue detection modules described in Section 5. We run visual cue detection on all the frames and select a subset of frames whose detection scores of any cue change significantly from the adjacent frames. These detected frames together with their adjacent frames are further reviewed by Amazon Mechanical Turk workers to determine whether there is a corresponding visual cue change and where is exactly the frame showing the change. We take the consensus of multiple workers on each segment as the final annotation, and verify the labelling quality of each worker with a small number of data with known labels. More details are in Section 6.1.

High-level annotation for semantics. Given the segmented tutorial clips, we are interested in understanding the goal or obtaining the high-level description. Expert knowledge is needed in this case to acquire reliable annotations. We employ expert Photoshop users from Upwork and build a web-based data annotation tool particularly for this task. This tool loads each video in a web page and displays the video as temporal segments detected in the low-level step. More details about this annotation interface are included in the supplementary material. Annotators are required to provide concise text descriptions and key words for each segment. In practice, we find this combination of web-based labeling tools and online expert annotator hiring to be scalable and cost-effective. In the end, we are able to label 13,856 video segments within 3 weeks with 4K dollars.

Dataset Statistics. Our collected Photoshop Video Tutorial dataset, abbreviated as PsTuts, is the first screencast video dataset with high-level task annotations to the best of our knowledge. In Table 1, we compare PsTuts with a few standard video recognition datasets as well as existing video tutorial datasets for other domains, including MPII Cooking [37], YouCook [9], TACoS [35]. Our dataset is the only one focusing on screencast video of software and has a comparable scale compared to other datasets.

We show a summary of statistics of the PsTuts dataset in Figure 3. The frequency of most commonly selected tools

Dataset	Domain	No. of clips	No. of Sentences	Length (Hours)
Scope: general videos				
MSVD [5]	Open	1970	70,028	5.3
MSR-VTT [45]	Open	10,000	200,000	41.2
ActivityNet [4]	Open	20,000	100,000	849.0
MPII-MD [46]	Movie	68,337	68,375	73.60
M-VAD [42]	Movie	48,986	55,904	84.60
Scope: tutorial purpose				
MPII-Cook [37]	Cooking	44	5,609	8
TACoS [35]	Cooking	7,206	18,227	15.9
YouCook2 [48]	Cooking	14,000	14,000	176
HowTo100M [30]	Life tasks	136M	136M	0.13M
Coin [41]	Life tasks	11,827	46,354	476.6
PsTuts (ours)	Software	13,856	13,856	71.4

Table 1: Comparisons of video understanding datasets. M denotes million. Our PsTuts is the only dataset focusing on screencast tutorials.

are shown on the left. Screencast tutorials from the web show a heavily unbalanced distribution over all the tools. The distribution of video clip length in our dataset is shown in the middle. Most clips have length between 2 to 30 seconds. The right of Figure 3 shows the word cloud of text descriptions in our dataset, where we can see the most commonly used words are quite different from those used for general videos [4, 36, 45].

5. Method

Visual cue detection and recognition. First, we apply an object detection algorithm to localize the low-level visual cues for tool icons and pop-up windows/panels on each frame of Photoshop tutorial videos. Considering the importance of context information in this problem, we use YOLO [34] instead of proposal-based detectors. To reduce labeling effort, we take an active learning strategy which starts with a small number of labeled examples and then test it on an unlabeled data set. Only the instances with low confidence scores are annotated and merged into the previous training set. As we repeat retraining the model with the combined dataset, the robustness of the model gradually improves and eventually converges. The total number of training frames through the iterations was only 1000 for each visual cue.

After localizing the visual cues, we recognize the selected tool on each frame to detect tool changes. To build a tool classifier, we collected training samples for all the tool

