2020

Video Content Understanding Using Text

Amir Mazaheri
University of Central Florida

VIDEO CONTENT UNDERSTANDING USING TEXT

by

AMIR MAZAHERI
M.S. University of Central Florida, 2016
B.S. Sharif University of Technology, 2013

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Spring Term
2020

Major Professor: Mubarak Shah

ABSTRACT

The rise of the social media and video streaming industry provided us a plethora of videos and their corresponding descriptive information in the form of concepts (words) and textual video captions. Due to the mass amount of available videos and the textual data, today is the best time ever to study the Computer Vision and Machine Learning problems related to videos and text. In this dissertation, we tackle multiple problems associated with the joint understanding of videos and text. We first address the task of multi-concept video retrieval, where the input is a set of words as concepts, and the output is a ranked list of full-length videos. This approach deals with multi-concept input and prolonged length of videos by incorporating multi-latent variables to tie the information within each shot (short clip of a full-video) and across shots. Secondly, we address the problem of video question answering, in which, the task is to answer a question, in the form of Fill-In-the-Blank (FIB), given a video. Answering a question is a task of retrieving a word from a dictionary (all possible words suitable for an answer) based on the input question and video. Following the FIB problem, we introduce a new problem, called Visual Text Correction (VTC), i.e., detecting and replacing an inaccurate word in the textual description of a video. We propose a deep network that can simultaneously detect an inaccuracy in a sentence while benefiting 1D-CNNs/LSTMs to encode short/long term dependencies, and fix it by replacing the inaccurate word(s). Finally, as the last part of the dissertation, we propose to tackle the problem of video generation using user input natural language sentences. Our proposed video generation method constructs two distributions out of the input text, corresponding to the first and last frames latent representations. We generate high-fidelity videos by interpolating latent representations and a sequence of CNN based up-pooling blocks.

EXTENDED ABSTRACT

The joint understanding of videos and text is the backbone of many real-world use-cases that is needed to exploit the knowledge captured in today's massive video data. However, understanding of text and videos could be very challenging since they have fundamentally different characteristics while understanding each of them poses challenges. Video is a rich media that can contain loads of useful or irrelevant information. For example, a single video might include humans, objects, and backgrounds that can appear or disappear at any time of the video. However, given a particular task like retrieval by multiple concept queries or answering a question, we need to capture useful part of information and discard the rest. Similarly, a video caption can contain general knowledge about a video, like describing the scene or specific details about a particular action happening in one segment of the video. Furthermore, some information might exist in the text but has no sensible form in the visual form. For example, in the video caption "He is skeptical about the conversation", the word "skeptical" correspond to a specific visual appearance. In this dissertation, we propose solutions that are essential to design of many practical solutions for real-world problems. In each chapter, we introduce and discuss a problem related to joint understanding of video and text, and propose a method to overcome the challenges of the proposed problem.

In Chapter 3, we address the problem of video retrieval. Effective and efficient video retrieval has become a pressing need in the "big video" era. Standard video retrieval methods typically employ single concept as queries. The objective of this work is to provide a principled model for computing the ranking scores of a video in response to one or multiple concepts, where the concepts could be directly supplied by users or inferred by the system from the user queries. Indeed, how to deal with multi-concept queries has become a central component in modern video retrieval systems that accept text queries. However, it has been long overlooked and simply implemented by weighted averaging the corresponding concept detectors' scores. Our approach, employing a latent

ranking SVM, integrates the advantages of various recent works in text and image retrieval, such as choosing ranking over structured prediction and modeling inter-dependencies between querying concepts and so on. Videos consist of shots and we use latent variables to account for the mutually complementary cues within and across shots. In fact, our model captures the positive and negative correlation among the shots and the whole system trains with a ranking-svm objective. We also introduce a simple and effective technique to make our model robust to outliers since the concept labels of shots in the training are scarce and noisy.

While the video retrieval is about finding videos related to a query (set of conceptual words), it does not fully exploit knowledge contained in both video and text. In Chapter 4, we get more focused in extracting detailed knowledge from video and text by studying a novel form of Visual Question Answering (VQA). Given a video and a description sentence with one missing word, "**source sentence**", Video-Fill-In-the-Blank (VFIB) problem is to find the missing word automatically. The contextual information of the sentence, as well as visual cues from the video, are important to infer the missing word accurately. Since the source sentence is broken into two fragments: the sentence's left fragment (before the blank) and the sentence's right fragment (after the blank), traditional Recurrent Neural Networks cannot encode this structure accurately because of many possible variations of the missing word in terms of the location and type of the word in the source sentence. For example, a missing word can be the first word or be in the middle of the sentence and it can be a verb or an adjective. In this chapter, we propose a framework to tackle the textual encoding: Two separate LSTMs (the LR and RL LSTMs) are employed to encode the left and right sentence fragments and a novel structure is introduced to combine each fragment with an *external memory* corresponding to the opposite fragments. For the visual encoding, end-to-end spatial and temporal attention models are employed to select discriminative visual representations to find the missing word. In the experiments, we demonstrate the superior performance of the proposed method on challenging VFIB problem. Furthermore, we introduce an extended and more

generalized version of VFIB, which is not limited to a single blank. Our method won the second place in Large Scale Movie Description Challenge- Fill In the Blank (LSMDC-FIB).

Our proposed approach above for the Video Fill In the Blanks (VFIB) has a lot of potentials to solve related problems and designing applications related to both videos and captions. In Chapter 5, we introduce a new problem called *Visual Text Correction (VTC)*, i.e., finding and replacing an inaccurate word in the textual description of a video. We propose a deep network that can simultaneously detect an inaccuracy in a sentence, and fix it by replacing the inaccurate word(s). Our method leverages the semantic interdependence of videos and words, as well as the short-term and long-term relations of the words in a sentence. Our proposed formulation can solve the VTC problem employing an End-to-End network in two steps: (1)Inaccuracy detection, and (2)correct word prediction. In detection step, each word of a sentence is reconstructed such that the reconstruction for the inaccurate word is maximized. We exploit both Short Term and Long Term Dependencies employing respectively Convolutional N-Grams and LSTMs to reconstruct the word vectors. For the correction step, the basic idea is to simply substitute the word with the maximum reconstruction error for a better one. The second step is essentially a classification problem, where the classes are the words in the dictionary as replacement options. Furthermore, to train and evaluate our model, we propose an approach to automatically construct a large dataset for the VTC problem. Our experiments and performance analysis demonstrates that the proposed method provides very good results and also highlights the general challenges in solving the VTC problem. To the best of our knowledge, this work is the first of its kind for the Visual Text Correction task.

In all the Chapters 3 to 5, we process videos and sentences (or set of concept words), to solve a variety of tasks. In Chapter 6, we propose to create the content of a video directly using a sentence. In fact, we tackle the text to video generation problem, which is a conditional form of generative models for videos. Video generation is one of the most challenging tasks in Machine Learning and Computer Vision fields of study. Humans can listen/read natural language sentences,

and can imagine or visualize what is being described; therefore, we believe that video generation from natural language sentences has a secure impact on Artificial Intelligence. Text-conditioned video generation is relatively a new field of study in Computer Vision, which is far from being solved. The majority of recent works deal with synthetic datasets or real datasets with very limited types of objects and scenes. To the best of our knowledge, this is the very first work on the text (free-form sentences) to video generation on more realistic video datasets like Actor and Action Dataset (A2D) or UCF101. We tackle the complicated problem of video generation by regressing the first and last frames' latent representations and employing a context-aware interpolation method to build up the latent representations of in-between frames. In Chapter 6, We propose a stacking "upPooling" block to sequentially generate RGB frames out of each latent representations and progressively increase the resolution. Moreover, our proposed Discriminator encodes videos based on single and multiple frames. We provide quantitative and qualitative results to support our arguments and show the superiority of our method over well-known baselines like Recurrent Neural Network (RNN) and Deconvolution (as known as Convolutional Transpose) based video generation methods.

To family and friends.

ACKNOWLEDGMENTS

I would like to thank my advisor, Prof. Mubarak Shah, who not only supported me during the doctorate program, but also made me a stronger and better person. Also, I would like to thank Prof. Peter Hancock , Prof. Ladislau Bölöni, and Prof. Fei Liu for being a part of my committee. Finally, I would like to thank all of the past and present members of Center for Research in Computer Vision (CRCV), Cherry Place, Tonya LaPrarie, Boqing Gong, Enrique Ortiz, Amir Roshan Zamir, Afshin Dehghan, Gonzalo Vaca, Nasim Souly, Shervin Ardeshir, Shayan Modiri, Aidean Sharghi, Mahdi Kalayeh, Aisha Orooj Khan, Krishna Regmi, Dong Zhang, Khurram Soomro, Waqas Sultani, Sarfaraz Hussein, Aayush Rana, Robert Browning, Mamshad Rizve, Praveen Tirupattur, Kevin Duarte, Jyoti Kini, Mahfuz Al-Hasan, Marzieh Edraki, Babak Ebrahimi, Furkan Çimen, and Uğur Demir for their help and the memories.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Video data is explosively growing due to the ubiquitous video acquisition devices. Recent years witness a surge of huge video volumes from surveillance, health care, and personal mobile phones to name a few. From the perspective of the IP traffic, Cisco's white paper on Visual Networking Index reveals several astonishing numbers about the Internet videos [2, 3, 4]:

- The sum of all forms of videos will be 82 percent of global traffic by 2021.
- It would take an individual more than 5 million years to watch the amount of video that will cross global IP networks each month in 2020.
- Mobile video traffic exceeded 50 percent of total mobile data traffic in 2012. Note that the global mobile data traffic reached 2.5 exabytes (2.5×10^{18} bytes) per month at the end of 2014.

Meanwhile, thanks to social-media and video streaming industry like Disney or Netflix, videos and corresponding meta-data (such as captions or conceptual tags) are being produced and stored every moment, continuously. This amount of video + text "big data" makes today the best time ever for Computer Vision and Machine Learning (ML) to introduce and solve tasks related to a common understanding of videos and text. In fact, in practice, Natural Language has been the primary tool for people to communicate with any video-based service. YouTube search query, textual descriptions as meta-data, textual comments for expressing feelings, and IMDB blogs for summarization convinces us that natural language is the best way for us to deal with video content.

In this dissertation, we address and discuss multiple problems related to utilizing video datasets, and textual captions of videos. We discuss motivations, challenges, and our proposed solution to tackle each problem.

1.1 Learning a Multi-Concept Video Retrieval Model with Multiple Latent Variables

Indeed, there have been way more videos being generated than what people can watch. Some people have started making fun of this fact; the PetitTube (*www.petittube.com*) randomly plays YouTube (*www.youtube.com*) videos with zero views to its visitors, such that a visitor would become the first one to watch the randomly displayed video. However, it is not funny at all to see the huge volumes of unwatched videos, but rather alarming, for example in the public security domain, and rather challenging for video providers to deliver the right videos upon consumers' requests.

By all means, effective and efficient video retrieval has become a pressing need in the era of "big video", whereas it has been an active research area for decades. Following the earlier research on *content* based video retrieval [5, 6], the most recent efforts have been mainly spent on (multi-) concept based video retrieval [7], an arguably more promising paradigm to bridge the semantic gap between the visual appearance in videos and the high-level interpretations humans perceive from the videos. Multi-Concept based video retrieval employs a learning algorithm which learns to rank videos in response to the queries consisting of more than one concept. It means that, in both training and testing stages, it handles multiple concepts per query. However, the corresponding learning process may become much harder than content-based or single concept-based video retrieval since the number of positive examples for the queries degenerates, the number of parameters of the model grows, and uncertainties increase since different concepts can take place in various parts of a video. These challenges lead us to propose a solution to train a structured model that can be trained and tested with multiple concepts and can deal with the latent inter/intra-shot correlations of the concepts of videos.

A concept corresponds to one or more words or a short description that is understandable by humans. To be useful in automatic video retrieval systems, the concepts (e.g., furniture, beach, etc.) have also to be automatically detectable, usually by some statistical machine learning algorithms,

employing the low-level visual cues (color, texture, etc.) in the videos. Some studies have shown that a rich family of concepts coupled with even poor detection results (10% mean average precision) is able to provide high accuracy results on news video retrieval—comparable to text retrieval on the Web [8]. Both the richness of the concept set and the performance of the concept detectors are essential. Correspondingly, a plethora of works has been devoted to learning concept detectors [9, 10, 11, 12, 13, 14].

Common users have been used to text-based queries for retrieving the target instances in their minds. Equipped with a set of concept detectors, a concept-based video retrieval system is able to accept text descriptions as the queries even when the videos have no textual metadata associated, thus offering users the same interface for the video retrieval as for document or website retrieval (e.g., [15]). As shown in Figure 1.1, users can directly select concepts from a checklist to compose the queries. Alternatively, the system can also let the user's query be a open-vocabulary text and then translate the queries to a subset of concepts. The latter itself is a very interesting problem to which a full treatment is beyond the scope of this chapter. Readers who are interested in this problem are referred to [16, 17, 18, 15, 19, 20, 21]. In either case, the central operation afterward is to use the resultant subset of concepts to retrieve related videos. In this chapter, we focus on retrieving whole videos as opposed to segments or shots of videos; however, the developed approach can be conveniently applied to video segments retrieval as well.

Despite being the key component in (multi-)concept based video retrieval, how to effectively retrieve videos that are related to a subset of concepts is left far from being solved. There is a lack of a principled framework or unified statistical machine learning model for this purpose. Instead, most existing works take the easy alternative by ranking videos according to the weighted average of the concept detection confidences [22, 23], where the weights are either uniform or in some systems derived from the similarities between an open-vocabulary user query and the selected concepts. This necessarily fails to take account of the reliabilities of the concept detectors, the

relationships between the selected concepts, and the potential contributions from the unselected concepts. Our empirical study actually shows that the unselected concepts can significantly boost the video retrieval performance when they are modeled appropriately.

The objective of this work is to provide a principled model for multi-concept based video retrieval, where the concepts could be directly provided by the users or automatically selected by the system based on user queries. Figure 1.1 highlights the main focus of this chapter in the central panel.

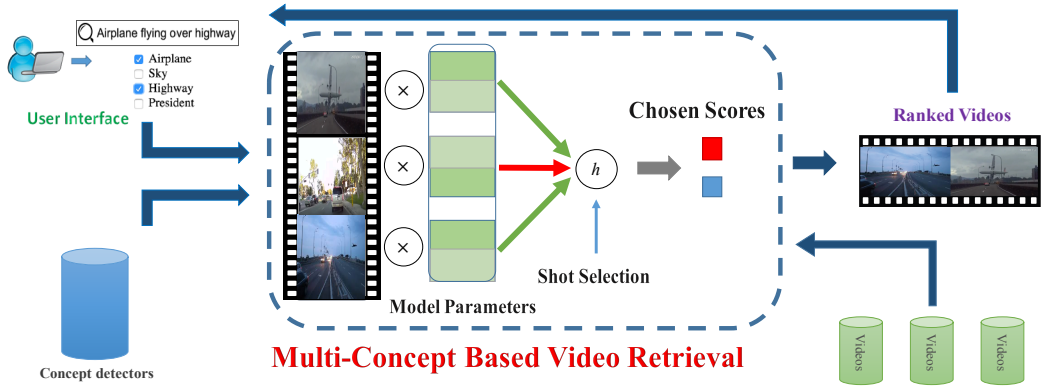


Figure 1.1: How to calculate the ranking scores of videos in response to one or more concepts is the central component in many video retrieval systems. It takes as input the multi-concept queries and then returns a ranked list of videos. The multiple concepts in a query could be directly supplied by the users or inferred by the systems from the users' text queries. We apply a set of pre-trained concept detectors on all the shots of the videos. Our model learns to select the most important shots to score a video using multiple latent variables.

Our approach, which can be considered as a latent ranking SVM [24], integrates different advantages of the recent works on text retrieval and multi-attribute based image retrieval. Particularly, we model the video retrieval as a ranking problem following [25], as opposed to the structured prediction problem used in [26, 27], in order to harvest better efficiency and larger modeling capacity to accommodate some latent variables. The latent variables help choose the shots which are the most responsive to the concepts in a user query, without the need of tediously labeling the shots in the training set. We use them to define the scoring functions both within and across the video

shots, closely tracking the unique temporal characteristic of videos. Besides, we incorporate the empirically successful intuitions from multi-attribute based image retrieval [26, 27] that the inter-dependencies between both selected and unselected concepts/attributes should be jointly modeled. Finally, we introduce a novel 0-1 loss based early stopping criterion for learning/optimizing our model parameters. This is motivated by the fact that the 0-1 loss is more robust to the outliers than the hinge loss, which is used to formalize the optimization problem.

The proposed model, along with a family of provided scoring functions, accounts for some inevitable caveats of the concept detection results: reliabilities of individual concept detectors, inter-dependencies between concepts, and the correlations between selected and unselected concepts in response to a user query. It expressively improves upon the conventional weighted average of the selected concept detection scores for the multi-concept based video retrieval. To this end, we stress again that, as the central component in modern video retrieval systems, how to effectively transform the selected concepts to the ranking scores of videos has been long overlooked and is under exploited. More advances and progress on this problem are in need since they will significantly increase the overall performance of the video retrieval systems.

1.2 Video Fill In the Blank using LR/RL LSTMs with Spatial-Temporal Attentions

This Section aims to solve the Video Fill-In-the-Blank (VFIB) problem: given a video and its sentence description one missing word, the goal is to find the missing word. The missing word can be at anywhere in the sentence, and it can correspond to any word class (e.g. noun, verb, adjective, etc.) Solving VFIB problem requires a strong encoding of the textural and visual cues. Figure 1.2 shows an example for the problem. VFIB problem has a broad variety of real-world applications such as corrupted data recovery, visual guided report generation, etc.

Single Blank:

He ___ up the steps of the stand and away. (Runs)

Multiple Blanks:

He ___ up the steps of the ___ and away. (Runs, Stand)

Figure 1.2: An example for the Video-Fill-in-the-Blank Problem [1].

Video Fill-In-the-Blank (VFIB) is a relatively new problem, and it is related to Video Question Answer (VQA) problem; however, it has significant differences and is a more challenging problem. First, the VQA problems usually have a bias to some specific forms of questions such as location, color, counting, etc. Therefore, the answers for each of these questions are limited to a small dictionary. For example in DAQUAR question answering dataset [28], in many cases, the word "table" is the answer to "What is" question, and "White" to "What color" questions [29]. Furthermore, for VQA problems, there is always a complete sentence to be the “question”. In this scenario, it is easier to encode the “question” in the model (for example, using standard models such as Recurrent Neural Networks) and then use it to predict the answer. Therefore it is easy and straightforward to use off-the-shelf techniques. However, there is no “question" for the VFIB problem, so it is tricky to encode the target sentence with the standard encoding techniques. Last but not least, for VQA problem, it is very expensive to collect datasets which limit its practical applications. It is time-consuming since the human annotators have to generate question and answer one by one by watching the videos. There are some efforts to make the question-answer generation

automatic [29], but these approaches tend to generate a lot of mistakes, and they tend to work well only on object related questions.