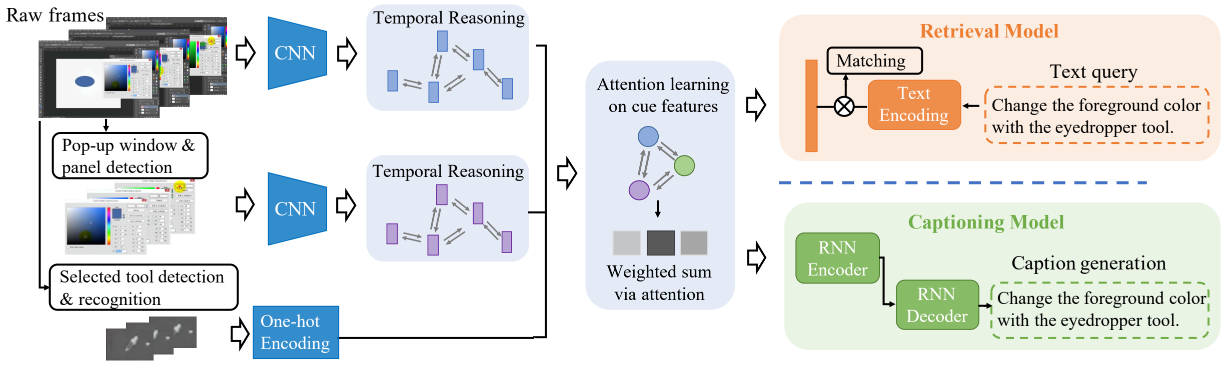

Raw frames
Pop-up window & panel detection
Selected tool detection & recognition
CNN
CNN
One-hot Encoding
Temporal Reasoning
Temporal Reasoning
Attention learning on cue features
Weighted sum via attention
Retrieval Model
Matching
Text Encoding
Text query
Change the foreground color with the eyedropper tool.
Captioning Model
RNN Encoder
RNN Decoder
Caption generation
Change the foreground color with the eyedropper tool.

Figure 4: The general structure of our visual cue reasoning (VCR) method for text-to-video retrieval and tutorial video captioning. The tutorial encoding is generated considering correlations between different visual cues as well as video frames.

Algorithm 1 Temporal video segmentation on visual cues.

Input: Visual cue detection (for pop-up panels/windows) and recognition (for tools) results $\{r_1, r_2, ..., r_n\}$ with confidence scores $\{s_1, s_2, ..., s_n\}$ for each of the n frames.
Output: Segmentation boundary set B.

```
1: for frame $i = k+1$ to $n-k-1$ do
2:     $\hat{r}_i$ = mode of $\{r_{i-k}, ..., r_i, ..., r_{i+k}\}$;
3: Initialize $L = 0$, $B = \{\}$
4: for frame $i = 2$ to $n$ do
5:     $L$+=1
6:     if $\hat{r}_i \neq \hat{r}_{i-1}$ and $s_i > \varepsilon$ and $L > \sigma$ then
7:         $B = B \cup i$ and $L = 0$
```

icon images in Photoshop and augmented them with small perturbations such as shifting and scaling towards simulating real screencast videos. The dataset consists of 60K and 20K images for training and validation purposes, respectively, for 39 tool classes in total. We trained a Resnet-101 model on this dataset with cross-entropy loss and achieved 99.83% validation accuracy in tool recognition.

Temporal video segmentation. A whole video is segmented into a sequence of clips according to the temporal consistency and the reliability of the visual cues. Algorithm 1 describes our segmentation algorithm based on the detected pop-up panels/windows and the recognized tools. We first smooth the detection and recognition results $\{r_i\}$ with a temporal window of length $2k$ to filter out outlier results. The final segmentation boundary set B is selected from the change points of the smoothed recognition results with minimal confidence score ε and minimal segment length σ. We set $k = 2$, $\varepsilon = 8$, $\sigma = 60$ in our experiments.

Visual Cue Reasoning (VCR) for High-level Tutorial Understanding. For high-level tutorial understanding tasks, we focus on tutorial clip retrievals for given text queries and caption generation for given tutorial clips. We build a compound feature descriptor based on visual cue reasoning to represent tutorial video clips. As shown in Fig. 4, besides the generic CNN feature of raw video frames V, the following visual cues are encoded in the video descriptor: the CNN features from the local regions of pop-up windows or panels; and the one-hot vector of selected tool category. These cues are important to characterize the user state in each video clip. Each of them is transformed to a D-dimensional embedding space via fully connected layers.

Considering the semantic correlations between frames, we do temporal reasoning on these cue features across frames. Specifically, for each kind of visual cue, we measure the pairwise affinity between features of different frames $F = \{f_1, ..., f_n\}, f_i \in \mathbb{R}^D$ in an embedding space to obtain the correlations via Eq. 1.

$$A(f_i, f_j) = e^{\varphi(f_i)^T \phi(f_j)}, \tag{1}$$

where $\varphi(f_i) = W_\varphi v_i$ and $\phi(f_j) = W_\phi f_j$ are two mapping functions. Parameters W_φ and W_ϕ can be learned via back propagation.