Due to these challenges, the VFIB problem cannot be well-solved by traditional Recurrent Neural Network (e.g. LSTM). Therefore, we propose a new framework which takes advantage of the target sentence structure and integrates the spatiotemporal attention models to fully exploit the visual information. There are three main contributions for the proposed method: First, it employs a novel approach to encode the VFIB target sentence using two separate LSTMs (for the left and right sentence fragments). This method naturally encodes the incomplete sentence structure accurately, and it is different from the Bidirectional LSTM [30] since the latter shares the input in reversed order, which can be confused by the sentence structure. Second, we propose a novel framework for the spatiotemporal attention models, which is trained end-to-end. Third, different visual features are employed by the spatial and temporal attention models, which exploit the visual cues from different domains.

1.3 Visual Text Correction

Text Correction (TC) has been a major application of Natural Language Processing (NLP). Text Correction can be in form of a single word auto-correction system, which notifies the user of misspelled words and suggests the most similar word, or an intelligent system that recommends the next word of an inchoate sentence. In this dissertation, we formulate a new type of Text Correction problem named *Visual Text Correction* (**VTC**). In VTC, given a video and an inaccurate textual description in terms of a sentence about the video, the task is to fix the inaccuracy of the sentence.

The inaccuracy can be in form of a phrase or a single word, and it may cause grammatical errors, or an inconsistency in context of the given video. For example, the word “car” in the sentence:

"He is swimming in a car" is causing a textual inconsistency and the word "*hand*" is causing an inaccuracy in the context of the video (See Figure 1.3).

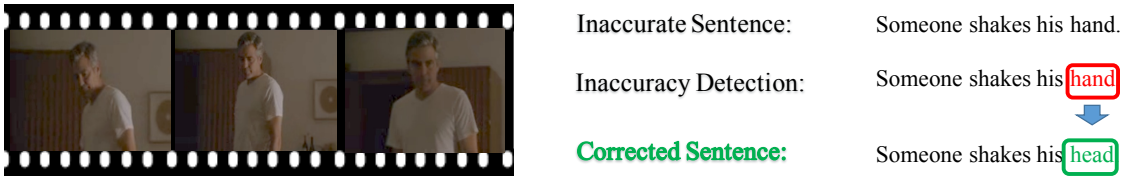


Figure 1.3: An example of an inaccurate sentence for a given video. The VTC task is to find the inaccuracy and replace it with a correct word.

Why Visual Text Correction? We believe that the VTC is very challenging and is a demanding problem to solve. During the last few years, the integration of computer vision and natural language processing (NLP) has received a lot of attention, and excellent progress has been made. Problems like Video Caption Generation, Visual Question Answering, etc., are prominent examples of this progress. With this chapter, we start a new line of research which has many potential applications of VTC in real-world systems such as caption auto correction for video sharing applications and social networks, false tolerant text-based video retrieval systems, automatic police report validation, etc.

Why is VTC challenging? Given a large number of words in a dictionary, many different combinations of words can take place in a sentence. For example, there are $\binom{|V|}{3}$ possible triplet combinations of words from a dictionary of size $|V|$, which makes pre-selection of all possible correct combinations impractical. Also, in many cases, even a meaningful combination of words may result in an incorrect or inconsistent sentence. Furthermore, sentences can vary in length, and the inaccuracy can be in the beginning, middle or at the end of a sentence. Last but not least, a VTC approach must find the inaccuracy and also choose the best replacement to fix it. The video can provide useful information in addition to text since some words of the sentence, like verbs and nouns, need to be consistent with the video semantics like objects and actions present in the video.

To formalize the problem, let sentence $\mathcal{S} = [w_1, w_2, ..., w_N]$ consisting of N words be an accurate description of the video $\mathcal{V}$, where $w_i \in \{0, 1\}^{|V|}$, and $|V|$ is the number of words in our dictionary. For an inaccurate sentence $\widetilde{\mathcal{S}} = [\widetilde{w}_1, \widetilde{w}_2, ..., \widetilde{w}_N]$, the VTC task is to find the inaccurate word $\widetilde{w}_{t^*}$ where $1 \leq t^* \leq N$ and also to estimate the replacement word w_t. There can be several inaccurate words in a sentence; However, we train our system using sentences with just one inaccurate word. Nonetheless, we show that our trained network can be applied to sentences with multiple inaccurate words.

Our proposed formulation can solve the VTC problem employing an End-to-End network in two steps: (1)Inaccuracy detection, and (2)correct word prediction. Figure 1.3 shows the proposed framework of our approach. During the first step, we detect the inaccuracy by reconstruction, that is, we embed each word into a continuous vector, and reconstruct a word vector for each of the words in the sentence based on its neighboring words. A large distance between the reconstructed vector and the actual word vector implies an inaccurate word. For the second step, the basic idea is to simply substitute the word with the maximum reconstruction error for a better one. The second step is essentially a classification problem where the classes are the words in the dictionary as replacement options.

Contributions of this chapter are three-fold. First, we introduce the novel VTC problem. Second, we propose a principled approach to solve the VTC problem by decomposing the problem into inaccurate word detection and correct word prediction steps. We propose a novel sentence encoder and a gating method to fuse the visual and textual inputs. Third, we offer an efficient way to build a large dataset to train our deep network and conduct experiments. We also show that our method is applicable to sentences with multiple inaccuracies.

1.4 Video Generation in the Wild from Text Employing Latent Path Construction for Temporal Modeling

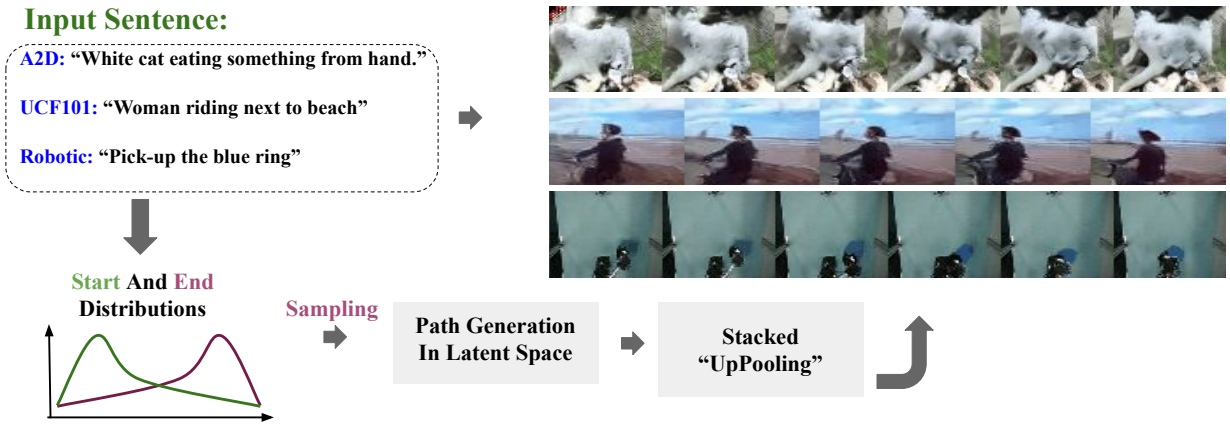


Figure 1.4: In Chapter 6,we tackle the problem of video generation from a sentence. In our proposed method, given an input sentence, we construct two distributions for the latent representations of the first and last frames. We build a path in the latent space between distributions of start and end frame. We generate high fidelity video frames by sampling from the latent constructed path through stacked "UpPooling" layers.

Videos and corresponding descriptions and captions are being produced and stored every moment, continuously. This amount of joint video and text "big data" makes today the best time ever for Computer Vision and Machine Learning (ML) to formulate and solve tasks related to a joint understanding of videos and text. In practice, Natural Language has been the primary tool for users to communicate with any video-based service. YouTube search query, textual descriptions as meta-data, textual comments for expressing feelings, and IMDB blogs for summarization convince us that natural language is the best way for us to deal with video content. We believe that similar to mentioned use cases, natural language is the best medium to create video content as well. It is easy for humans to express in detail what they want to see in the video, i.e., describing the colors, actions, objects. Plus, language is universal! Any human, with any background and skills, can express what he/she needs to create as a video! Considering all, we believe that generating videos from textual descriptions is a valuable task to study and solve, from both computer vision and

real-world usability perspectives.

Realistically, a text to video content creator method must support free-form language and cover a broad set of activities, objects, etc. However, to the best of our knowledge, current works on text to video generation are mainly focused on synthetic datasets, or real datasets with a limited content domain, like only cooking videos. In Chapter 6, we tackle the task of Video Generation with textual inputs for more realistic datasets i.e. videos in the wild; containing more natural videos and sentences compared to prior works. Please refer to Figure 1.4 for high-level overview of our approach.

Modeling video content is one of the ultimate goals of Computer Vision (CV). A video generation model must be able to produce the spatial and temporal variations which have a natural coherence and consistency of a real video. Meanwhile, having a textual input sentence, while constraints the possible variations of a video, adds more complexity to the generative model, since the context of the input text must be visible and understandable from the generated video. Temporal dynamics is the key difference between videos and images, and makes the video generation a more challenging problem. Traditionally, Recurrent Neural Networks (RNNs) and Deconvolutions are intuitive options to model the temporal dynamics of videos. Instead, in this work, we propose a novel method to capture temporal dynamics by first regressing the first and last frames' latent representations from text description and employing a context-aware interpolation method to build up the latent representations of in-between frames. We show that our proposed simple but yet efficient model produces superior results compared to other known techniques like RNNs, or Deconvolution.

1.5 Organization

In Chapter 2, we review existing literature on joint understanding vision and language, and also video retrieval. In Chapter 3, we present our proposed multi-concept video retrieval model with multiple latent variables. In Chapter 4, we explain in details our VQA model for FIB-style questions and the spatial and temporal attentions employed for video understanding. In Chapter 5, we propose a new problem related to FIB, named Visual Text Correction (VTC), which has many real world applications. We also propose a new formulation to solve the VTC problem. Finally, in Chapter 6, we propose to generate video content out of textual input through a latent path construction.

CHAPTER 2: LITERATURE REVIEW

In this chapter, we comprehensively study the literature related to this dissertation. In Section 2.1 we go through the general video retrieval models and we get more specific on retrieval models which work with concepts. In Section 2.2, we briefly review the problems related to both Computer Vision (CV), and Natural Language Processing (NLP). Furthermore, we have a brief review on Deep Learning methods to process the text, and also more specifically methods for correcting errors in natural language sentences in Section 2.3. Finally, in Section 2.4, we we review the related works on Generating Videos from Text which includes some brief discussions about Generative Adversarial Networks (GANs), recent Video Generation works, and Video Generation with a Textual Constraint.

2.1 Video Retrieval

One of the biggest known challenges in Artificial Intelligence is having systems with understandings close to humans. It has been shown that visual concepts have a promising effect in this understanding of videos or images specially for retrieval problems [31, 7, 32]. Concepts not only capture the semantics of an environments, they also bridge the gap between low-level visual cues and higher level representations which humans can understand.

User queries for a retrieval system can include one or more concepts [16, 17, 18, 15, 19, 20, 21, 33, 34, 35]. The order of the retrieved videos is supposed to be ranked by the relevance of each video to the query concepts. In many works, this is achieved by manually defined similarities and heuristic fusion techniques [36, 22, 23, 37] by mapping the user query concepts to the pool of existing concept detector's scores. Some works like [38] take into account the positive correlation

of concepts occurring in videos. In contrast to existing works, we introduce a principled model to automatically learn both positives and negative correlations of concepts at video and shot-level to rank videos based on queries containing multiple concepts.

Broadly speaking, concept detection encompasses a wide spectrum of topics that are potentially useful for the concept-based video retrieval. Some earlier works [39, 40, 41] approached this task with tens to hundreds of concepts, while the most recent work has significantly upgraded the scale [42, 14]. Concepts can take different forms. For instance, object bank [43], action bank [44], image cell based detections [45], classes [46], data-driven concepts [11], sentiment concepts [42], events [14], and so on. Whereas many concept detectors are trained from manually labeled datasets [47, 48], some other works harvest detectors from the noisy Web data [49]. Our approach benefits from these works. Almost all the existing concept detectors can be seamlessly integrated into our multi-concept based video retrieval model by the two types of scoring functions to be described in Section 3.1.

Work on image retrieval and ranking using multiple attributes [26, 27] is the most relevant to the multi-concept based video retrieval. However, due to the vast number of video shots in a database, the structural SVM [50, 51] model used in [26, 27] becomes intractable in our experiments especially when the annotations are incomplete like the datasets we use in this research. Instead, we develop a simpler ranking model with a variety of scoring functions for our retrieval problem. Our proposed model, which uses multiple latent variables to select shots, is flexible to implement and merges many different scoring functions and also is robust to partially annotated datasets. Also, in contrast to [52], which learns to assign a "genre" to each video, our system ranks the videos based on multiple-concepts. Since, a genre can be too specific or too general for what a user is searching for and doesn't match the requirements.

There is a pile of works on learning to rank using the large-margin principle [53, 54, 55, 56, 25, 24].

The conventional ranking SVMs [54, 55] only learn the ranking function for one query, however, our model is learned for all the queries in the training set and we show it can be generalized to the queries not seen in training as well without extra training. The model in [25] is the closest to this work when coupled with our video-level scoring functions, and the latent ranking SVM [24] is the closest to ours when coupled with the shot-level scoring functions. However, our scoring functions are particularly tailored to model the characteristics of videos and despite of [24], we have zero knowledge about the spatial or temporal structure of the concepts in the videos or queries. Besides, we introduce a novel training approach to learn the model parameters by involving the 0-1 loss in an early stopping criterion.

2.2 Visual Question Answering and Fill-In-the-Blank

Deep Convolutional Neural Networks (CNNs) [57, 58, 59, 60] have gain dramatic success in computer vision, for detection (e.g. object detection [61]) and classification (e.g. action classification [62]). And Recurrent Neural Networks (RNN) [30, 63, 64] has been demonstrated useful in Natural Language Processing (NLP), for translation. Recently, new problems such as Visual Captioning (VC) [65, 66, 67], Visual Question Answer (VQA) have drawn a lot of attention, as these are very challenging problems and are extremely valuable for both computer vision and natural lauguage processing. Both Visual Captioning and Visual Question Answer are related to the Video-Fill-in-the-Blank (VFIB) problem.

Visual Captioning (VC) needs deep understanding for both visual cues (i.e. image or video) and textual cues. Many approaches for VC has been introduced recently. Some of these algorithms focus on leveraging RNNs which takes visual input and produces words one by one. Another type of approaches, has the benefits to focus on different spatial locations and leverage attention models to produce better captions [67, 68]. Furthermore, in [68] a dense captioning method has been

proposed, where for a given image, multiple captions are produced and each caption comes with a bounding box. Visual Question Answering (VQA)[69, 29, 28, 29, 70, 71, 72] is also related to our problem. It has a deep root in textual question answering [73, 74, 75]. The goal of textual question answering is to answer a question based on a collection of sentences.

The major difference between VQA with the VC problem is the existence of a question other than the image or video. This makes the problem harder since the question could be about details of the image or video; however, for the VC problem, any fact about the image can be a correct caption. Some methods use a combination of the question with the image/video through LSTMs [76] and use the output to answer the question. Other methods combine the image features with the words one by one, for instance, the method in [71] is an extension of [74], and both visual and textural information is input into the dynamic memory networks and converted to a sequence through an RNN with an iterative attention process. In [72], a binary question answering (yes/no) on abstract (unreal) scenes is discussed. Another binary answering problem which verifies existence of a relation between two concepts (for example: dogs eat ice cream) is proposed in [77] and it uses CNNs to verify the questions. In [78], it shows another form of VQA, however the questions in this case come as a video and a subtitle. In [75], it uses multiple choice questions, but it was demonstrated that the visual information does not contribute much to the final results.

It is clear that the Video-Fill-in-the-Blank (VFIB) problem is very different from the Visual Captioning (VC) problem, since the inputs to these problems are totally different. And the major difference between VFIB problem and the VQA problem is that, the VFIB problem does not have a complete sentence as the “question"", and the sentence comes as fragments with a blank that can be at anywhere in the sentence. Therefore, we propose a new method to encode the incomplete sentence, and we employ spatiotemporal attention models to capture the object-centric and motion-centric visual features to fill in the blank.

2.3 Visual Text Correction

Several NLP works benefit from N-Grams [79, 80], and convolutional N-Grams [81, 82] to encode the neighborhood dependencies of the words in a sentence. The recent work in [82] show the superiority of N-Gram Convolutions over LSTM methods in sequence to sequence translation task. Therefore, in this paper we leverage N-Grams convolutions and Gating Linear Unit [83] in encoding the text and also incorporating visual features in our inaccuracy detection network. In addition, studies on encoding semantics of words [84, 85], phrases and documents [86, 87] into vectors have been reported. The main goal of all these studies is to represent the textual data in a way that preserves the semantic relations. In this research, we use the representation and distance learning to reconstruct each word of a sentence and find the inaccurate word based on the reconstruction error.

Fill-In-the-Blank (FIB) [88, 89, 90] is the most related to our work. FIB is a Question Answering task (as explained in Section 2.2), where the question comes in the form of an incomplete sentence. In the FIB task, the position of the blank word in each sentence is given and the aim is to find the correct word to fill in the blank. Although FIB is somehow similar to the proposed VTC task, it is not straightforward to correct an inaccurate sentence with a simple FIB approach. In FIB problem the position of the blank is given, however in VTC it is necessary to find the inaccurate word in the sentence first and then substitute it with the correct word.

Traditional TC tasks like grammatical and spelling correction have a rich literature in NLP. For instance, the authors in [91] train a Bayesian network to find the correct misspelled word in a sentence. Other line of works like [92, 93], try to rephrase a sentence to fix a grammatical abnormality. In contrast to works in [91, 94, 92, 92, 93], there is no misspelled word in our problem, and we solve the VTC problem even for cases when the grammatical structure of the sentence is correct. Also, reordering the words of a sentence [93] cannot be the solution to our

problem, since we need to detect and replace a single word while preserving the structure of the sentence. Moreover, this is the first work to employ the videos in the Textual Correction task.

2.4 Video Generation Using Text

Generative Adversarial Networks (GANs) Currently, Generative Adverserial Networks (GANs) are some of the most popular frameworks in Deep Learning. There exist numerous variations of GANs, including Wasserstein GAN [95], Cycle GAN [96], ProgressiveGAN [97], Hinge-loss [98] GAN, and etc.; however, the backbone of any GAN framework deals with a competition between a generative and discriminative neural networks, named G and D [99]. GANs are powerful tools to reconstruct the true distribution of a set of data points, even if it is impossible to estimate the distribution parameters analytically. In this work, we employ a hinge-loss GAN, to generate videos out of natural sentences inputs.

Video Generation: Modeling the temporal coherency and consistency of natural video frames makes the video synthesis problem as one of the most challenging generative problems. Some forms of the video synthesis task has been studied in recent years. Authors in [100] solve the task of video synthesis as a conditional generation problem where the condition is the human pose skeleton; however, it strongly depends on the human pose estimator and needs training data for each of human subjects, and scenes. Similarly, authors in [101] animate any arbitrary object given a driving video sample. This method detects a few key-points in each frame, and estimates a dense warping map by generalizing the key-points motion to all similar points.