Then we construct a fully-connected graph $G = (F, A)$, where F acts as the node set and affinity matrix A acts as the edge set. Graph Convolutional Networks (GCN) [22] are applied to perform reasoning on this fully-connected graph. Each node's response is obtained according to its neighbors defined by the graph relations. Residual connections [17, 27, 38] are applied to the GCN as in Eq. 2.

$$F_i^* = F_i + (AFW_g)W_r[i, :], \tag{2}$$

where W_r is the parameter of the residual structure and W_g is the weight matrix of the GCN layer. The output $F^* = \{f_1^*, ..., f_n^*\}, f_i^* \in \mathbb{R}^D$ is the relationship enhanced representation. We learn such a residual-GCN model for each visual cue c except for cue T because the selected tool is the same for different frames within each video clip.

We further generate the final representation for tutorial videos considering the correlations between different visual cues. This is achieved by attention learning on different cue representations. For each visual cue c, we do average-pooling across frames on F_c^* (on F for cue T) to get cue representation R_c. Then we build up a fully-connected graph, where cue representation set R acts as the node set and its affinity matrix acts as the edge set. We can perform similar process as Eq. 1 and 2 on these cue representations to

Visual cue used	AMT quality	Precision	Recall
Pop-up window	0.94	0.90	0.96
Pop-up panel	0.92	0.86	0.91
Selected tool	0.95	0.89	0.94

Table 2: Quantitative evaluation of the temporal segmentation. AMT quality is ensured by randomly including samples that already have ground-truth labels. Precision reflects the true positive rate and recall measures how likely the model may miss ground-truth segmentation boundaries.

get R^*. After that, cue attention weights w_c can be learned from relationship enhanced cue representations R^* through a fully-connect layer. They are used to weighted combine the corresponding R_c to get the final video representation V, where $V = \sum w_c R_c$.

Text-to-tutorial clip retrieval. We follow common approaches for cross-modal retrieval which learn a similarity metric for two heterogeneous embedding spaces. We replace the visual embedding with the video representation V obtained by our visual cue reasoning method, so that they are more applicable for tutorial video content. Specifically, we follow [12, 25] to use a GRU [8] based text encoder to map the text query to the same D-dimensional semantic vector space $\mathbb{R}^D$ as V. A improved hinge-based triplet ranking loss with emphasis on hard negatives is then adopted to train this matching space [12, 25].

Tutorial clip captioning. In order to generate text description for a given tutorial video clip, we build upon the sequence to sequence model S2VT [44] that accommodates both input frame sequence and output word sequence of variable lengths. We use our VCR method to get improved visual representation for S2VT model. To keep the sequence structure, cue attention weights are used to weighted combine the corresponding frame feature F_c^* for each cue c instead of whole cue representation R_c to get frame-level representations V, where $V = \sum w_c F_c^*$. They are then fed into a staked GRU with two layers to encode the input video clip followed by decoding the output sentence describing the video content. We adopt a neural attention mechanism [3, 26] which allows the model to focus on certain parts of the encoded features during the decoding phase.

6. Experiments

In this part, we first design experiments to quantitatively evaluate the performance of the proposed visual-cue-based temporal video segmentation method. Then, we conduct text-to-tutorial video retrieval and tutorial video captioning experiments on our PsTuts Dataset.

6.1. Temporal Video Segmentation

We randomly select 800 video samples from Photoshop tutorial videos obtained from the Youtube as described in Section 4, and run our temporal video segmentation model on these videos based on each of the proposed visual cue.