Video synthesis can be combined with other computer vision tasks, like object [102] or semantic segmentation [103], Robot manipulation [104], and etc. Authors in [102] utilize video synthesis as an unsupervised technique to learn rich features for the task of Video Object Segmentation

(VOS) with limited supervised data. They train a video generator by decomposing any video into a foreground object and a background scene. Similarly, authors in [104] learn unsupervised features for robotic object manipulation tasks. Also, the work proposed in [103] generates videos conditioned on the first semantically segmented frame. Similarly, authors in [105] can generate videos out of a sequence of semantically segmented input frame.

Video generation can be also in form of video prediction [104, 106, 107, 108], inpainting [109], etc. Video prediction is to estimate the future frames of a video given the preceding ones. Video prediction is the most established and popular kind of video generation. The video inpainting task [109], similar to image inpainting [110], is to modify a specific spatial region of every single frame in a video.

A simplified form of video generation problem is to generate a video given a class label. Authors in [111] show that it is possible to generate high fidelity videos on a large number of classes. Similarly, the proposed method in [112] decomposes a video into content and motion subspaces and generates a video by sampling a point and a path in the two subspaces, respectively.

Generation with Textual Constraint: Textual sentences are the simplest form of natural human language; and transforming them into other mediums like video [113, 114, 115], image [116, 117], or speech [118, 119] is one of the most interesting problems in Artificial Intelligence. Authors in [116] propose a progressive [97] text to image generation method which leverages text to image attention at multiple resolutions. Authors in [115], crawl YouTube by some selected search queries, and clean results to obtain a dataset for training the text to video neural network that produces a gist image from a sentence, and animate the gist image; however, sentences in [115] are mostly in the form of "Action" + "Place", which is a simple-form compared to the sentences of our target dataset, A2D [120]. In this work, we use videos in the wild datasets like A2D [121, 120] and UCF101 [62] (We provide the sentence annotations for nine classes of UCF101 in this paper). Datasets of our

interest are not curated for the task of text to video generation and have complicated sentence structures. Authors in [114] solve the task of video generation using text on simpler datasets like MNIST moving digits [122] and KTH action [123], using a Negative-Log-Likelihood (NLL) loss. 3D Deconvolutions and LSTMs have been used in [113] and [124] to generate multiple frames to form a video. In this work, we propose our novel method to generate any number of needed frames to synthesis a video, and we show the performance of text to video generation on more challenging datasets.

2.5 Summary

In this Section, we provide and discussed several works related to the materials of this dissertation. We include discussions about various types of methods for Video Retrieval, Visual Question Answering, Text Correction, and also Video Generation.

CHAPTER 3: LEARNING A MULTI-CONCEPT VIDEO RETRIEVAL MODEL WITH MULTIPLE LATENT VARIABLES

The work in this Chapter have been published in the following papers:

Mazaheri, Amir, Boqing Gong, and Mubarak Shah. "Learning a Multi-concept Video Retrieval Model with Multiple Latent Variables." 2016 IEEE International Symposium on Multimedia (ISM). IEEE, 2016.[125]

Mazaheri, Amir, Boqing Gong, and Mubarak Shah. "Learning a multi-concept video retrieval model with multiple latent variables." ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 14.2 (2018): 46. [126]

With respect to the recent exponential grow of video data, the goal of this Chapter is to advance the video retrieval systems. In this chapter, we introduce our main approach to multi-concept based video retrieval. First, we formalize a ranking model for retrieval problem and then we describe different scoring functions and how to integrate them into our retrieval model.

3.1 Methodology

3.1.1 A ranking model for multi-concept based video retrieval

Ranking videos according to one or more concepts selected by the users/systems is the main component in modern video retrieval systems (cf. Figure 3.1). Whereas most of existing recent works

consider a weighted-average of the detection confidences of concepts to rank the videos, we aim to enhance this component by a principled ranking model, which is flexible enough to incorporate different kinds of concept detectors and also can be generalized to unseen queries and deal with temporal dynamics of videos and the correlation of concepts.

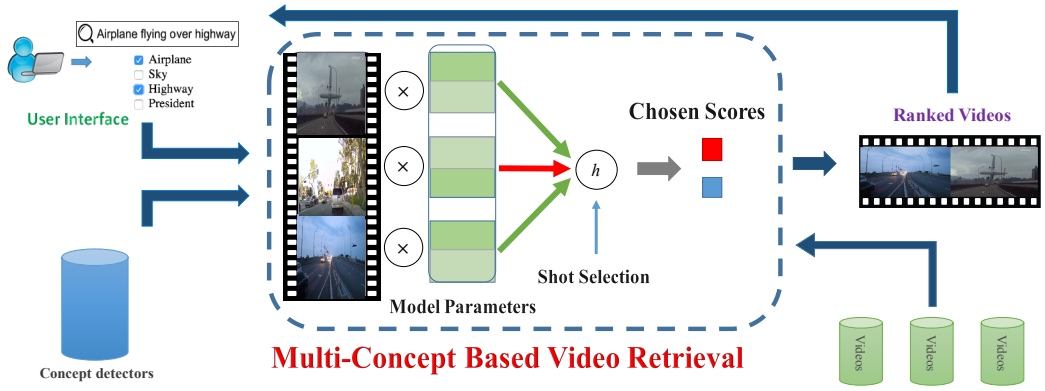


Figure 3.1: How to calculate the ranking scores of videos in response to one or more concepts is the central component in many video retrieval systems. It takes as input the multi-concept queries and then returns a ranked list of videos. The multiple concepts in a query could be directly supplied by the users or inferred by the systems from the users' text queries. We apply a set of pre-trained concept detectors on all the shots of the videos. Our model learns to select the most important shots to score a video using multiple latent variables.

We denote all the concepts defined for the system by $\mathcal{Q}$ which users compose queries with and all the videos in the dataset by $\mathcal{V}$. $R(Q) \subset \mathcal{V}$ is the set of all videos that are related to a multi-concept query $Q \subset \mathcal{Q}$. Therefore, an ideal retrieval system must use a learning algorithm to select the best possible subset, named $R(Q)$, among all other subsets of videos from $\mathcal{V}$ such that:

$$\forall S \subset \mathcal{V}, S \neq R(Q),\ R(Q) \text{ is a better output than } S. \tag{3.1}$$

Directly modeling this notion gives rise to a structured prediction model presented in [26] and strengthened in [27]. We appreciate that this is perhaps the most natural choice for the retrieval

model. There exists $2^{|\mathcal{V}|}$ distinct subsets from $\mathcal{V}$. Unfortunately, this exponential number of subsets makes the computations expensive. Moreover, the expressiveness of the model's scoring function is limited to special forms, to tailor the function to utilize the training and testing algorithms for structured prediction [50, 127]. Our own experiments tell that it is computationally intractable to use this retrieval model for the shot-level concept detections (cf. Section 3.1.3).

3.1.1.1 Retrieval as ranking

To avoid the high cost of aforementioned structure model, we relax the retrieval problem and solve it as a ranking problem following [25]. We accommodate multiple latent variables in our model to keep track of the shot-level detections. In particular, the rigorous criterion (eq. (3.1)) for retrieval is replaced by a less constrained ranking criterion,

$$\forall V_i \in R(Q), \forall V_j \notin R(Q),\ V_i \text{ ranks ahead of } V_j, \tag{3.2}$$

given V_i and V_j as a pair of relevant and irrelevant videos in the database $\mathcal{V}$ according to the query Q.

Comparing eq. (3.1) with eq. (3.2), the former calls for a model to operate over $2^{|\mathcal{V}|}$ subsets of videos while for the latter we only need a model to assign a ranking score for each video $V \in \mathcal{V}$. The exponential computation complexity makes the solution impractical even for datasets with a few hundreds of videos. We use the following ranking model in this work $\mathcal{F} : \mathcal{Q} \times \mathcal{V} \mapsto \mathbb{R}$,

$$\mathcal{F}(Q, V) = \frac{1}{|Q|} \sum_{q \in Q} f(q, V|\mathbf{\Theta}), \tag{3.3}$$

which breaks down into several ranking scoring functions $f(q, V|\mathbf{\Theta}), q \in Q$, each for an individual concept, and $\mathbf{\Theta}$ denotes the model parameters. We shall write $f(q) \triangleq f(q, V|\mathbf{\Theta})$ in the following

for brevity, and leave the discussion of the scoring functions to Sections 3.1.2 and 3.1.3.

We rank the videos in the database given a multi-concept query Q using $\mathcal{F}$. Top ranked videos in the database will be returned to the user as the video search results. Compared to the retrieval model based on structured prediction [26, 27], our model is not able to optimize the number of videos to output. However, we argue that this does not reduce the usability of our ranking model, considering that common users are used to ranking lists due to text retrieval.

3.1.1.2 Learning model parameters

To train the described model, with parameters $\mathbf{\Theta}$, we follow the ranking SVM [55, 54] strategy. We write our objective function such that:

$$\min_{\mathbf{\Theta}} \quad \sum_{Q} \frac{1}{|\mathcal{N}(Q)|} \sum_{(i,j)\in\mathcal{N}(Q)} L\left(\mathcal{F}(Q,V_i) - \mathcal{F}(Q,V_j)\right), \tag{3.4}$$

given $\mathcal{N}(Q)$ as the set of all the pairs of videos V_i and V_j in eq. (3.2) for the query Q and $L(x) \geq 0$ is a loss function. i and j takes indices of positives and negative videos in the dataset regarding to query Q. One can obtain the pairs from user annotated ranking lists of videos. In Section 3.2.1, we provide more details about the selection of positive and negatives in our experiments. The loss function will impose some amount of penalty when the ranking scores of a pair of videos violate the ranking constraint of Eq. (3.2).

We exploit two types of loss functions in this work, the hinge loss $L_{\text{hinge}}(x) = \max(1-x, 0)$ and 0-1 loss $L_{\text{0-1}}(x)$ which takes the value 1 when $x > 0$ and 0 otherwise. *Note that we cannot actually solve the optimization problem with the 0-1 loss by gradient descent;* we instead use it to define a novel early stopping criterion when we solve the problem with hinge loss. Namely, the program stops when the change of the objective function value, computed from the 0-1 loss, is less than a

threshold (10^{-10} in our experiments).

As a result, we are able to take advantage of the fact that the 0-1 loss is more robust than the hinge loss when there are outliers in the data. The hinge loss alone would be misled by the outliers and results in solutions that are tuned away from the optimum, while the 0-1 loss helps avoid that situation by suppressing the penalties incurred by the outliers. Improvements using this novel 0-1 loss stopping criterion, is more sensible in multi-concept video retrieval problem, since for most of the concepts there are many outliers or videos with very different appearances from usual (e.g. cartoons, partial occlusion, distance and point of view and etc.) and also noisy annotations for big datasets. These outliers make the training stage much more harder. Indeed, the novel stopping criterion by the 0-1 loss significantly improves the results of hinge loss in our experiments.

Note that the 0-1 loss based stopping criterion is another key point clearly differentiating our approach from [25], which motivates our ranking model. In addition, we introduce a family of new scoring functions for different concept detections, especially the one with latent variables in Section 3.1.3.

3.1.2 Video-level concept detection

It is a common practice that the concept detection results $\boldsymbol{\phi}(V)$ over each video $V \in \mathcal{V}$ are computed off-line and stored somewhere to speed up the responding to the users' queries. We use $\boldsymbol{\phi}$ as the shorthand of $\boldsymbol{\phi}(V)$. Note that $\boldsymbol{\phi}$ is a $|\mathcal{Q}|$-dimensional vector whose entry ϕ_q corresponds to the detection confidence of the concept q (in a video V). We next describe how to use $\boldsymbol{\phi}$, the video-level concept detection results, to design the scoring functions $f(q), q \in Q \subset \mathcal{Q}$ (cf. Section 3.1.1). We start from the weighted average approach which prevails in the existing video retrieval works.

3.1.2.1 Weighted average

Recall that the overall scoring function $\mathcal{F}(Q, V)$ breaks down into several individual functions $f(q, V|\mathbf{\Theta}) \triangleq f(q), q \in Q$, each of which accounts for one concept (eq. (3.3)). A common practice to rank the videos given a multi-concept query Q is by the average of the corresponding concept detection confidences:

$$f^{\text{V}}_{\text{avg}}(q) = \phi_q \triangleq \langle \mathbf{1}^q, \boldsymbol{\phi} \rangle \,, \tag{3.5}$$

where the weights are simply uniform and $q \in Q$. $\mathbf{1}^q \in \{0, 1\}^{|\mathcal{Q}|}$ denotes an one-hot vector which is 1 at the q-th entry and 0's else, or, when the query is an open-vocabulary text, the weights could be the similarities inferred between the concepts in Q and the user query [17, 21]. We include the uniform weights in our experiments without loss of generality.

The mentioned average method fails to model the correlations between the concepts in Q, and also the correlations between Q and the remaining unselected concepts in $\mathcal{Q}$. To make it more clear, we have re-written the score function by the one-hot vectors on the rightmost of Eq. (3.5). Therefore, the model parameters become $\mathbf{\Theta} = (\mathbf{1}^1, \mathbf{1}^2, \cdots, \mathbf{1}^{|\mathcal{Q}|})^T = I \in \mathbb{R}^{|\mathcal{Q}| \times |\mathcal{Q}|}$, i.e., actually an identity matrix. The entry $\mathbf{\Theta}_{qp}$, which is supposed to encode the relationship of concepts q and p, is 0 in the weighted average (Eq. (3.5)).

3.1.2.2 Encoding concept correlations

To this end, the natural extensions to the weighted average scoring function are the following,

$$f^{\text{V}}_{\text{corr-1}}(q) = \langle \boldsymbol{\theta}^q, \boldsymbol{\phi} \rangle \,, \text{ such that } \boldsymbol{\theta}^q_p = 0 \text{ if } p \notin Q, \tag{3.6}$$

and

$$f^{\text{V}}_{\text{corr-2}}(q) = \langle \boldsymbol{\theta}^q, \boldsymbol{\phi} \rangle \ , \ \text{such that} \ \ \boldsymbol{\theta}^q \in \mathbb{R}^{|\mathcal{Q}|}, \tag{3.7}$$

where $f^{\text{V}}_{\text{corr-1}}(q)$ considers the contributions from the other concepts $p \in Q$ when it scores the concept q for a video V. Indeed, the existence of "computer" could affect the confidence of "furniture" when both are selected to the query $Q \subset \mathcal{Q}$. The other function $f^{\text{V}}_{\text{corr-2}}(q)$ further considers all the other concepts not in the query, when it scores for instance the concept $q =$"furniture" in a video, capturing the case that the lack of "beach" may reinforce the confidence about "furniture". Note that the parameters $\boldsymbol{\Theta} = (\boldsymbol{\theta}^1, \cdots, \boldsymbol{\theta}^{|\mathcal{Q}|})^T$ become a full matrix now, offering more modeling flexibilities.

3.1.3 Shot-level concept detection

Many concept detectors are designed to take frames or video shots as input [14, 44, 45]. We split any given video V in the database into H shots. This can be done by applying a shot detector [128] or uniformly select the shot boundaries inside a full sequence video. We compute and store the concept detection results $\boldsymbol{\phi}^h \in \mathbb{R}^{|\mathcal{Q}|}, h = 1, \cdots, \mathsf{H}$ for all the concepts $\mathcal{Q}$ over the shots of the video. Shot-level detections provide more detailed information about the videos than video-level concept detections considering the dynamics of the video and sudden changes of the scene. Therefore, we propose our novel family of scoring functions which leverage the shot-level detection scores.

3.1.3.1 Scoring the best shot for the querying concepts

One potential benefit we can have from the shot-level concept detections is that, among all the shots of a video V, we can select the most informative shot for the scoring function:

$$f^{\text{S}}_{\text{latent}}(q) = \max_{h \in \{1,2,\cdots,\text{H}\}} \left\langle \boldsymbol{\theta}^q, \boldsymbol{\phi}^h \right\rangle, \tag{3.8}$$

where the model parameters $\boldsymbol{\theta}^q \in \mathbb{R}^{|\mathcal{Q}|}$, which correspond to the concept $q \in Q \subset \mathcal{Q}$, represent the contributions to q from all the concepts within the shot, which will be selected by the latent variable $h \in \{1, 2, \cdots, \text{H}\}$.

We argue that this formulation can better model the *negative* correlations between concepts, if any, than the $f^{\text{V}}_{\text{corr-2}}(q)$ defined over the video level. Indeed, consider a set of negatively correlated concepts (e.g., "beach", "furniture", etc.). They could all have very strong responses across a video. For instance, a tourist may capture a video within a hotel room and then shift to the beach outside. As a result, both "beach" and "furniture" will be detected with high confidences in the video but they are exclusive over a single shot. As a result, the video-level scoring function may be confused by this video but $f^{\text{S}}_{\text{latent}}(q)$ scores a single best shot and is thus robust to this scenario.

3.1.3.2 Scoring best shot for each querying concept

While $f^{\text{S}}_{\text{latent}}(q)$ well captures the negative concept correlations by possibly negative off-diagonal entries, it may be inadequate to track the *positive* correlations between different concepts by focusing on only a shot; very few concepts could appear simultaneously in a single shot. We thus

compensate it by a second term:

$$f^{\text{VS}}_{\text{latent}}(q) = \max_{h\in\{1,\cdots,\text{H}\}} \left\langle \boldsymbol{\theta}^q, \boldsymbol{\phi}^h \right\rangle + \sum_{p\in\mathcal{Q}} \max_{g\in\{1\cdots,\text{H}\}} \upsilon_p^q \phi_p^g, \tag{3.9}$$

where $\max_g \upsilon_p^q \phi_p^g$ max-pools the confidences of each concept across all the shots of video V. Note that we therefore provide two complementary types of modeling capabilities in $f^{\text{VS}}_{\text{latent}}(q)$. The first term is robust to the concepts which are negatively correlated with q and the products in the second term strengthen the detection score of concept q from some positively correlated concepts in the video. The model parameters $\boldsymbol{\theta}^q$ and $\boldsymbol{\upsilon}^q$ are learned by solving eq. (3.4) with sub-gradient descent. Details are given as follow.

3.1.4 Optimization

We use (Sub-)Gradient Descent(SGD) as a solver to learn our model parameters $\boldsymbol{\Theta}$. As discussed in Section 3.1.1.2, the loss function L is a non-zero and positive value for each pair $\{(V_i, V_j)\}$, in which the negative video V_j has higher ranking score than the positive V_i. As a result, the gradients for those pairs are non-zero and zero for the others.