Method	R@1	R@5	R@10	MedR
MEE (V) [29]	6.7	18.1	26.9	46
SCAN (V) [25]	8.7	21.2	27.6	50
VSE++ (V) [12]	9.0	20.7	27.8	61
Ours-S (V)	11.8	25.6	33.3	42
Ours (V)	12.4	26.9	34.4	34
P+T MEE	0.2	1.4	3.1	165
P+T SCAN	1.9	8.7	14.2	168
P+T VSE++	2.9	10.5	16.5	138
Ours (P+T)	4.3	12.4	19.2	85
V+T MEE	6.9	19.1	27.3	43
V+T SCAN	9.2	21.8	31.2	28
V+T VSE++	9.5	23.5	32.7	26
Ours (V+T)	13.4	28.2	35.7	22
V+P MEE	8.8	24.4	33.5	29
V+P SCAN	10.2	26.8	36.1	25
V+P VSE++	12.8	29.5	38.2	23
Ours (V+P)	14.5	31.8	41.2	21
V+P+T MEE	9.4	25.8	34.7	26
V+P+T SCAN	11.5	29.1	38.2	23
V+P+T VSE++	14.2	31.3	40.4	22
Ours-S (V+P+T)	15.5	33.4	42.0	18
Ours (V+P+T)	**17.0**	**34.9**	**43.8**	**15**

Table 3: Quantitative evaluation results of the text-to-video clip retrieval performances on the test set of PsTuts dataset interms of Recall@k (R@k) and median rank (MedR).

In order to evaluate detection recall with limited manual annotation, we also generate a negative set for each visual cue with the most likely segmentation boundaries that our model may miss ($s_i < \varepsilon$ or $L < \sigma$ as in Algorithm 1).

Following the annotation and evaluation method described in Section 4, to construct an evaluation set for the segmentation results, we randomly select 1.8K positive segmentation boundaries detected by our model as well as 1.8K most likely negative segmentation boundaries that our model may miss. We employ AMT Turkers to annotate whether there is a corresponding visual cue change at each of the segment boundary by reviewing the adjacent frames before and after the boundary. To measure Turkers' labeling quality, we also include 400 additional segmentation boundaries that are manually annotated by the authors with high confidence. The entire 4K segments are presented in random order for multiple rounds. The quantitative evaluation results in Table 2 show that our method can guarantee a decent segmentation quality that supports further high-level understanding tasks. Our final segmentation boundaries used for high-level tasks are obtained by combining the segmentation results from all the three visual cues.

6.2. Text-to-tutorial Clip Retrieval

The entire PsTuts dataset is split into training, validation and testing with a ratio of 8:1:1. This results in 11086 samples for training, 1385 samples for validation and 1385 samples for testing. Following LSMDC dataset [36], we evaluated models by considering every video segment as the ground-truth for the corresponding text query. The performance of text-to-video retrieval is measured using Recall@ 1/5/10 and Median Rank (MedR). Recall@k (higher is better) means the percentage of the ground truth video included

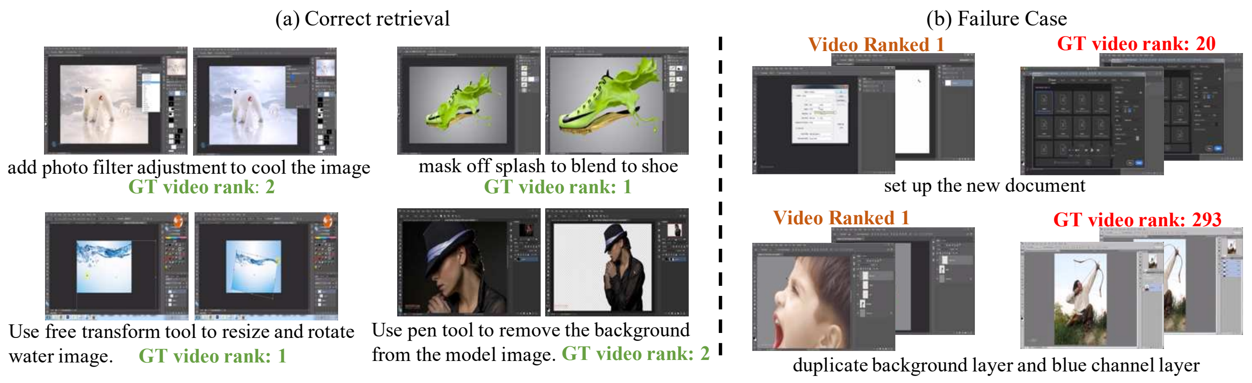

(a) Correct retrieval
add photo filter adjustment to cool the image
GT video rank: 2
mask off splash to blend to shoe
GT video rank: 1
Use free transform tool to resize and rotate water image.
GT video rank: 1
Use pen tool to remove the background from the model image.
GT video rank: 2
(b) Failure Case
Video Ranked 1
GT video rank: 20
set up the new document
Video Ranked 1
GT video rank: 293
duplicate background layer and blue channel layer

Figure 5: Examples of text-to-video clip retrieval results. "GT video rank" represents the rank of the ground-truth clip in the retrieved results for each input query. (a) shows correct results across different objects and operations; and (b) presents failure cases mainly due to too high-level queries or non-visual user actions.

in the first k retrieved videos and the MedR (lower is better) indicates the median rank of the ground truth video.