Denoting by

$$\mathcal{S}_j = \frac{\partial L}{\partial \mathcal{F}(Q, V_j)} \times \frac{\partial \mathcal{F}(Q, V_j)}{\partial \boldsymbol{\Theta}}, \tag{3.10}$$

we thus have the overall gradients of eq. (3.4) by

$$\sum_Q \frac{1}{|\mathcal{N}(Q)|} \sum_{(i,j)\in\mathcal{N}(Q)} (\mathcal{S}_i - \mathcal{S}_j)\,. \tag{3.11}$$

Note that the model parameters Θ consist of two parts $(\boldsymbol{\theta}, \boldsymbol{\upsilon})$, corresponding to the two terms of $f^{\text{VS}}_{\text{latent}}$ (cf. eq. (3.9)), respectively. We compute the gradients with respect to the first part $\boldsymbol{\theta}$ using the $softmax$ derivation to approximate a smooth gradients, as suggested by [129]:

$$\frac{\partial \mathcal{F}(Q, V_j)}{\partial \boldsymbol{\theta}} = \sum_{h \in \{1,2,\cdots,\mathsf{H}\}} \frac{e^{\langle \boldsymbol{\theta}^q, \boldsymbol{\phi}^h \rangle} \boldsymbol{\phi}^h}{\sum_{j \in \{1,2,\cdots,\mathsf{H}\}} e^{\langle \boldsymbol{\theta}^q, \boldsymbol{\phi}^j \rangle}}. \tag{3.12}$$

We write out the gradients with respect to the second part, $\boldsymbol{\upsilon}$, over different dimensions of $\boldsymbol{\upsilon}$. Recall that he second term of $f^{\text{VS}}_{\text{latent}}$ (cf. eq. (3.9)), $\max_g \upsilon^q_p \phi^g_p$, max-pools over all the shots of a video for each single concept. As a result, we have the following:

$$\frac{\partial \mathcal{F}}{\upsilon^q_p} = \phi^{g*}_p, \quad p = 1, 2, \cdots, \mathsf{H},\ q = 1, \cdots, \mathsf{H}, \tag{3.13}$$

where $g*$ is determined by $g* \leftarrow \max_g \upsilon^q_p \phi^g_p$.

3.2 Results

This section presents our experiments to evaluate the proposed multi-concept video retrieval model along with the various scoring functions. We firstly describe the experiment setup and then report the retrieval results. We further give some detailed analyses and qualitative results.

3.2.1 Experiment setup

We use two separate datasets in our experiments respectively for video retrieval and training the concept detectors. We further exploit four sets of multi-concept queries. Two sets consist of 50 queries each in the form of concept pairs, one for training and testing and the other just for testing.

Two other sets contain 50 triplets and 30 single concept queries just for testing, respectively. We train our model using only the first set of queries on the training set and then test it by all four sets of queries on the test set.

Table 3.1: The multi-concept queries used in our experiments. The concept-pair queries shown in this table are mentioned as **seen queries** and has been used for training the system.

	Queries
50 Concept Pairs	(Chair, Computers), (Boat/Ship, Oceans), (Classroom, Computers), (Instrumental_Musician, Singing), (Chair, Classroom) (Boat/Ship, Running), (Chair, Nighttime), (Boat/Ship, Quadruped), (Bicycling, Forest), (Chair, Hand), (Chair, Flags) (Boat/Ship, Forest), (Nighttime, Singing), (Cheering, Singing), (Forest, Lakes), (Chair, Telephones), (Running, Stadium) (Chair, Forest), (Beach, Boat/Ship), (Oceans, Quadruped), (Forest, Quadruped), (Beach, Quadruped), (Running, Forest) (Bridges, Forest), (Boat/Ship, Bridges), (Instrumental_Musician, Nighttime), (Highway, Nighttime), (Beach, Oceans) (Bus, Highway), (Bus, Chair), (Nighttime, Forest), (Highway, Forest), (Computers, Telephones), (Nighttime, Running) (Bridges, Highway), (Cheering, Flags), (Cheering, Instrumental_Musician), (Cheering, Nighttime), (Forest, Oceans) (Chair, Highway), (Chair, Quadruped), (Boat/Ship, Lakes), (Running, Quadruped), (Nighttime, Flags), (Bridges, Chair) (Boat/Ship, Nighttime), (Demonstration_Or_Protest, Flags), (Airplane, Boat/Ship), (Boat/Ship, Chair), (Chair, Running)
50 Concept Triplets	(Ocean, Quadruped, Boat/ship), (Chair, Classroom, Computers), (Beach, Boat/Ship, Quadruped), (Cheering, Boat/Ship, Flags) (Chair, Computers, Telephones), (Cheering, Instrumental_Musician, Singing), (Instrumental_Musician, Nighttime, Singing) (Forest, Boat/Ship, Oceans), (Lakes, Boat/Ship, Oceans), (Beach, Ocean, Boat/Ship), (Quadruped, Beach, Ocean) (Forest, Lakes, Quadruped), (Boat/Ship, Bridges, Forest), (Chair, Nighttime, Forest), (Bridges, Forest, Lakes) (Boat/Ship, Forest, Quadruped), (Boat/Ship, Forest, Lakes), (Running, Oceans, Boat/Ship), (Highway, Boat/Ship, Bridges) (Running, Beach, Oceans), (Quadruped, Running, Forest), (Cheering, Instrumental_Musician, Nighttime) (Boat/Ship, Nighttime, Forest), (Chair, Ocean, Boat/Ship), (Running, Oceans, Quadruped), (Chair, Highway, Bridges) (Beach, Forest, Oceans), (Bridges, Oceans, Boat/Ship), (Lakes, Bridges, Boat/Ship), (Running, Beach, Boat/Ship) (Cheering, Singing, Flags), (Demonstration, Bus, Flags), (Lakes, Boat/Ship, Quadruped), (Bus, Chair, Computers) (Boat/Ship, Flags, Oceans), (Forest, Bus, Boat/Ship), (Quadruped, Forest, Bicycling), (Oceans, Forest, Lakes) (Bridges, Chair, Boat/Ship), (Boat/Ship, Bridges, Bus), (Forest, Boat/Ship, Highway), (Cheering, Nighttime, Singing) (Quadruped, Ocean, Forest), (Flags, Demonstration, Nighttime),(Bus, Bridge, Highway), (Boat/Ship, Nighttime, Oceans) (Chair, Boat/Ship, Forest), (Forest, Chair, Quadruped), (Highway, Bus, Chair), (Bridges, Highway, Forest)

3.2.1.1 The IACC.2.B dataset for video retrieval

We mainly test our approach over the IACC.2.B dataset which is the test set used in the Semantic Indexing (SIN) task of TRECVID 2014 [130] challenge. The dataset comprises 2,371 Internet videos which are "characterized by a high degree of diversity in creator, content, style, production qualities, original collection device/encoding, language, etc." [130]. The video durations range from 10 seconds to 6.4 minutes with the mean of 5 minutes. Standard shot partitions (106,000 shots in total) are provided by the dataset and 30 concepts are annotated for the shots. We use TRECVID 2015 SIN Task's test set, named IACC.2.C dataset, as the second dataset for more

experiments as a correctness proof of our approach. All the settings such as splits ratios are as same as IACC.2.B.

We randomly split IACC.2.B to 712 videos as the training set, 474 videos as the validation set and 1,185 videos as the test set. We select 50 pairs of concepts from the total $\binom{30}{2}$ possible pairs. We select them based on the number of positive examples in the training set and call them the seen queries which always will be used in training and in one experiment as the test. A video is a positive or related sample for a query when there is at least one shot annotated as positive for all the concepts in that query. This results in minimally 27, maximally 86, and on average 44 out of the 1,185 videos in the database (i.e., the test set) related to the concept-pair queries. See Table 3.1.

Additionally, we build a set of concept-triplet queries with the size of 50. On average each concept-triplet query has 24 related videos. As a matter of fact, by increasing the number of concepts the retrieval task becomes more challenging due to less number of positive examples and we show in our experiments that our method can handle them very well. We never use concept-triplets as training and use them just for a test as unseen queries. In Table 3.1 we show our set of concept-triplets.

3.2.1.2 Evaluation

We use one of the most popular metrics in information retrieval, Normalized Discounted Cumulative Gain (NDCG) [131], to evaluate the ranking lists returned by our model in response to the multi-concept queries. Given a ranking list for query Q, NDCG is calculated by:

$$\text{NDCG@}k = \frac{1}{Z} \sum_{j=1}^{k} \frac{G[j]}{1 + \log_2 j}, \tag{3.14}$$

given the gain function $G[j] = \text{rel}(j)^2$ and j as the rank of a video which has rel(j) number of concepts that video shares with the query Q. By changing the power in the gain function, we can tune how important is for us that a retrieved video contains all the desired concepts. The partition Z is the ideal gain value computed by ground truth which will produce an ideal ranking list. As a result, any NDCG@k value is normalized between 0 and 1. We shall report the results at $k = 5, 10, \cdots, 50$ in the following experiments.

3.2.1.3 Concept detectors

Learning robust concept detectors has a rich literature [10, 14, 12]. All kinds of concept detectors can be potentially employed in our retrieval model. We train our independent concept detectors following the practice of [132].

In particular, we train 60 independent detectors from the training data (IACC.1.tv10.training, IACC.1.A, IACC.1.B, and IACC.1.C) of the TRECVID 2014 SIN task [130] for the concepts with key frame annotations, including the 30 concepts annotated in IACC.2.B. To this end, we extract dense SIFT [133] (DSIFT) and Convolutional Neural Network (CNN) features [57] from the annotated (both positive and negative) key frames for the 60 concepts. The DSIFT features computed using VLFeat [134] toolbox, are encoded by Fisher vectors [135] as an image representation and then input to linear SVMs. For CNN features, we use the activations of "relu6" and "fc7" layers as two types of image representations, train SVMs with histogram intersection kernels for each of them, and then average the detection scores of the two types of SVMs. Overall, we thus harvest two complementary detection confidences for any concept, one from the DSIFT and the other from the CNN features. They are both transformed to probabilities using the Platt calibration. At the testing stage, we firstly average them to obtain the concept detection confidences for each frame, then max-pool the scores within a shot to have the shot-level results $\boldsymbol{\phi}^h$, and finally arrive at the

video-level concept detection results ϕ by max-pooling.

3.2.1.4 Practical considerations in implementation

In order to prevent the model from overfitting, we have employed an $L2 - norm$ as regularizer. In particular, we add $\sum_{q \in \mathcal{Q}} \lambda \left\| \boldsymbol{\theta}^q \right\|_2^2 + \gamma \left\| \boldsymbol{v}^q \right\|_2^2$ term to regularize the optimization problem in Eq. (3.4). We have used the validation set to tune the hyper-parameters λ and γ. Note that $\gamma = 0$ except for the scoring function $f^{\text{VS}}_{\text{latent}}(q)$. We also remove all the videos which has no shot annotated as negative or positive for the given query.

Table 3.2: Comparison results of different scoring functions in pair-concept based video retrieval.

Baselines

Functions		NDCG@5	@10	@15	@20	@25	@30	@35	@40	@45	@50	Mean
PAMIR [25]		04.3%	04.2%	04.1%	04.3%	04.7%	05.2%	05.7%	06.1%	06.5%	06.7%	05.22%
Fast0Tag [136]		07.6%	07.2%	07.3%	07.7%	08.0%	08.2%	08.6%	08.9%	09.0%	09.1%	08.2%
Common practice	$f^{\text{V}}_{\text{avg}}$	62.6%	57.1%	55.6%	56.1%	57.5%	58.8%	59.7%	61.0%	62.0%	62.6%	59.3%
TagProp [137]		30.0%	27.3%	25.6%	26.8%	27.7%	28.6%	29.4%	30.1%	30.8%	31.4%	28.8%
Rank-SVM [138]		57.9%	52.9%	52.2%	52.6%	54.3%	55.4%	56.5%	56.8%	57.7%	58.1%	55.5%
Co-occurrence [38]		59.4%	50.7%	48.6%	49.5%	51.8%	53.4%	54.9%	55.6%	56.4%	57.4%	53.8%
PicSOM 2013 [139]		63.0%	57.1%	55.5%	55.9%	57.3%	58.1%	59.2%	60.5%	61.5%	62.1%	59.0%
$\lvert\mathcal{Q}\rvert$= **30 concepts**												
Video-level	$f^{\text{V}}_{\text{corr-1}}$	61.6%	57.0%	55.5%	55.8%	57.0%	58.1%	59.5%	60.5%	61.5%	62.2%	58.9%
Video-level	$f^{\text{V}}_{\text{corr-2}}$	64.8%	59.2%	57.4%	57.6%	58.9%	60.3%	61.5%	62.3%	63.1%	63.3%	60.9%
Shot-level	$f^{\text{S}}_{\text{latent}}$	*68.2%*	*61.0%*	*59.5%*	59.5%	61.2%	62.1%	63.6%	64.9%	65.8%	66.1%	*63.2%*
Shot-Video-level	$f^{\text{VS}}_{\text{latent}}$	62.9%	58.8%	58.3%	*60.0%*	*61.8%*	*63.1%*	*64.7%*	*65.4%*	*66.2%*	*67.1%*	62.8%
$\lvert\mathcal{Q}\rvert$= **60 concepts**												
Video-level	$f^{\text{V}}_{\text{corr-1}}$	61.6%	57.0%	55.5%	55.8%	57.0%	58.1%	59.5%	60.5%	61.5%	62.2%	58.9%
Video-level	$f^{\text{V}}_{\text{corr-2}}$	64.8%	58.8%	57.3%	57.3%	58.8%	60.0%	61.1%	62.4%	62.9%	63.2%	60.7%
Video-level with RBF Distance	$f^{\text{V}}_{\text{corr-2}}$	64.0%	59.3%	57.3%	57.3%	58.7%	60.1%	61.1%	62.4%	62.9%	63.3%	60.6%
Shot-level	$f^{\text{S}}_{\text{latent}}$	67.6%	61.8%	59.8%	59.7%	61.5%	62.6%	64.1%	65.1%	65.8%	66.5%	63.5%
Shot-Video-level	$f^{\text{VS}}_{\text{latent}}$	69.8%	**63.8%**	**61.7%**	**60.9%**	**63.0%**	**64.1%**	**65.4%**	**66.4%**	**66.8%**	**67.4%**	**64.9%**
Shot-Video-level with RBF Distance	$f^{\text{VS}}_{\text{latent}}$	**70.2%**	62.2%	60.0%	60.3%	62.1%	63.2%	64.5%	65.4%	66.1%	67.1%	64.1%

3.2.2 Comparison results

We compare different scoring function results in Table 3.2. For this experiment, we have used IACC.2.B dataset and the 50 pair-concepts queries shown in Table 3.1 for both training and testing.

We evaluate using NDCG@k where $k = 5, 10, \cdots, 50$ for different columns. The most left column shows the mean of NDCG for all k values.

Following the common practice in the existing concept-based video retrieval systems, we empirically test a variety of fusion methods [35, 36, 139] as the (old) baselines—our approach offers a new set of simple yet more advanced ranking scheme for the multi-concept-based video retrieval. Probably because our detectors output probabilities after the Platt calibration, the average operation in $f^{\mathrm{V}}_{\mathrm{avg}}$ performs the best among the fusion techniques discussed in [35]. We thus only show the results of $f^{\mathrm{V}}_{\mathrm{avg}}$ and the second best, PicSOM [139], in the rows tagged by "Common practice" and "PicSOM 2013", respectively, in Table 3.2. The PicSOM fusion strategy [139] involves a convex combination of the product and the average of the querying concepts' detection scores. Also, we used an another common technique as explained in [38] to capture just positive correlation between pairs of concepts, using a Co-occurrence matrix of them built in training stage.

We further include ranking SVM [55] and TagProp [137] in the table as more advanced baselines. Both take as input the video-level representations; they are not able to handle the set of shot-level features in each video. We use two types of inputs, the video-level concept detection scores, and CNN features as the video representations to train the TagProp and ranking SVM models. Note that we train a ranking SVM model for each of the pair-concept queries. TagProp is a state-of-the-art image tagging algorithm. It uses K-nearest neighbor and metric learning to propagate the tags of training examples to any testing instance. We report the best results for each after parameter tuning on our validation set. Probably because our training set is relatively small, the TagProp results are very low. Ranking SVM gives comparable results with the other fusion techniques.

Moreover, we adopt Fast0Tag [136] and Passive Aggressive Model for Image Retrieval (PAMIR) [25] methods on all three datasets (cf. Table 3.4). Both of these methods try to find a mapping between image feature and concepts. Fast0tag uses the popular pre-trained word2vec [140] model that is

trained on a large text corpus, to embed each concept in a high dimensional space where word (concept) similarities can be measured more accurately. It maps each image to its corresponding unique direction to rank concept scores. PAMIR also follows the same idea; however, this method learns to represent each query of multi-concept by a single vectors.

There are four types of scoring functions in our approach, $f^{\text{V}}_{\text{corr-1}}$ and $f^{\text{V}}_{\text{corr-2}}$ accounting for the video-level concept detections and $f^{\text{S}}_{\text{latent}}$ and $f^{\text{VS}}_{\text{latent}}$ for the shot-level concept detections. We learn these models' parameters by the 0-1 loss based early stopping and the 50 pair-concept queries (cf. Table 3.1) using the videos in the training set. Experiments comparing the hinge loss and the 0-1 loss are presented in Section 3.2.3.

To further study our model's behavior, we replace the linear distance between positive and negative neighbors described in Eq. 3.4 with a non-linear distance. To be more specific, we replace $\mathcal{F}(Q, V_i) - \mathcal{F}(Q, V_j)$ in Eq. 4, with $sign(\mathcal{F}(Q, V_i) - \mathcal{F}(Q, V_j)).(1 - K(\mathcal{F}(Q, V_i), \mathcal{F}(Q, V_j)))$, where K is the Radial Basis Function (RBF) similarity function. We keep the $sign$ of the distance since we are solving a ranking problem and the order of the ranked samples must be preserved.

Comparison

There are $|\mathcal{Q}| = 30$ concepts labeled for our video database $\mathcal{V}$, which are drawn from the IACC.2.B dataset. All our queries are constructed from these concepts such that we have the ground truth ranking list for evaluation. We first show the video retrieval results in the top half of Table 3.2. The variations of our model with different scoring functions all improve the common practice $f^{\text{V}}_{\text{avg}}$. The margins between $f^{\text{V}}_{\text{avg}}$ and our latent shot-level functions are especially significant.

The benefit of more concepts: Though the queries are built from the vocabulary of 30 concepts, we are actually able to harvest more concept detectors from another independent dataset,

TRECVID 2014 SIN task training set. Our model is flexible to include them by expanding the concept detection vectors ϕ (see Section 3.1). The bottom half of Table 3.2 shows the results corresponding to 60 concept detectors. We see that the observations about the relative performances of the model variations from the $|\mathcal{Q}| = 30$ concepts still hold. In addition, the video retrieval results using the shot-level scoring functions have been significantly improved over those of the 30 concepts. This is in accordance with our intuition as well as the results in [27]. Indeed, the inter-dependences of more concepts may provide better information for our scoring functions and make them more robust to the unreliable concept detection confidences.

Note that, however, introducing more concepts does not change the results of the video-level scoring function $f^{\mathrm{V}}_{\text{corr-2}}$ too much, $f^{\mathrm{V}}_{\text{corr-1}}$ is not affected by extra concepts. We argue that this is mainly due to the fact that our detectors are not developed for the video-level concept detections. For the future work, it will be interesting to see whether more video-level concept detectors, such as the action classifiers [141, 142], can benefit our video-level function $f^{\mathrm{V}}_{\text{corr-2}}$ as well. Another interesting direction would be to pursue the weak attributes/concepts in the video retrieval task [27].

3.2.3 The effect of the 0-1 loss

We study the effect of the novel 0-1 loss based stopping criterion in this section. Figure 3.2 shows the retrieval results of both the video-level scoring function $f^{\mathbf{V}}_{\text{corr-2}}$ and shot-level function $f^{\mathrm{VS}}_{\text{latent}}$, respectively, with and without using the 0-1 loss in the optimization. We can see that coupling the 0-1 loss as a stopping criterion with the hinge loss in optimization significantly improves the performance of the hinge loss alone for both types of scoring functions. This is not surprising. The 0-1 loss is advantageous over the hinge loss especially when there are "difficult" positive-negative pairs which heavily violate the ranking constraint in the training/optimization stage. The hinge loss would penalize more those pairs and consequently ignore the other pairs, but the 0-1 loss is

resilient to those pairs. Although in practice we cannot fully harvest the appealing modeling power of the 0-1 loss due to the gradient descent, our results in Figure 3.2 verify that the nice properties of the 0-1 loss can be transferred indirectly by defining the new stopping criterion with the 0-1 loss. And also, it shows a better performance for top retrieved videos (smaller k values in Figure 3.2) where the 0-1 stopping criterion shows a lot of improvement. We also provide qualitative results in Figure 3.6 for more clarifications.

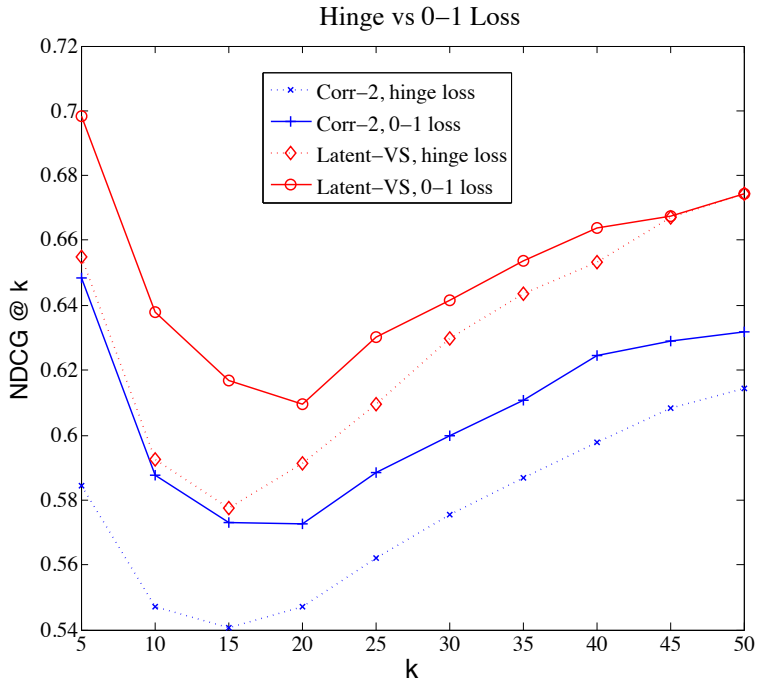


Figure 3.2: The effects of the 0-1 loss based stopping criterion in optimization. For both the video-level scoring function $f^{\text{V}}_{\text{corr-2}}$ and shot-level function $f^{\text{VS}}_{\text{latent}}$, the introduction of the 0-1 loss significantly improves the performance of the hinge loss.

3.2.4 Generalizing out to unseen queries

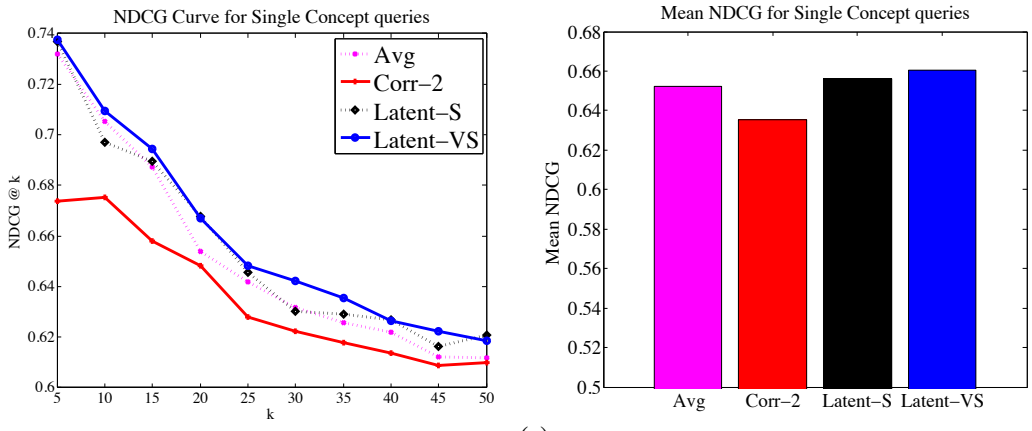
We expect our model to generalize well to other multi-concept queries. To demonstrate this, we train our system with pair-concept queries shown in Table 3.1 and test it on three other set of queries

of which none of them have any query in common with the ones used in training: (a) 30 single-concept queries, (b) 50 new pair-concept queries, and (c) 50 new triple-concept queries. None of them are used to train our model. The 50 concept triplets are shown in Table 3.1. Figure 3.3 shows the retrieval results using different variations of our model. We can see our model with the shot-level scoring functions $f^{\mathrm{S}}_{\mathrm{latent}}$ and $f^{\mathrm{VS}}_{\mathrm{latent}}$ performs quite well upon the new queries. The results are not only significantly better than the simple average, but also comparable to those for the previously seen pair-concept queries (cf. Table 3.2). The video-level scoring function $f^{\mathrm{V}}_{\text{corr-2}}$ unfortunately degrades and gives similar or worse performance compared to the averaging baseline $f^{\mathrm{V}}_{\mathrm{avg}}$. It implies that our multi-latent variable based scoring function is able to generalize the trained model and use it to retrieve unseen queries.

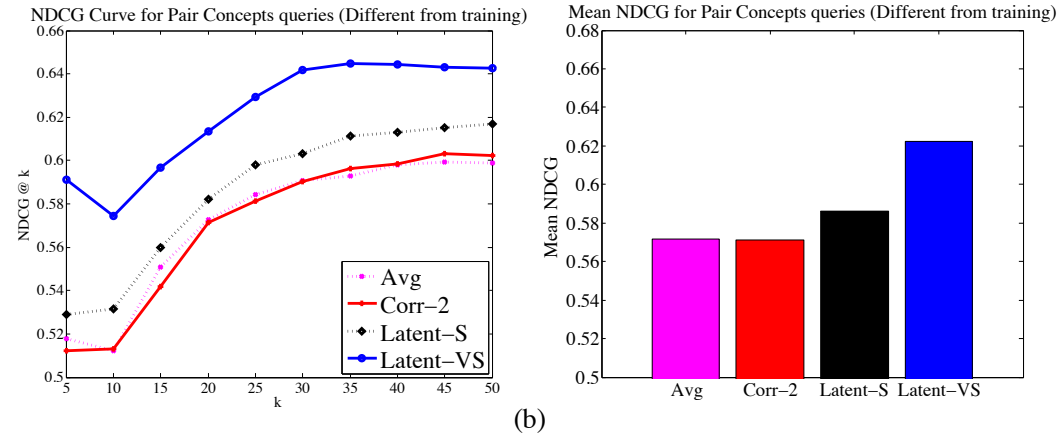
In a complementary experiment, we show our model can be used on TRECVID SIN challenge. Even though this challenge is based on short video shots and single concept queries retrieval, an improvement is expected due correlation of concepts that our model captures. For this experiment, we use the same testing pipeline. Shots are considered as videos and each frame as a single shot. Th results are given in Table 3.3. We use the same settings as explained in Section 3.2.1 and the learned model is the same as used in other experiments. To give an insight into performance of the independent concept detectors, we try them on the full set as well. The numbers are reported using mean InfAP [143] over 30 concepts. First 2,000 retrieved shots for each concept are considered. Note that our model is not applicable to full dataset due to 50% usage of that in training stage.

Table 3.3: Results for TRECVID 2014 SIN challenge.

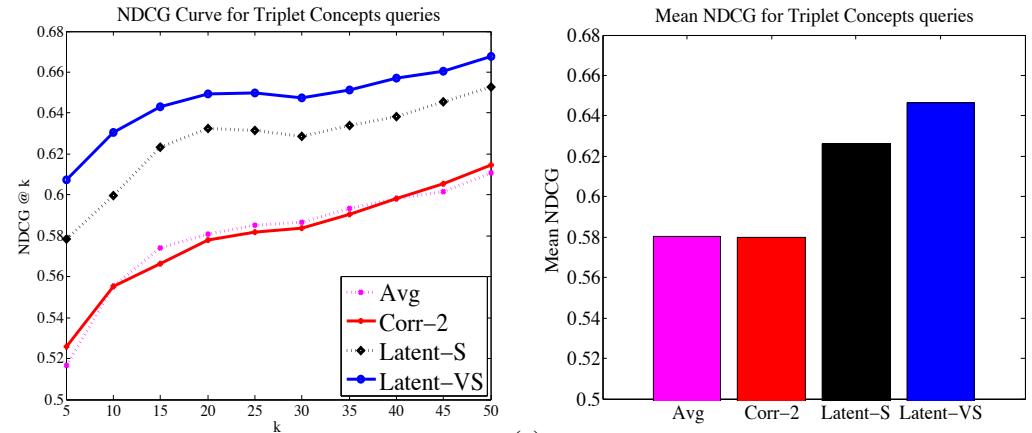
Method	$f^{\mathrm{V}}_{\mathrm{avg}}$	$f^{\mathrm{VS}}_{\mathrm{latent}}$
IACC.2.B (Full) - TRECVID SIN 2014	24.01	-
IACC.2.B (Half) - TRECVID SIN 2014	24.56	24.80



(a)



(b)



(c)

Figure 3.3: *previously unseen* queries: (a) single-concept queries, (b) pair-concept queries, and (c) triple-concept queries.

Table 3.4: Baseline averages of NDCG@5-50 on three different datasets

Dataset	IACC.2.B	IACC.2.C	IACC.2.B + C
PAMIR	5.2%	38.3%	18.0%
Fast0Tag	8.2%	30.6%	18.4%
TagProp	28.8%	24.1%	12.3%
Rank-SVM	55.5%	26.4%	39.5%
Co-occurrence	53.8%	44.4%	30.2%
PicSOM	59.0%	53.2%	40.3%
$f^{\mathrm{V}}_{\mathrm{avg}}$	59.3%	53.7%	40.4%
$f^{\mathrm{V}}_{\text{corr-2}}$	60.7%	50.1%	37.8%
$f^{\mathrm{VS}}_{\mathrm{latent}}$	**64.9%**	**56.9%**	**43.5%**

3.2.5 Extensive Experiments on IACC.2.C dataset

To have an extensive set of experiments we use IACC.2.C dataset. As explained in Section 3.2.1.1, we use similar setting for extracting queries and also training/validation/test sets as IACC.2.B. We go further integrate IACC.2.C and IACC.2.B to build one super dataset. In Table 3.4, we show a side by side comparison between three different datasets as well as different scoring functions and baselines. Clearly, our multi-latent variables method gives the superior performance in all three cases.

Notice a drop in all the performances in Table 3.4. To explain this, we compute the ratio of an average number of positive of all queries and the total number of videos in each dataset (Table 3.5). Smaller ratio makes the retrieval harder since there are less number of samples in training stage and also a higher drop even a single positive sample is lost in the testing stage.

Table 3.5: Mean ratio of positives to total number of videos for pair-concept queries

Dataset	IACC.2.B	IACC.2.C	IACC.2.B + C
Mean Ratio	**0.0159**	0.0134	0.0106

3.2.6 Time Complexity

In Table 3.6, we show the time complexities comparison between different methods, where n is the number of training samples, $|H| << n$ is the number of shots in the video and $|\mathcal{Q}| << n$ is the number of concepts.

Table 3.6: Time Complexities Comparison.

Method	Time Complexities
PicSOM	$\mathcal{O}(1)$
Co-occurrence	$\mathcal{O}(n)$
Fast0tag	$\mathcal{O}(n)$
Rank-SVM	$\mathcal{O}(n^2)$
Tagprob	$\mathcal{O}(n^2)$
$f^{\text{V}}_{\text{corr-2}}$	$\mathcal{O}(n^2)$
$f^{\text{V}}_{\text{corr-2}}$	$\mathcal{O}(n^2)$
$f^{\text{S}}_{\text{latent}}$	$\mathcal{O}(\|H\| \times n^2)$
$f^{\text{VS}}_{\text{latent}}$	$\mathcal{O}(\|H\| \times \|\mathcal{Q}\| \times n^2)$

3.2.7 User study on YouTube search engine

We build a new dataset using 230 videos gathered from YouTube. These videos are retrieved using 9 pair-concept queries shown in Table 3.1. 20 to 30 videos for each of the queries are downloaded and their actual ranks in YouTube are saved. Shots are also built by clipping videos into 2-second shots. We apply our method for each query and get a new ranking list. 10 randomly selected users are asked to compare these two ranking lists. One of them is YouTube original list and the other is from our method, and we ask users to choose the best list for corresponding query search, just based on visual cues. User voting is completely blind and participants do not have access to other participants' decisions. Even though our detectors are trained on training set explained in Section 3.2.1.1, which is different from YouTube videos, in Figure 3.4 we show for some queries,

users mostly like the re-ranked list using our method. YouTube ranking has a bias toward meta-data coming with the videos and our method can retrieve videos containing the actual concepts instead.

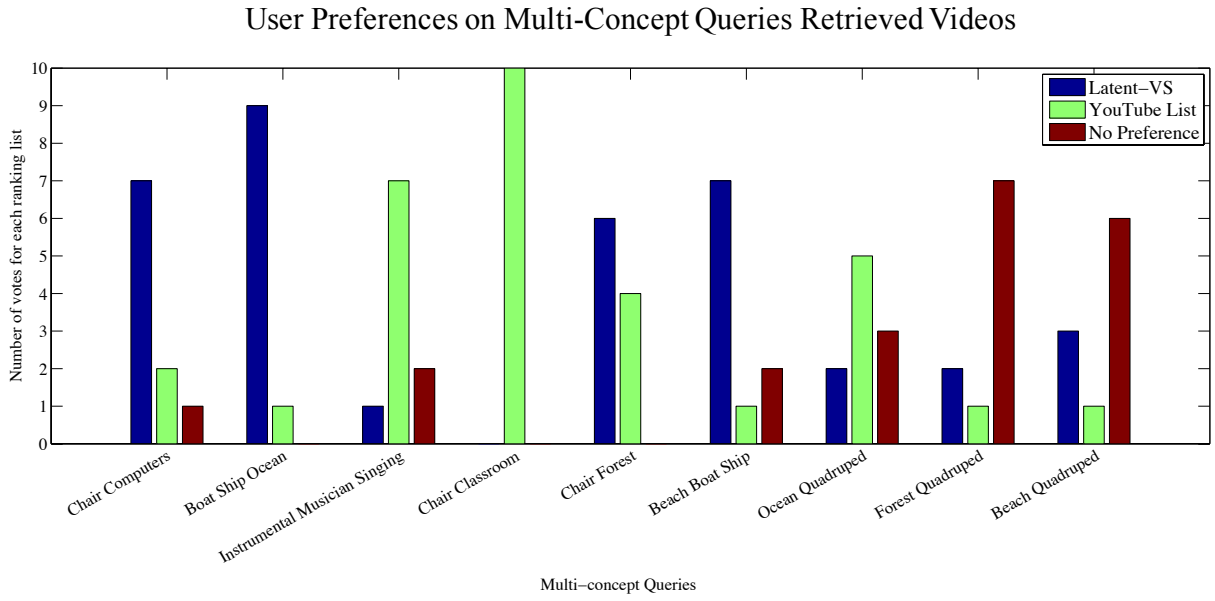


Figure 3.4: The number of user votes for different ranking lists and queries. Users preferred the ranking lists by our algorithm to Youtube's default ranking lists for six out of nine queries.

3.2.8 Qualitative analyses

We show some videos and their ranks using $f^{\mathrm{V}}_{\mathrm{avg}}$ and also after applying $f^{\mathrm{VS}}_{\mathrm{latent}}$ in Figure 3.5. In general, our approach works especially well under the following two scenarios. One is when the concepts appear in different video shots, and the other is when the concepts are not very relevant and more unlikely to happen in the same video. (e.g., bridge, chair, and highway).

We also show how our method can correctly rank videos for a given query. In Figure 3.6, we

show top five videos retrieved by our method for one of the pair-concept queries and a comparison with the average baseline. The top ranked videos are more important especially for common users searches since people tend to find a proper video at the top of their searching list result.

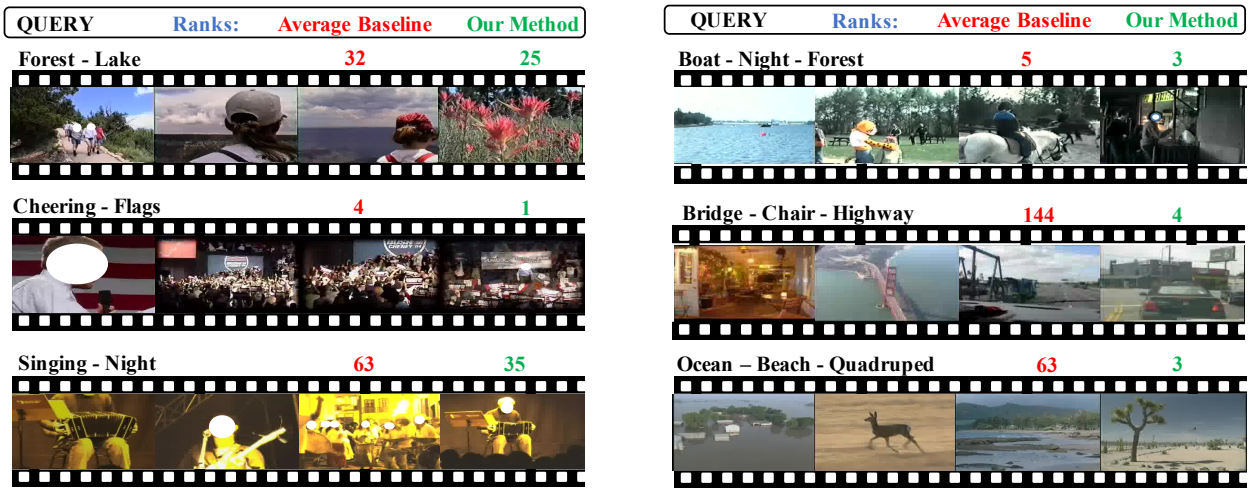


Figure 3.5: Some examples of queries and the rank of one video containing all the concepts. We show two numbers for each video, the red one is the rank of the video using the Average baseline and the green one is the rank of the video using our method. We show queries with pair concepts and triplet concepts on the left and right columns respectively. In all of the provided examples, the ranking value has been improved using our method.