Implementation details. For the video feature, we equidistantly sample 30 frames from each video and extract 2048-dimensional ResNet-152 [17] feature for each frame. We augment the video feature with the visual-cue-based descriptors in the same way as described in Section 5. We use a 2048-dimensional zero vector if there are no windows or panels detected in a frame and compute the 39-dimensional selected tool feature. For the model training, we use the Adam optimizer [21] to train the model with 40 epochs. We start training with learning rate 0.0002 for 20 epochs, and then lower the learning rate to 0.00002 for the rest 20 epochs. Following [25], we obtain results from two trained VCR models by averaging their predicted similarity scores. Single model results are noted as "Our-S".

Quantitative Results and Analysis. We show results for the proposed method as well as state-of-the-art retrieval models with different input features in Table 3. "V", "P" and "T" represent using visual feature of the entire frame, feature of the pop-up window or panel and one-hot vector encoding for the selected tool respectively. Taking only the visual feature of entire frame as input is the standard setting in general video understanding [4, 37, 45]. It could retrieve some reasonable tutorial videos for the given text descriptions. However, the median rank of ground truth videos is around 50 on the testing set. The reason behind is that the entire frame of a screencast tutorial video includes too much complex information. Some of the information could be redundant or distracting for understanding the user action as described by query text.

From Table 3 we also find that adding extra information from visual cues contributes to better retrieval performance. "V+P+T" models achieve the best results with a big improvement over "V" only. This proves that our visual cues capture key information representing user actions and tutorial content, which effectively helps to learn a better text-to-tutorial video embedding. Besides, the proposed method can improve upon the existing cross–modal retrieval methods, which shows the strength of visual cue reasoning.

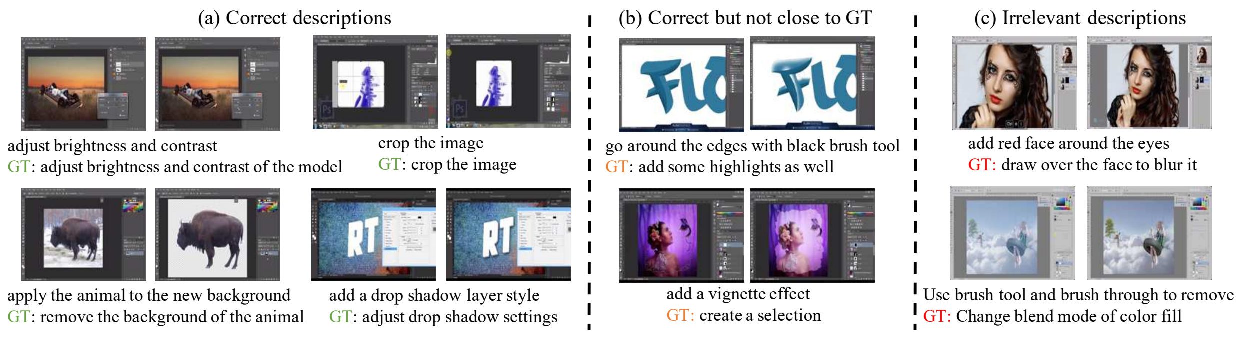
Method	B@1	B@2	B@3	B@4	M	R	Cr
S2VT (V) [44]	26.86	17.17	12.61	9.96	12.15	26.29	90.39
S2VT$_{Att}$ (V)	27.82	18.90	14.26	11.65	13.33	28.83	112.88
Ours (V)	29.90	20.23	14.89	11.65	14.23	30.45	117.92
P+T S2VT	22.03	14.52	10.65	8.80	11.79	28.67	100.77
P+T S2VT$_{Att}$	21.47	14.29	10.52	8.42	10.91	27.87	100.59
Ours (P+T)	22.85	15.23	11.10	9.22	11.95	29.05	104.45
V+P S2VT	29.19	19.51	14.43	11.51	13.62	29.44	111.71
V+P S2VT$_{Att}$	29.34	19.98	15.05	11.99	14.10	30.73	122.05
Ours (V+P)	30.95	21.38	15.94	13.02	14.93	32.08	126.85
V+T S2VT	30.16	20.99	16.17	13.16	14.29	30.71	121.06
V+T S2VT$_{Att}$	31.31	21.59	16.31	13.15	15.02	32.18	128.52
Ours (V+T)	32.38	22.78	17.35	14.66	15.53	33.27	136.84
V+P+T S2VT	31.13	21.84	16.90	13.93	15.01	32.42	129.61
V+P+T S2VT$_{Att}$	32.29	22.31	17.10	13.78	15.50	33.13	133.55
Ours (V+P+T)	**32.63**	**23.42**	**18.32**	**15.12**	**16.06**	**33.92**	**145.02**