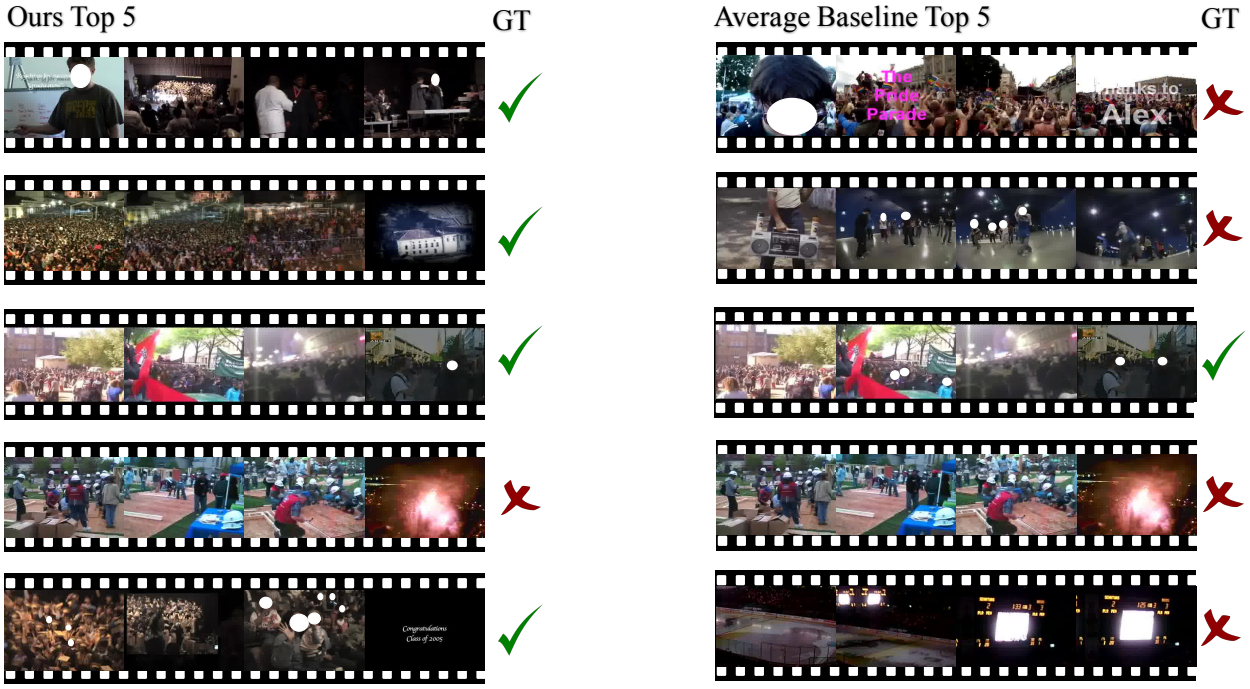


Figure 3.6: Top Ranked videos for the query "Cheering - Night Time". The left panel shows the first top 5 videos ranked by our method and the right panel shows the same for the average baseline method. We also show which of the videos are tagged as positive in ground truth. Note that, a positive video must have both concepts included.

3.3 Summary

In this Chapter, we introduced a new baseline for multi-concept based video retrieval. Our method uses a principled model which can be integrated and leverage from other existing methods of video/image retrieval. We have designed a multi-latent variable scoring function which can deal with noisy and incomplete annotations of large datasets while it can learn both inter and intra shots dependencies of concepts. We show a technique named 0-1 loss based early stopping criterion which can make the training process more robust to outlier data. Our extensive experiments show our model superiority over other methods. As multi-concept based video retrieval plays a central role in video retrieval systems, we expect our model to advance the state-of-the-art research in video retrieval.

CHAPTER 4: VIDEO FILL IN THE BLANK USING LR/RL LSTMS WITH SPATIAL-TEMPORAL ATTENTIONS

The work in this Chapter have been published in the following paper:

Mazaheri, Amir, Dong Zhang, and Mubarak Shah. "Video fill in the blank using lr/rl lstms with spatial-temporal attentions." Proceedings of the IEEE International Conference on Computer Vision. 2017. [90]

Video Fill-In-the-Blank (VFIB) is a visual question answering problem where the question is in the form of a video description with one missing word. In this Chapter, we propose our solution to solve the VFIB. Our method is composed of a textual encoder that can deal with a fragmented sentence, and also spatial and temporal attention mechanisms which can select spatial regions and temporal segments of a video to answer an input question. Note that, since there is not spatial and temporal annotations available in the dataset, similar to Chapter 3, we formulate the spatial and temporal attentions as latent variables.

4.1 Methodology

We formulate the VFIB problem as a word prediction problem with two sentence fragments (left sentence fragment "q_l", and right sentence fragment "q_r") and the video $\boldsymbol{v}$:

$$\hat{b} = \arg\max_{b \in \beta} p(b|\boldsymbol{q}_l, \boldsymbol{q}_r, \boldsymbol{v}, \boldsymbol{\theta}), \tag{4.1}$$

where $\beta \subset V$ is the set of words to fill in the blank and V is the dictionary of all the words in our dataset. $\hat{b}$ is the prediction (the word to be filled in the blank), and θ is the model parameters. Our framework is depicted in Figure 4.1. The proposed approach consists of four components: 1) source sentence encoding, 2) spatial attention model, 3) temporal attention model, and 4) inference of the missing word. We discuss these four components in the following subsections.

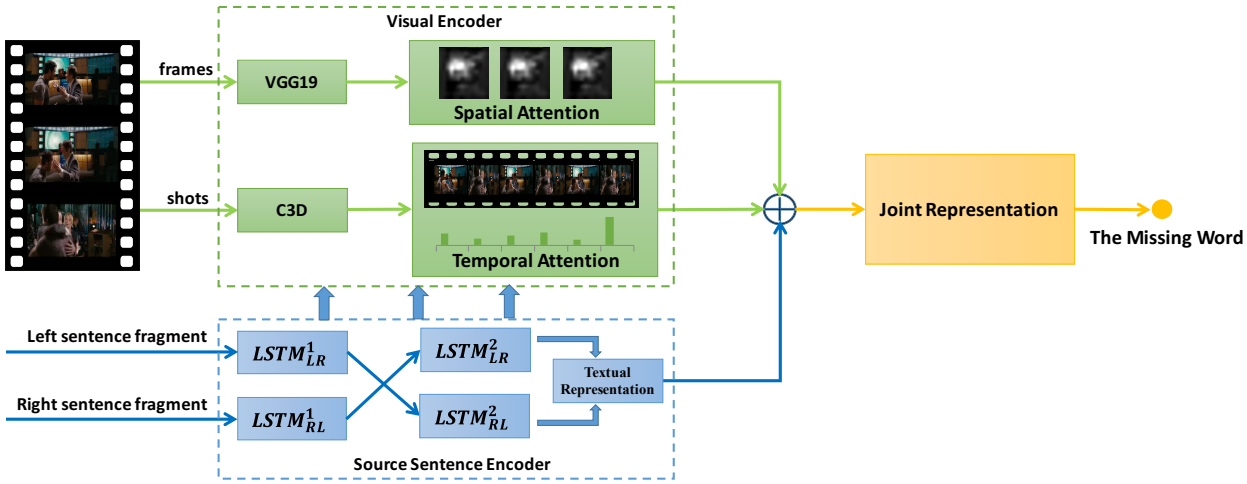


Figure 4.1: Our proposed method to solve Video Fill In the Blank (VFIB) problem. Source sentence encoding, spatial and temporal attention models are shown.

4.1.1 Source Sentence Encoding

In this section, we introduce how to encode the source sentence which consists of left and right fragments and a blank between them. Figure 4.2 shows an illustration of the proposed source sentence encoding approach. The blank can be anywhere in a source sentence, thus a single LSTM architecture will not work very well, since left or right fragment can be very short, and in many cases the missing word depends on both fragments. However, a simple LSTM will fail if one fragment is very short or both fragments are needed for prediction. Also, the blank can belong to

any classes of words (e.g. verb, adjective, etc.). These complexities of the source sentence make the textual encoding difficult.

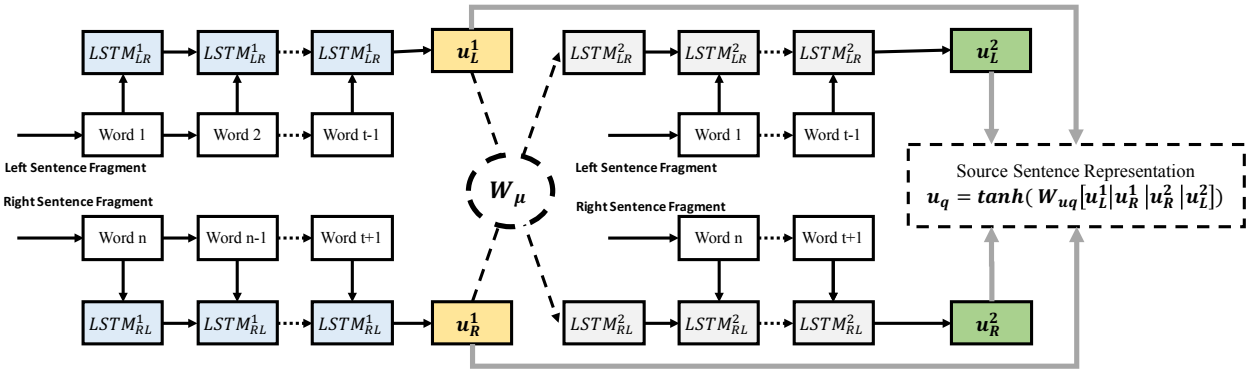


Figure 4.2: An illustration of the source sentence encoding approach. In the first stage, Each source sentence is formatted as two fragments, the left and right sentence fragments. Each sentence fragment is passed through an LSTM. In the second stage, left and right fragments' LSTMs are combined with a memory from the opposite side.

We treat the source sentence as two fragments; the left fragment from the first word to the word before the blank and the right fragment backward from the last word to the word after the blank. Our source sentence encoding module has three stages. In the first stage, we encode each of the left and right fragments separately with two independent LSTMs. In the second stage, we encode left and right fragments along with the encoded fragments from the opposite side in stage one. Namely, we use the encoded left fragment in the first stage as an external memory to encode the right fragment in stage two and vice versa. We call it "external memory", since it is computed using the opposite fragment. In fact, the external memory makes the model to understand each fragment better, since it has some information from the opposite fragment. Finally, the model learns to combine the output of both stages and generates the final source sentence's representation called u_q (Figure 4.2). Our approach has two major differences compared to BiLSTM [144, 30]. First, we use the the opposite fragments as an external memory for each sides, and second, our method learns how to combine the left and right encoded fragments. In the following, we provide

more details.

Assume that the source sentence has n words, and the t'th word is missing. The left and right fragments' sequences can be embedded as:

$$\begin{aligned} \boldsymbol{q}_l^1 &= \boldsymbol{W}_x^1[\boldsymbol{x}^1, \boldsymbol{x}^2, ..., \boldsymbol{x}^{t-1}] \\ \boldsymbol{q}_r^1 &= \boldsymbol{W}_x^1[\boldsymbol{x}^n, \boldsymbol{x}^{n-1}, ..., \boldsymbol{x}^{t+1}], \end{aligned} \tag{4.2}$$

where $\boldsymbol{x}^i \in \{0,1\}^{|V|}$ is an one-hot vector representation of i'th word in the source sentence, and $\boldsymbol{W}_x^1 \in \mathbb{R}^{c \times |V|}$ is a word embedding matrix ($|V|$ is the size of the dictionary, and c is the encoding dimension). $\boldsymbol{q}_l^1$ and $\boldsymbol{q}_r^1$ are two sequences where each element $q_{l_{(j)}}^1 \in \mathbb{R}^c$ is a continuous vector representing j'th word in q_l^1 sequence. We model the left and right sentence's fragments separately using two LSTMs:

$$\begin{aligned} \boldsymbol{u}_l^1 &= LSTM_{LR}^1(\boldsymbol{q}_{l_{(i)}}^1), \quad (i = 1, ..., (t-1)) \\ \boldsymbol{u}_r^1 &= LSTM_{RL}^1(\boldsymbol{q}_{r_{(i)}}^1), \quad (i = 1, ..., (n-t)) \end{aligned} \tag{4.3}$$

where $\boldsymbol{u}_l^1, \boldsymbol{u}_r^1 \in \mathbb{R}^h$ are the last hidden states from the "LR" and "RL" LSTMs respectively (Fig. 4.2) and h is the LSTMs' hidden state size. Since the missing word is related to both left and right fragments of a sentence, we have an extra stage to encode Left/Right fragments with respect to an external memory coming from the opposite side, namely Right/Left fragments. In this way, the first stage processes each of fragments separately and the second stage processes them with respect to opposite side's encoded representation.

$$\begin{aligned} \boldsymbol{q}_l^2 &= [\boldsymbol{\mu}_l, \boldsymbol{W}_x^2[\boldsymbol{x}^1, \boldsymbol{x}^2, ..., \boldsymbol{x}^{t-1}], \boldsymbol{\mu}_l] \\ \boldsymbol{q}_r^2 &= [\boldsymbol{\mu}_r, \boldsymbol{W}_x^2[\boldsymbol{x}^n, \boldsymbol{x}^{n-1}, ..., \boldsymbol{x}^{t+1}], \boldsymbol{\mu}_r], \end{aligned} \tag{4.4}$$

where $\boldsymbol{W}_x^2$ is of the same size as $\boldsymbol{W}_x^1$ and $\boldsymbol{\mu}_r, \boldsymbol{\mu}_l \in \mathbb{R}^c$ are two external memory vectors obtained

by:

$$\begin{aligned} \boldsymbol{\mu}_r &= \boldsymbol{u}_l^1 \boldsymbol{W}_\mu \\ \boldsymbol{\mu}_l &= \boldsymbol{u}_r^1 \boldsymbol{W}_\mu \end{aligned} \tag{4.5}$$

where $\boldsymbol{W}_\mu \in \mathbb{R}^{h \times c}$ encodes the LSTMs outputs to memory vectors. $\boldsymbol{q}_l^2$ and $\boldsymbol{q}_r^2$ are two sequences while $|\boldsymbol{q}_l^2| - |\boldsymbol{q}_l^1| = |\boldsymbol{q}_r^2| - |\boldsymbol{q}_r^1| = 2$ because of the external memory vectors attached to them and each element of them is a continuous vector. Similar to Eq. 4.3, we encode these two sequences with two different LSTMs, namely $LSTM_{LR}^2$ and $LSTM_{RL}^2$ (Fig 4.2), to obtain $\boldsymbol{u}_l^2, \boldsymbol{u}_r^2 \in \mathbb{R}^h$ as encoded left and right fragments in the second stage. Note that; since $\boldsymbol{W}_x^1$ and $\boldsymbol{W}_x^2$ are two different matrices, LR/RL LSTMs of first and second stages, observe completely different sequence of vectors. Also, none of these four LSTMs share any parameters with the other ones.

Finally, a proper combination of $\boldsymbol{u}_l^1$, $\boldsymbol{u}_r^1$, $\boldsymbol{u}_l^2$ and $\boldsymbol{u}_r^2$ as the final representation of the source sentence is needed. We concatenate and combine them by a fully connected layer as the final representation of the source sentence:

$$\boldsymbol{u}_q = \tanh(\boldsymbol{W}_{uq}[\boldsymbol{u}_l^1|\boldsymbol{u}_r^1|\boldsymbol{u}_l^2|\boldsymbol{u}_r^2]), \tag{4.6}$$

where $\boldsymbol{W}_{uq} \in \mathbb{R}^{d \times 4h}$ is a trainable weights matrix applied to learn a proper combination of four vectors. We refer $\boldsymbol{u}_q \in \mathbb{R}^d$ as the source sentence representation (textual feature) in the following sections and it is a bounded vector due to $\tanh(\cdot)$ activation function.

4.1.2 Spatial Attention Model

We use a CNN (i.e., VGG-19 [58], more details are provided in Section 4.2.4) to extract visual features. The output from the last pooling layer of CNNs can be considered as a feature map with spatial information about the input image. Figure 4.3 shows an illustration of the spatial attention

model. First, we apply max-pooling over the raw CNN features (i.e the output from the last pooling layer of VGG-19 pre-trained network) from all video key-frames to obtain a spatial visual feature from the whole video:

$$\boldsymbol{\Phi}_F = \tanh(\boldsymbol{W}_f \Theta(\boldsymbol{\Phi}_f(f^t))|_{f^t \in F}), \tag{4.7}$$

where $\boldsymbol{\Phi}_f(\cdot)$ is the spatial visual feature map extraction function (more details are provided in Section 4.2.4), F represents all the video key-frames in v, f_t is a video frame at time t, $\Theta(\cdot)$ is the max-pooling function, $\boldsymbol{W}_f$ is a trainable transformation matrix, and $\boldsymbol{\Phi}_F \in \mathbb{R}^{d \times m}$ is the intermediate visual feature matrix where each column is a feature vector corresponding to a spatial region in the original video frames, d is the same as $\boldsymbol{u}_q$ in Eq.4.6, and m is the number of spatial regions in the video frame.

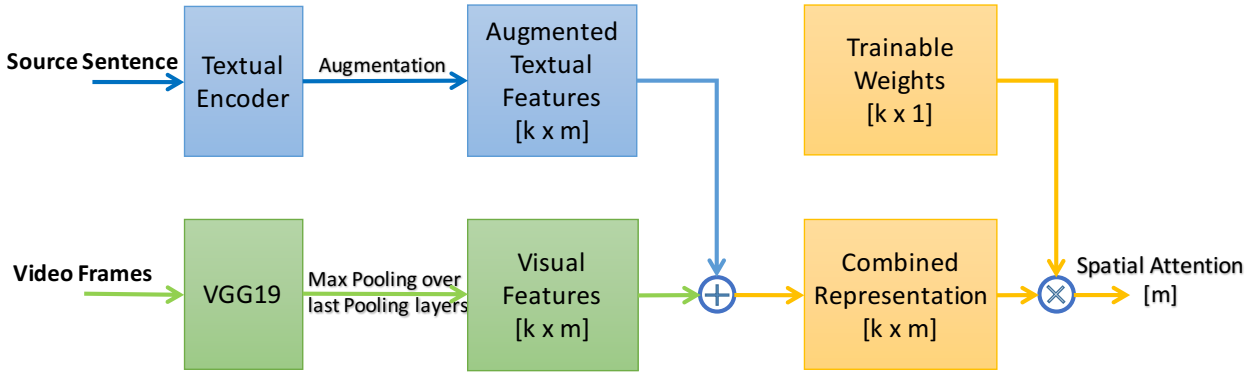


Figure 4.3: An illustration of the spatial attention model. The model assigns importance score to each region of an image based on the source sentence.

We use spatial attention model [145] to pool the intermediate visual features $\boldsymbol{\Phi}_F$. The first step is to combine the source sentence representation $\boldsymbol{u}_q$ with the intermediate visual features $\boldsymbol{\Phi}_F$:

$$\boldsymbol{\Psi}_F = \tanh((\boldsymbol{W}_F \boldsymbol{\Phi}_F) \oplus (\boldsymbol{W}_u \boldsymbol{u}_q + \boldsymbol{b}_u)), \tag{4.8}$$

where $\boldsymbol{W}_F \in \mathbb{R}^{k \times d}$ and $\boldsymbol{W}_u \in \mathbb{R}^{k \times d}$ are two transformations on the intermediate visual features

and source sentence representation to align them and have the same dimensions. $\boldsymbol{b}_u$ is the bias term, and $\oplus$ is a summation between a matrix and an augmented vector (i.e. the source sentence representation $\boldsymbol{u}_q$ has to be expanded, or repeated m times, in order to have the same dimension as $\boldsymbol{W}_F\boldsymbol{\Phi}_F$. The matrix $\boldsymbol{\Psi}_F \in \mathbb{R}^{k\times m}$ is used to find the final attention scores over all the regions:

$$\boldsymbol{p}_{sp} = \text{softmax}(\boldsymbol{\Psi}_F^T \boldsymbol{w}_{sp}), \tag{4.9}$$

where $\boldsymbol{w}_{sp} \in \mathbb{R}^{k\times 1}$ is a trainable weight vector and $\boldsymbol{p}_{sp} \in \mathbb{R}^{m\times 1}$ is the spatial attention vector. The final spatial pooled visual vector is a weighted average over all m regional intermediate spatial feature vectors:

$$\boldsymbol{u}_{sp} = \Phi_F \boldsymbol{p}_{sp}, \tag{4.10}$$

where $\boldsymbol{u}_{sp} \in \mathbb{R}^d$ is the spatial pooled visual vector. It is a bounded vector since $\boldsymbol{\Phi}_F$ is bounded.

4.1.3 Temporal Attention Model

Temporal dynamics of a video and the motion information plays a significant role in video understanding, especially in actions and events understanding. However, in our spatial-pooled visual representation presented in previous section, the whole video is represented as one vector, and there is no temporal information. Therefore, we propose to model the temporal dynamics of the video using a temporal attention model. Figure 4.4 shows an illustration of this component in our approach. We divide a video into a number of shots and represent the shots as below:

$$\boldsymbol{\Phi}_G = \tanh(\boldsymbol{W}_g[\boldsymbol{\Phi}_g(g^1), \boldsymbol{\Phi}_g(g^2), ..., \boldsymbol{\Phi}_g(g^{|G|})]), \tag{4.11}$$

where $\boldsymbol{\Phi}_g(\cdot)$ is the feature extraction function (C3D [146], which encodes the temporal information. More details are provided in Section 4.2.4), G represents the set of video shots and g^i is the

ith video shot. $\boldsymbol{W}_g \in \mathbb{R}^{d \times z}$ is a transformation matrix, where z is the original dimension of the feature vector $\boldsymbol{\Phi}_g(g^i)$, and k is the encoding dimension.

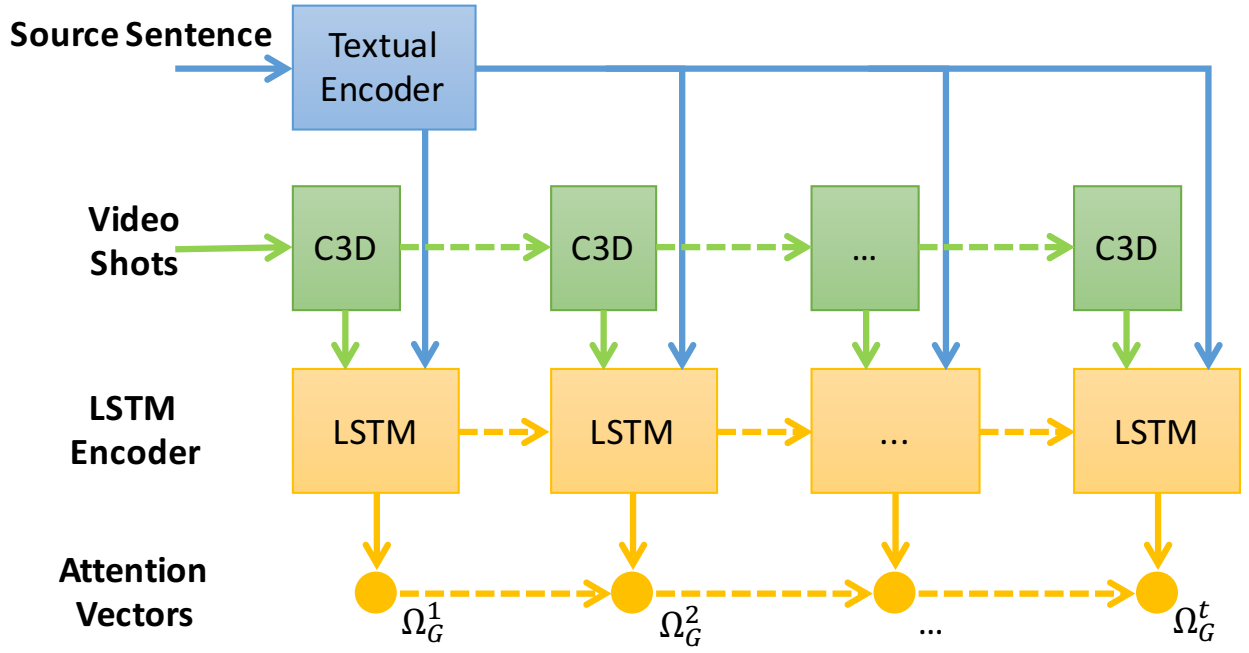


Figure 4.4: An illustration of the temporal attention model. An LSTM is used to find relevant shots based on the source sentence.

We combine the video shots representation $\boldsymbol{\Phi}_G \in \mathbb{R}^{d \times |G|}$ and the source sentence representation $\boldsymbol{u}_q$ as:

$$\boldsymbol{\Psi}_G = \tanh((\boldsymbol{W}_G \boldsymbol{\Phi}_G) \oplus (\boldsymbol{W}_u \boldsymbol{u}_q + b_u)), \tag{4.12}$$

where $\boldsymbol{W}_G \in \mathbb{R}^{k \times d}$, is a mapping for shot representation $\boldsymbol{\Phi}_G$. We share $\boldsymbol{W}_u$ and $\boldsymbol{b}_u$ with the spatial attention model in Eq. 4.8. Similar to Eq. 4.8, in order to apply the summation $\oplus$, $\boldsymbol{u}_q$ is repeated $|G|$ times to have the same number of columns as $\boldsymbol{\Phi}_G$. Each column of matrix $\boldsymbol{\Psi}_G \in \mathbb{R}^{k \times |G|}$ is the combination of a single shot and the source sentence. An LSTM is employed to model the

dynamics between shots and the source sentence:

$$\mathbf{\Omega}_G^i = \text{LSTM}(\Psi_G^i) \quad (i = 1...|G|), \tag{4.13}$$

where $\mathbf{\Psi}_G^i$ is the i'th column of $\mathbf{\Psi}_G$. The output of this LSTM is a sequence of attention vectors corresponding to each shot $\mathbf{\Omega}_G = [\Omega_G^1, \Omega_G^2, ..., \Omega_G^{|G|}]$ (See Fig. 4.4). However, we need to make probabilities out of these vectors and for this purpose we simply use a $softmax$ operator:

$$\boldsymbol{p}_{tp} = softmax(\mathbf{\Omega}_G \boldsymbol{w}_{tp}), \tag{4.14}$$

where $\boldsymbol{w}_{tp} \in \mathbb{R}^{k \times 1}$ is a trainable weight vector and $\boldsymbol{p}_{tp} \in \mathbb{R}^{|G| \times 1}$ is the temporal attention of the shots and the final temporal-pooled representation is a weighted average over all the shot features $\mathbf{\Phi}_G$ with the attention model:

$$\boldsymbol{u}_{tp} = \mathbf{\Phi}_G \boldsymbol{p}_{tp}, \tag{4.15}$$

where $\boldsymbol{u}_{tp} \in \mathbb{R}^d$ is of the same dimension as the spatial-pooled features and the source sentence representation. Its values are also bounded since $\mathbf{\Phi}_G$ is obtained by passing through a $\tanh(\cdot)$ activation in Eq. 4.11. Using this temporal attention model, we capture the dynamics of the visual features which are related to the source sentence representation.

4.1.4 Inference of the Missing Word

Here, we discuss how to infer the missing word or fill in the blank. Let β be the vocabulary to fill in the blank ($|\beta|$ as its size). We aim to find a probability for each word candidate in β, which needs a joint representation (summation fusion [145]) of all three components as mentioned earlier:

$$\boldsymbol{u} = [\boldsymbol{u}_q + \boldsymbol{u}_{sp} + \boldsymbol{u}_{tp}], \tag{4.16}$$

where $\boldsymbol{u} \in \mathbb{R}^d$ is an joint representation of all three features: source sentence representations, spatially- and temporally-pooled visual representations. Note that all three vectors $\boldsymbol{u}_q$, $\boldsymbol{u}_{sp}$ and $\boldsymbol{u}_{tp}$ in equations 4.6, 4.10 and 4.15 are bounded, since they have been obtained by passing through a $\tanh(\cdot)$ activation. They also have the same dimension d and this makes the summation fusion [145] applicable and effective. For the final inference of the missing word, we compute a probability of each candidate word as follows:

$$P_{blank} = softmax(\boldsymbol{W}_{blank}\boldsymbol{u}), \tag{4.17}$$

where $\boldsymbol{W}_{blank} \in \mathbb{R}^{|\beta|\times d}$. This is followed by a multinomial logistic regression "$softmax$" to find the probabilities vector $\boldsymbol{P}_{blank} \in \mathbb{R}^{|\beta|}$. Based on Eq. 4.1, the final answer is:

$$\hat{b} = \arg\max_{b\in\beta} P_{blank}(b). \tag{4.18}$$

4.2 Results

We perform experiments on two datasets: the original LSMDC Dataset [88] to evaluate the single blank VFIB problem (i.e. there is only one blank in the source sentence), and an extended LSMDC Movie Dataset to evaluate our performance on the multiple blanks VFIB problem (i.e. there are multiple blanks in the sentence).

4.2.1 LSMDC Movie Dataset (Single Blank)

In this set of experiments, we use the movie dataset [88, 1, 147], which has been used in Large Scale Movie Description and Understanding Challenge (LSMDC) [88]. Movies are a rich source of visual information and become much more valuable when proper textual meta-data is provided.

Movies benefit from many textual data like the subtitle, audio descriptions and also movie synopsis. LSMDC dataset consists of respectively "$91,908$", "$6,542$", "$10,053$" and "$9,578$" movie clips as Training, Validation, Public and Private Test sets. We use the standard splits provided by [88]. Each clip comes with a sentence annotated by an expert. There can be multiple source sentences built for one clip. We use respectively "$296,960$", "$21,689$" and "$30,349$" samples as training, validation and test as the standard split provided by [88].

4.2.1.1 Quantitative Results

Here, we compare our proposed method with other approaches and baselines to show its superior performance for the VFIB task. We have chosen some of these baselines from methods for visual question answering problem, which are applicable to this problem as well. The comparison table (Table 4.1) has four parts. The first part is for methods which only use the text to find the missing word; the second part is for methods which just use the video; the third part is for methods which use both text and video; and the last part is for different configurations of the proposed method. We report the accuracy (same as in [88]) of each method which is the ratio of number of missing words that are inferred correctly to the total number of blanks. Here are some details about these methods:

LSTM Left/Right Sentence fills the blank by just looking at the left/right fragment of the missing word. This experiment shows that both fragments are equally important. **BiLSTM** finds the missing word based on a BiLSTM [30], which encodes the input sentence using two different LSTMs; one takes the input from the last word to the first word and the other one in the reverse. The blank word is recovered based on BiLSTM's output in missing word location. **Our Sentence Embedding** as described in section 4.1.1. This approach finds the missing word by using just vector $\boldsymbol{u}_q$, without any visual features. For a fair comparison with BiLSTM method, we have fixed

the LSTM cells sizes and also the word embedding lengths in all the experiments. The authors in [88] report a few baselines using **GoogleNet** [59] and **C3D** [146] features. The difference between the baselines in[88] and ours, shows the actual importance of our attention models and integration of the textual encoding and visual modules in our method. **Video+ Textual Encoding** corresponds to "IMG+LSTM" in [29]. Key-frames are passed through the VGG-19 pre-trained network to extract $4,096$ dimensional vector of "fc7" layer for each key-frame. Then, a max-pooling over all the features of all frames will generate a video feature vector and the rest of steps are the same as explained in [29]. We have used simple BiLSTM instead of LSTM to deal with two fragments of sentence in VFIB. **2Videos+ Textual Encoding** [29], similar to previous case, uses two different representations of the video. One is attached to the beginning of each fragment and the other one to the end. **Ask Your Neurons** [76] encodes the visual CNN feature and concatenate with each of words and pass them through left and right LSTMs one by one. The answer is inferred based on the last output of LSTMs. **SNUVL** [148] is the best reported method on LSMDC FIB. It uses a concept detection method over the videos, following by an attention model over the detected concepts, to find the missing word. **Ensemble model** [149] is a technique to boost the performance, when the optimization process reaches different local optima, based on random factors like initialization. We train the model multiple times with different initializations and sum the final scores from all the trained models.

We also test our model performance by removing each of components, namely spatial attention, temporal attention and also replacing our source sentence encoding with the BiLSTM in baselines. We also show the results of our text encoder **without Second Stage**, by removing u_l^2 and u_r^2 from Eq. 4.6. In one more complementary experiment, we try our textual encoding with C3D and VGG19 features without any attentions. These experiments show that all the components contribute to final results.

4.2.2 Multiple Blanks VFIB

In this section, we explore a harder version of the VFIB problem where more than one word is missing from the sentence. We have generated a new dataset based on the original LSMDC Movie Dataset by inserting multiple blanks in the sentences, and we call it the "Extended LSMDC Movie Dataset". To be specific, we remove all the words which have appeared at least once as a blank in the original LSMDC dataset from all the sentence. In this case, most of sentences have more than one blank and this makes the LSMDC dataset suitable to be extended for multiple blanks problem. In Figure 4.5, we show some statistics about the number of blanks in sentences. About 79.3% of the sentences have more than one blank. To clarify, in each sentence, there are known number of blanks (with known locations), but there are various number of blanks in different sentences. For multiple blanks' experiments, we include all the sentences with one or more blanks in all sets and also for the evaluation, we consider equal value for all the blanks.

We employ two strategies to encode the source sentence for the multiple blanks VFIB problem. For the first one, we consider the left and right fragments of a missing word, as a fragment from that word to the next blank, or if there is no other blank, to the end of the sentence (both left and right fragments are used). For example, for the source sentence *"She took her Blank1 out of the garage and Blank2 at the house for a moment."*. The left phrase of "Blank1" is *"She took her"* and right phrase is *"out of the garage and"* and we find the left and right phrases for "Blank2" with the same approach as well. We call it the **"Subdivision"** approach since it makes multiple fragments out of the source sentence and each blank has one left and one right fragment. The second approach is to remove all other blanks and treat them as left and right fragments as normal. In our example, the left fragment of "Blank2" is *"She took her out of the garage"* and the right fragment of the "Blank2" is *"out of the garage and at the house for a moment"*. In this case, we deal with each blank similar to single blank problem and we just ignore other blanks in each of left

and right fragments. We call it **"Masking"** approach since we are masking the other missing words from each fragment. After finding left and right fragments based on any of these approaches, we can apply our method or any other baselines (Table 4.2).

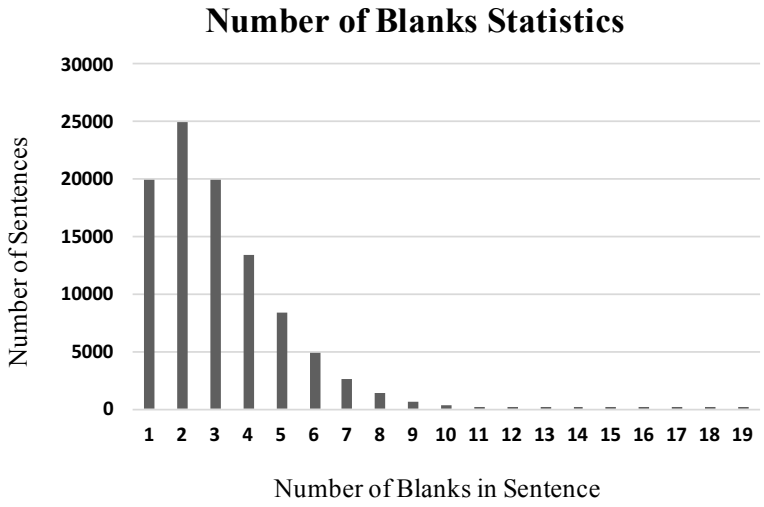


Figure 4.5: Number of sentences as a function of the number of blanks in training set of LSMDC.

4.2.3 Qualitative Results

In Fig. 4.6, we show some qualitative results. For generating the attention map, we have reshaped the $\boldsymbol{p}_{sp} \in \mathbb{R}^{m=196}$ in Eq. 4.9 into a 14×14 matrix, then up-sampled it back to the original frames size. We smooth the attention map by a Gaussian filter and also suppress low intensity pixels to be zero. Brighter parts have higher attention score than the darker parts. For the temporal attention model, we extract the $\boldsymbol{p}_{tp}$ vector as the temporal attention. In Fig. 4.6, we show one frame from each shot and the color bar under the sequence of shots shows the attention scores. Yellow and blue respectively represent the maximum and minimum attentions. In Fig. 4.7 we provide examples of multiple blanks VFIB and predicted missing words using different methods.

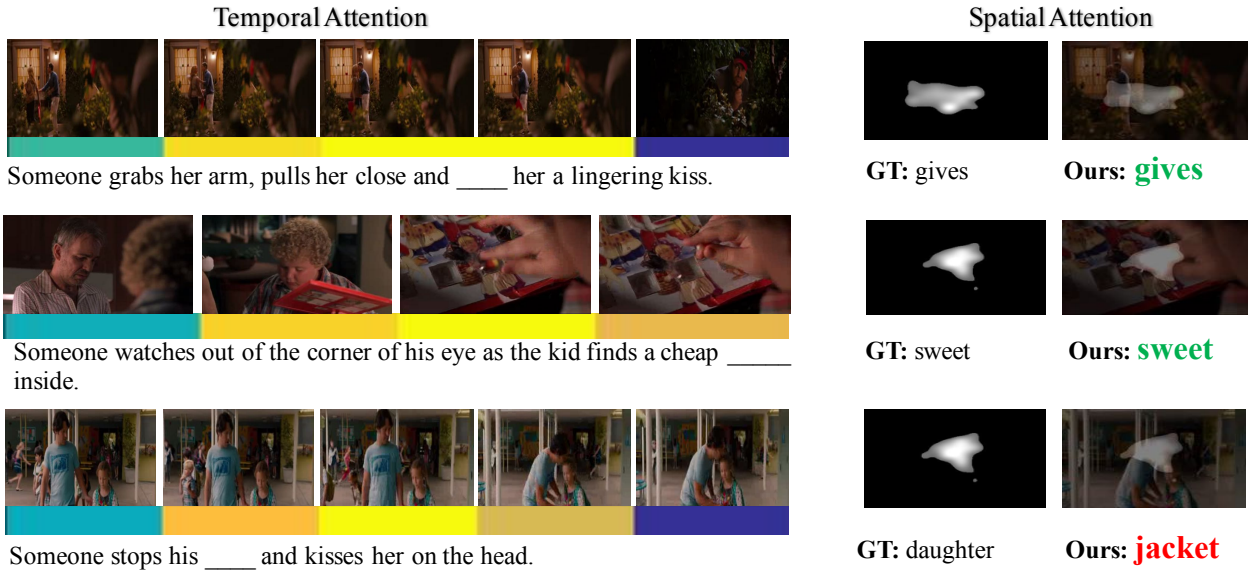


Figure 4.6: On the left we show representative frames from different shots. The colors below the frames show the temporal attention: yellow/blue means the most/least attention. On the right, we show an spatial attention map obtained by our method and also we show the attention map on one of selected key-frames.