Table 4: Quantitative evaluation results of the video captioning model performances on the test set of PsTuts dataset. All values are reported in percentage (%).

Qualitative Results. As shown in Figure 5, our "V+P+T" model can retrieval correct tutorial clips involving various objects and operations for the given text description. However, given descriptions for some general operations, the model may retrieve multiple clips that can match well, resulting in a high rank of the ground-truth clip. Besides, the model can not well handle queries for user actions that lead to imperceptible visual effects and are hard to infer according to current visual cues.

6.3. Tutorial Clip Captioning

We treat each video clip in our dataset as an independent video sample and generate a text description of the major user actions in this clip. We keep the same setting as in Section 6.2. To evaluate the captioning performance, we use standard metrics including BLEU [32], METEOR [10], ROUGE-L [28] and CIDEr [43] with the codes released by [6]. We apply the same preprocessing as in Section 6.2 to obtain visual cue features.


(a) Correct descriptions

adjust brightness and contrast
GT: adjust brightness and contrast of the model

crop the image
GT: crop the image

apply the animal to the new background
GT: remove the background of the animal

add a drop shadow layer style
GT: adjust drop shadow settings

(b) Correct but not close to GT

go around the edges with black brush tool
GT: add some highlights as well

add a vignette effect
GT: create a selection

(c) Irrelevant descriptions

add red face around the eyes
GT: draw over the face to blur it

Use brush tool and brush through to remove
GT: Change blend mode of color fill

Figure 6: Examples of the generated captions along with ground-truth annotations (GT) by Photoshop experts. (a) includes the correct examples almost the same to GT; (b) presents the reasonable outcomes but in a different view from GT, and (c) shows the failure case with irrelevant descriptions.

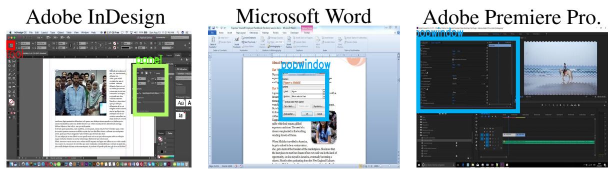


Figure 7: Visual cue detectors generalize well (without fine-tuning) on other software with common design principles, such as Adobe InDesign (left) and Microsoft Word (middle). It fails on Adobe Premiere Pro (right) which includes UI elements quite similar to pop-up windows.

Results and analysis. The captioning performance of different methods is shown in Table 4. "Att" denotes the model with RNN attention mechanism. From the results, we find that incorporating extra visual cues can effectively improve the video captioning quality. This confirms the observation we make in Section 6.2. We also notice the proposed visual cue reasoning method can help improve captioning performance for this new video content. For qualitative results shown in Figure 6, our model can generate reasonable text descriptions for tutorial clips involving various objects and operations.

7. Discussions

In this section, we briefly discuss the generalization and limitation of this work. Adobe Photoshop represents one of the most widely used professional software and shares many common design principles and common UI elements with others, such as InDesign, Illustrator, and Microsoft Word and PowerPoint. We directly (without fine-tuning) apply some of our low-level cue detection modules to screencast videos of these software. As shown in Figure 7, our system can correctly capture the visual events for Adobe InDesign and Microsoft Word. For particular visual cues, such as tool recognition, it may be necessary to collect data and train models specifically for each software. However, it is sufficient to collect a small amount of training data for this step as what we do for Adobe Photoshop. In terms of high-level task understanding, our data annotation pipeline and framework are general enough to be extended to other software. Admittedly, nowadays software is being deployed everywhere, e.g., personal computers, tablets, mobile phones and embedded devices, thus, the visual space of software UI is enormous and full of variation. Figure 7 (right) shows a failure case of cue detection for Adobe Premiere Pro. It is mainly because the UI of Premiere Pro includes regions that are very similar to pop-up windows in Photoshop. Therefore, our model (without fine-tuning on the Premiere Pro data) incorrectly treats these UI regions as pop-up windows. We fully recognize the difficulties to automatically understand all software UIs. We hope this work embarks the first step toward this challenging goal.

8. Conclusion

We propose a new research problem to the computer vision community on visual understanding of screencast tutorials. It is a domain-specific video understanding problem but has significant practical importance as more and more of our life depends on computers and software. We collect the PsTuts dataset which contains diverse and interesting tutorial videos for Adobe Photoshop. We propose an effective bottom-up pipeline to understand Photoshop video tutorials, leveraging state-of-the-art object detection and recognition algorithms and domain-specific visual cues. We evaluate the proposed visual cue reasoning model and existing algorithms for high-level tasks such as tutorial video retrieval and captioning. We conduct extensive experiments on the PsTuts dataset, which demonstrate that our system is a meaningful step towards understanding of clips in tutorial video. In the future it may be illuminating to include more cues into the framework, such as OCR as well as other non-visual signals e.g., speech-to-text transcripts, user logs, etc. Understanding more structured tasks and complex workflows for a wider range of software tools is also a great avenue for further research.

Acknowledgements: This work was supported by Adobe Research funding.

References

[1] Upwork. https://www.upwork.com. 3

[2] Jean-Baptiste Alayrac, Piotr Bojanowski, Nishant Agrawal, Ivan Laptev, Josef Sivic, and Simon Lacoste-Julien. Unsupervised learning from narrated instruction videos. In *Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015. 6

[4] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*, 2015. 1, 2, 4, 7

[5] David L Chen and William B Dolan. Collecting highly parallel data for paraphrase evaluation. In *ACL*, 2011. 4

[6] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*, 2015. 7

[7] Pei-Yu Chi, Sally Ahn, Amanda Ren, Mira Dontcheva, Wilmot Li, and Björn Hartmann. Mixt: automatic generation of step-by-step mixed media tutorials. In *UIST*. ACM, 2012. 2

[8] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv*, 2014. 6

[9] Pradipto Das, Chenliang Xu, Richard F Doell, and Jason J Corso. A thousand frames in just a few words: Lingual description of videos through latent topics and sparse object stitching. In *CVPR*, 2013. 2, 4

[10] Michael Denkowski and Alon Lavie. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the ninth workshop on statistical machine translation*, 2014. 7

[11] Berna Erol and Ying Li. An overview of technologies for e-meeting and e-lecture. In *ICME*, 2005. 2

[12] Fartash Faghri, David J Fleet, Jamie Ryan Kiros, and Sanja Fidler. Vse++: Improving visual-semantic embeddings with hard negatives. In *BMVC*, 2018. 6

[13] David F. Fouhey, Weicheng Kuo, Alexei A. Efros, and Jitendra Malik. From lifestyle vlogs to everyday interactions. In *CVPR*, 2018. 2

[14] Adam Fourney and Michael Terry. Mining online software tutorials: Challenges and open problems. In *CHI*. ACM, 2014. 2

[15] C. Ailie Fraser, Mira Dontcheva, and Scott Klemmer. Software videos: Rich content and learning potential, but a challenge for sensemaking. In *CHI Sensemaking Workshop*. ACM, 2018. 2

[16] Tovi Grossman, Justin Matejka, and George Fitzmaurice. Chronicle: capture, exploration, and playback of document workflow histories. In *UIST*. ACM, 2010. 2

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 5, 7

[18] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014. 2

[19] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 2

[20] Juho Kim. Toolscape: enhancing the learning experience of how-to videos. In *CHI*. ACM, 2013. 2

[21] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv*, 2014. 7

[22] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017. 5

[23] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. Hmdb: A large video database for human motion recognition. In *ICCV*, 2011. 2

[24] Greg C Lee, Fu-Hao Yeh, Ying-Ju Chen, and Tao-Ku Chang. Robust handwriting extraction and lecture video summarization. *Multimedia Tools and Applications*, 2017. 2

[25] Kuang-Huei Lee, Xi Chen, Gang Hua, Houdong Hu, and Xiaodong He. Stacked cross attention for image-text matching. In *ECCV*, 2018. 6, 7

[26] Kunpeng Li, Ziyan Wu, Kuan-Chuan Peng, Jan Ernst, and Yun Fu. Guided attention inference network. *IEEE transactions on pattern analysis and machine intelligence*, 2019. 6

[27] Kunpeng Li, Yulun Zhang, Kai Li, Yuanyuan Li, and Yun Fu. Visual semantic reasoning for image-text matching. In *ICCV*, 2019. 5

[28] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*, 2004. 7

[29] Antoine Miech, Ivan Laptev, and Josef Sivic. Learning a text-video embedding from incomplete and heterogeneous data. *arXiv preprint arXiv:1804.02516*, 2018. 6

[30] Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. In *ICCV*, 2019. 2, 4

[31] Mathew Monfort, Bolei Zhou, Sarah Adel Bargal, Alex Andonian, Tom Yan, Kandan Ramakrishnan, Lisa Brown, Quanfu Fan, Dan Gutfruend, Carl Vondrick, et al. Moments in time dataset: one million videos for event understanding. *arXiv preprint arXiv:1801.03150*, 2018. 2

[32] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, 2002. 7

[33] Suporn Pongnumkul, Mira Dontcheva, Wilmot Li, Jue Wang, Lubomir Bourdev, Shai Avidan, and Michael F Cohen. Pause-and-play: automatically linking screencast video tutorials with applications. In *UIST*. ACM, 2011. 2

[34] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016. 4

[35] Michaela Regneri, Marcus Rohrbach, Dominikus Wetzel, Stefan Thater, Bernt Schiele, and Manfred Pinkal. Grounding action descriptions in videos. *Transactions of the Association of Computational Linguistics*, 1:25–36, 2013. 1, 2, 4

[36] Anna Rohrbach, Atousa Torabi, Marcus Rohrbach, Niket Tandon, Chris Pal, Hugo Larochelle, Aaron Courville, and Bernt Schiele. Movie description. *International Journal of Computer Vision*, 2017. 4, 6

[37] Marcus Rohrbach, Sikandar Amin, Mykhaylo Andriluka, and Bernt Schiele. A database for fine grained activity detection of cooking activities. In *CVPR*. IEEE, 2012. 2, 4, 7

[38] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. The truly deep graph convolutional networks for node classification. *arXiv preprint arXiv:1907.10903*, 2019. 5

[39] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. In *arXiv:1212.0402*, 2012. 2

[40] Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. Videobert: A joint model for video and language representation learning. In *ICCV*, 2019. 2

[41] Yansong Tang, Dajun Ding, Yongming Rao, Yu Zheng, Danyang Zhang, Lili Zhao, Jiwen Lu, and Jie Zhou. Coin: A large-scale dataset for comprehensive instructional video analysis. In *CVPR*, 2019. 2, 4

[42] Atousa Torabi, Christopher Pal, Hugo Larochelle, and Aaron Courville. Using descriptive video services to create a large data source for video annotation research. *arXiv preprint arXiv:1503.01070*, 2015. 1, 2, 4

[43] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *CVPR*, 2015. 7

[44] Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. Sequence to sequence-video to text. In *ICCV*, 2015. 6, 7

[45] Jun Xu, Tao Mei, Ting Yao, and Yong Rui. Msr-vtt: A large video description dataset for bridging video and language. In *CVPR*, 2016. 1, 2, 4, 7

[46] Li Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Pal, Hugo Larochelle, and Aaron Courville. Describing videos by exploiting temporal structure. In *CVPR*, 2015. 4

[47] Dongsong Zhang and Jay F Nunamaker. A natural language approach to content-based video indexing and retrieval for interactive e-learning. *IEEE Transactions on multimedia*, 2004. 2

[48] Luowei Zhou, Chenliang Xu, and Jason J Corso. Towards automatic learning of procedures from web instructional videos. In *AAAI*, 2018. 4