Someone _____ and _____ his _____.

Cutting:

2Videos – Textual Encoding	(turns, faces, gaze)
Our Model	(smiles, raises, wife)

Masking:

2Videos – Textual Encoding	(smiles, shakes, gaze)
Our Model	(smiles, shakes, head)
Ground Truth:	(smiles, shakes, head)

In his _____, someone _____ at his _____ glowering.

Cutting:

2Videos – Textual Encoding	(room, sits, desk)
Our Model	(office, glances, desk)

Masking:

2Videos – Textual Encoding	(office, stares, desk)
Our Model	(office, sits, computer)
Ground Truth:	(office, sits, desk)

Figure 4.7: Examples for Multiple Blank VFIB problem which requires a higher level of video and text alignment to find all the missing words correctly.

4.2.4 Implementation Details

We use VGG-19 [150] network, pre-trained on ImageNet [61], and extract last pooling layer ("pool5") as our spatial visual features consumed in section 4.1.2. The output feature map is a $14 \times 14 \times 512$ matrix which can be reshaped as a 196×512 matrix and each of 512 dimensional vectors are representing a 32×32 pixels region of input frame. We believe any other very deep CNN network like GoogLeNet [59] or ResNet [60] can produce similar results. We extract and pass the frames through this network with 2fps rate. For temporal attention in section 4.1.3, we use pre-trained 3D CNN (C3D) network [146] pre-trained on [151] and followed settings defined in [146]. We extract the "fc6" output of the network for each 16 frames (one shot) of videos. We assume each video has 10 shots. For shorter videos, we use all-zero vectors for remaining shots and for longer ones we uniformly select 10 shots.

Table 4.1: Results on "Movie Fill-in-the-Blank" dataset.

Method	**Accuracy**
Text Only	
Random Guess	0.006
LSTM Left Sentence	0.155
LSTM Right Sentence	0.165
BiLSTM	0.320
Our Sentence Encoding (w/o Second Stage)	0.340
Our Sentence Encoding	**0.367**
Human [88]	*0.302*
Video Only	
BiLSTM Just Video	0.055
Text + Video	
GoogleNet-2D [88]	0.349
C3D [88]	0.345
GoogleNet-2D-Finetuned [88]	0.353
GoogleNet-2D + C3D-Finetuned [88]	0.357
Video + Textual Encoding [29]	0.341
2Videos + Textual Encoding [29]	0.350
Ask Your Neurons [76]	0.332
SNUVL [148]	0.380
SNUVL (Ensembled Model) [148]	0.407
Human [88]	*0.687*
Ours	
Single Model (VGG19 + C3D w/o Attention)	0.378
Single Model (w/o Spatial Attention)	0.390
Single Model (w/o Temporal Attention)	0.392
Single Model (w/o Second Stage)	0.396
Single Model (w/o LR/RL LSTMs)	0.387
Single Model	**0.406**
Ensembled Model	**0.434**

Table 4.2: Results on the LSMDC Dataset (Multiple Blanks). Our method has superior results.

Method	**Accuracy**
Baselines	
Random Guess	0.006
Left LSTM (Masking)	0.104
Bi-LSTM (Masking)	0.156
2Videos + Textual (Subdivision) [29]	0.136
2Videos + Textual (Masking) [29]	0.177
Ours	
Text Only (Subdivision)	0.136
Text + Video (Subdivision)	0.148
Text Only (Masking)	0.180
Text + Video (Masking)	**0.201**

4.3 Summary

We proposed a new method for the Video-Fill-in-the-Blank (VFIB) problem which leverages the "source sentence" structure and also spatial-temporal attention models. We have introduced "external memory" to deal with the complexity of "source sentence" in VFIB problem. We have achieved superior performance over all other reported methods. Also, an extension and more general version of VFIB which deals with multiple blanks in sentences, is introduced and discussed.

CHAPTER 5: VISUAL TEXT CORRECTION

The work in this Chapter have been published in the following paper:

Mazaheri, Amir, and Mubarak Shah. "Visual Text Correction." Proceedings of the European Conference on Computer Vision (ECCV). 2018. [152]

In this Chapter, we formulate the Visual Text Correction (VTC) task which is one of the real-world use cases of the VFIB. In fact, we decompose the VTC problem into an inaccurate word detection, and a correct word prediction to replace the inaccurate word. The second part of the approach is essentially the VFIB. However, our proposed approach is end-to-end trainable meaning that we train both of inaccuracy detection, and correct word prediction sub-modules simultaneously.

5.1 Methodology

To formulate the VTC problem, assume $\widetilde{\mathcal{S}} = [\widetilde{w}_1, \widetilde{w}_2, ..., \widetilde{w}_N]$ is a given sentence for the video $\mathcal{V}$. Our aim is to find the index of the incorrect word, t^*, and correct it with ${w^*}_{t^*}$ as follows:

$$(t^*, w^*_{t^*}) = \arg\max_{1 \leq t \leq N, w_t \in \beta} p((t, w_t) | \widetilde{\mathcal{S}}, \mathcal{V}), \tag{5.1}$$

where $w_i \in \{0, 1\}^{|V|}$ is an one-hot vector representing the $i'th$ word of the sentence, $|V|$ is the size of our dictionary and N is the length of the sentence. Also, $\beta \subseteq V$ represents the set of all potential substitution words. Since t^* and ${w^*}_{t^*}$ are sequentially dependent, we decompose the Equation 5.1

into two sub-tasks: Inaccurate word detection as:

$$t^* = \arg\max_{1 \leq t \leq N} p(t|\widetilde{\mathcal{S}}, \mathcal{V}), \tag{5.2}$$

and the accurate word ${w^*}_{t^*}$ prediction as:

$${w^*}_{t^*} = \arg\max_{w \in \beta} p(w|\widetilde{\mathcal{S}}, \mathcal{V}, t^*\ ). \tag{5.3}$$

5.1.1 Inaccuracy Detection

We propose detection by reconstruction method to find the most inaccurate word in a sentence, leveraging the semantic relationship between the words in a sentence. In our approach, each word of a sentence is reconstructed such that the reconstruction for the inaccurate word is maximized. For this purpose, we build embedded word vector $x_i \in \mathbb{R}^{d_x}$ for each corresponding word w_i using a trainable lookup table $\theta_x \in \mathbb{R}^{|V| \times d_x}$. We exploit both Short Term and Long Term Dependencies employing respectively Convolutional N-Grams and LSTMs to reconstruct the word vectors.

5.1.1.1 Short-Term Dependencies:

Convolutional N-Gram networks[82] capture the *short-term* dependencies of each word surrounding. Sentences can vary in length, and a proper model should not be confused easily by long sentences. The main advantage of N-Gram approach is its robustness to disrupting words in long sentences, since it considers just a neighboring block around each word.

Let $X = [x_1; x_2; \ldots; x_N]$ be the stacked vectors representing embedded word vectors. Since the location of each word provides extra information about the correctness of that word in a sentence,

we combine it with word vectors X. We denote $p_t \in \mathbb{R}^{d_x}$ as an embedded vector associated to the t'th *position* of each sentence, which is one row of the trainable matrix, $P \in \mathbb{R}^{N \times d_x}$. We use p_t values as gates for the corresponding word vectors x_t for each sentence and get final combination I as:

$$I_t = x_t \odot \sigma(p_t), \tag{5.4}$$

where $\odot$ denotes element-wise multiplication, and $I \in \mathbb{R}^{N \times d_x}$ is the input to a 1-D convolution with $2d_x$ filters and receptive field size of m. We call the resulting activation vectors $C \in \mathbb{R}^{N \times 2d_x}$. Furthermore, we use Gated Linear Units (GLU) [83] as the non-linear activation function. First, we split the C matrix in half along its depth dimension:

$$\begin{aligned} [A, B] &= C, \\ \Phi &= A \odot \sigma(B), \end{aligned} \tag{5.5}$$

where $A, B \in \mathbb{R}^{N \times d_x}$, and $\Phi = [\phi_1; \phi_2; \ldots; \phi_N]$, and $\phi_i \in \mathbb{R}^{d_x}$. The idea is to use the B matrix as gates for the matrix A. An open gate lets the input pass, and a close gate changes the input to zero. By stacking multiple 1-D convolutions and GLU activation functions the model goes deeper and the receptive field becomes larger. The output, Φ, from each layer is the input, I, for the next layer. We call the final output Φ, from the last Convolutional N-Grams layer, $\hat{X}^C \in \mathbb{R}^{N \times d_x}$. In Figure 5.1, we illustrate one layer of the N-Grams encoding.

5.1.1.2 Long-Term Dependencies:

Recurrent networks, and specifically LSTMs, have been successfully used to capture the *long-term* relations in sequences. Long-term relations are beneficial to comprehend the meaning of a

text and also to find the possible inaccuracies. To reconstruct a word vector based on the rest of the sentence using LSTMs, we define a left fragment and a right fragment for each word in a sentence. The left fragment starts from the first word of the sentence to one word before the word under consideration; and the right fragment is from the last word of the sentence to one word after the word under consideration in a reverse order. We encode each of the left and right fragments with a LSTM and extract the last hidden state vector of the LSTM as the encoded fragment:

$$\hat{x}_t^R = W_c \times [u_t^l | u_t^r], \tag{5.6}$$

where $u_t^{l/r} \in \mathbb{R}^h$ are the encoded vectors of left/right fragments of the t'th word, and $W_c \in \mathbb{R}^{d_x \times 2h}$ is a trainable matrix to transform the $[u_t^l | u_t^r]$ into the $\hat{x}_t^R$.

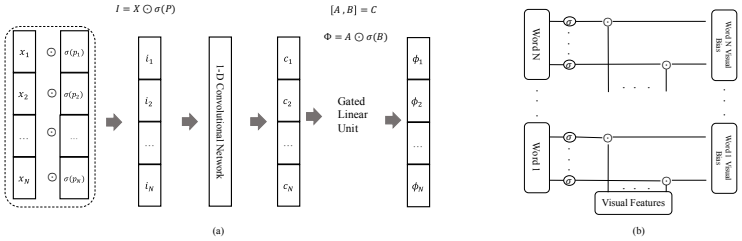


Figure 5.1: (a) One layer of Convolutional Text Encoding which captures the neighboring relationships. To extend one layer to multiple layers, we simply consider the ϕ_i vectors as I_i for the next layer. (b) Our proposed Visual Gating Bias process. Given each word vector, we filter out some parts of a given visual feature through a gating process.

5.1.1.3 Detection Module:

We design a module to learn the distance between an actual word vector x_t and the reconstructed $\hat{x}_t$ as explained above. This module learns to assign a larger distance to the inaccurate words and

reconstruct the predictions as follows:

$$\mathcal{D}_t = W_d \times (\frac{\hat{x}_t}{\|\hat{x}_t\|} \odot \frac{x_t}{\|x_t\|}), \tag{5.7}$$

where $W_d \in \mathbb{R}^{1 \times d_x}$, and $\mathcal{D}_t$ is a scalar. $\hat{x}_t$ is the output of the text encoding; namely, $\hat{x}_t = \hat{x}_t^C$ for Convolutional N-Grams or $\hat{x}_t = \hat{x}_t^R$ in case of Recurrent Networks. Next, we combine both as a vector $\hat{x}_t = \hat{x}_t^R + \hat{x}_t^C$ to capture both long term and short term dependencies of a sentence. We design our distance module as a single layer network for simplicity; however, it can be a deeper network.

5.1.1.4 Visual Features as Gated Bias:

Visual features can contribute in finding the inaccuracy in a video description; however, it can be very challenging since some words may not correspond to any visible form or shape (e.g. 'weather'), while some others may correspond to distinct visual appearances (e.g. 'cat'). We introduce a gating model to incorporate the visual features to measure the inconsistency of each word. The main idea is to find a dynamic vector for the visual features which changes for each word as follows (see Figure 5.1):

$$\Psi_{\mathcal{V}} = W_v \times \Omega(\mathcal{V}), \tag{5.8}$$

where $\Omega(\mathcal{V}) \in \mathcal{R}^{d_v}$ is the visual feature vector, and $W_v \in \mathcal{R}^{d_x \times d_v}$ is a transformation matrix for the visual features. We build the visual bias v_t for each word vector x_t:

$$v_t = \frac{\Psi_{\mathcal{V}}}{\|\Psi_{\mathcal{V}}\|} \odot \sigma([W_g \times x_t]), \tag{5.9}$$

and $W_g \in \mathcal{R}^{d_x \times d_x}$ is transformation matrix, and $\|.\|$ denotes L2-Norm of a vector. The Sigmoid ($\sigma(.)$) operator bounds its input into $(0, 1)$. It makes the model capable of refusing or accepting

visual features dynamically for each word in a sentence.

The most intuitive way to incorporate the V vectors in Equation 5.7, is to use them as a bias term. In fact, the features which are refused by the word gates will have zero value and will act as neutral. Therefore, we use the following updated form of Equation 5.7 with the video contribution:

$$\mathcal{D}_t = W_d \times (\frac{\hat{x}_t}{\|\hat{x}_t\|} \odot \frac{x_t}{\|x_t\|} \oplus v_t), \tag{5.10}$$

where $\oplus$ denotes element-wise summation.

For the last step of the detection process, we find the word with maximum $\mathcal{D}$ value:

$$t^* = \arg\max_{1 \leq t \leq N} (\mathcal{D}_t). \tag{5.11}$$

5.1.1.5 Detection loss:

We use the cross-entropy as detection loss function. Given the ground-truth one-hot vector $y \in \{0, 1\}^N$, which indicates the inaccurate word, and the $T^* = softmax(D)$ as probabilities, we compute the detection loss l_d.

5.1.2 Correct Word Prediction

The second stage of our proposed method to solve the VTC problem is to predict a substitute word for the inaccurate word. Proposed correct word prediction consists of three sub-modules: 1- Text Encoder, 2- Video Encoder, and 3- Inference sub-modules.

5.1.2.1 Text Encoder:

This sub-module must encode the input sentence in such a way that the network be able to predict the correct word for the t^*'th word. We leverage the reconstructed word vectors $\hat{x}_t$ in equation 5.7, since these vectors are rich enough to detect an inaccuracy by reconstruction error. We can feed the output of inaccuracy detection, t^*, to our accurate word prediction network; however, the $argmax$ operator in Equation 5.11 is not differentiable and prevents us to train our model End-to-End. To resolve this issue, we approximate the Equation 5.11 by vector $T^* = Softmax(D)$, which consists of probabilities of each of N words being incorrect in the sentence. We build the encoded text vector q_t:

$$q_t = tanh(W_q \times \hat{x}_t), \tag{5.12}$$

where $W_q \in \mathbb{R}^{d_q \times d_x}$ is trainable matrix. $q_t \in \mathbb{R}^{d_q}$ is in fact a hypothetical representation of the textual description. To be more specific, q_t is the encoded sentence, assuming that the word t is the incorrect word, which is to be replaced by a blank, according to the Equation 5.12. Finally, the **textual representation** $u_q \in \mathbb{R}^{d_q}$, is formulated as a weighted sum over all q_t vectors:

$$u_q = \sum_{t=1}^{N} T_t^* q_t. \tag{5.13}$$

Note that, due to the "$tanh(.)$" operator in Equation 5.12, both q_t and u_q vectors have bounded values.

5.1.2.2 Video Encoding:

We leverage the video information to find the accurate word for t^*'th word of a sentence. While the textual information can solely predict a word for each location, visual features can help it to predict

a better word based on the video, since the correct word can have a specific visual appearance. We extract the visual feature vector $\Omega(\mathcal{V})$ and compute our video encoding using a fully-connected layer:

$$u_{\mathcal{V}} = tanh(W_{\mathcal{V}} \times \Omega(\mathcal{V})), \tag{5.14}$$

where $W_{\mathcal{V}} \in \mathbb{R}^{d_q \times d_v}$, and $u_{\mathcal{V}} \in \mathbb{R}^{d_q}$ is our visual representation, which has bounded values. For simplicity, we have used just one layer video encoding; however, it can be a deeper and more complicated network.

5.1.2.3 Inference:

For the inference, we select the correct substitute word from the dictionary. In fact, this amounts to a classification problem, where the classes are the words and the inputs are the textual representation and the visual features:

$$w^*_{t^*} = \arg\max_{w \in \beta}(W_i \times [u_q + u_{\mathcal{V}}]), \tag{5.15}$$

where $W_i \in \mathbb{R}^{|\beta| \times d_q}$. Finally, we use cross-entropy to compute the correct word prediction loss, namely l_f. The total loss for our VTC method is $l = l_f + l_d$ and we train both sub-tasks together.

5.2 Results

5.2.1 Dataset

In this section, we describe our visual text correction dataset and the method to generate it. The main idea behind our approach to build a dataset for the VTC task is to remove one word from

each sentence and substitute it with an inaccurate word; however, there are several challenges to address in order to build a realistic dataset. Here, we list a few and also propose our approach to address those challenges.

Our goal is to build a large dataset with a variety of videos with textual descriptions. We require that the vocabulary of the dataset and the number of video samples be large enough to train a deep network; hence we choose "Large Scale Movie Description Challenge(LSMDC)" dataset [88, 1], which is one of the largest video description datasets available. Also, LSMDC has been annotated for "Video Fill In the Blank (FIB)" task. In FIB dataset, each video description contains one or more blanks, which needs to be filled in. For the VTC problem, we introduce inaccurate word in place of the blanks in FIB dataset. If there is more than one blanks in a sentence of the FIB dataset, we generate multiple examples of that sentence.

Note that there are some important points related to selection of the replacement words, which we need to keep in mind. First, there shouldn't be a high correlation between the original and replacement words. For example, if we exchange the word "car" with "bicycle" frequently, any method will be biased and will always suggest replacing "bicycle" with "car" in all sentences. Second, we want our sentences to look natural even after the word substitution. Therefore, the replacement word should have the same "Part Of Speech" (POS) tag. For example, a singular verb is better to be replaced by another singular verb.

It is costly to manually annotate and select the replacement words for each sample, because of the significant number of videos, and the vast vocabulary of the dataset. Also, it is hard for the human annotators to prevent the correlation between the original and replacement words. We have considered all the mentioned points to build our dataset. Following we describe how we build a proper dataset for the VTC problem.

5.2.1.1 Random Placement:

In this approach, for each annotated blank in the LSMDC-FIB dataset, we place a randomly selected word from dictionary. This approach evidently is the most straightforward and simple way to introduce the incorrect word. However, in this method, a bias towards some specific words may exist, since the selected inaccurate words may not follow the natural distribution of the words in the dictionary. For example, we have many words with less than 4 or 5 occurrences in total. By Random Placement approach, rare words and the words with high frequencies have the same chance to show up as an inaccurate word. This increases the rate of "inaccurate occurrences to accurate occurrences" for some specific words. This imbalanced dataset allows any method to detect the inaccuracy just based on the word itself not the the word in the context. Also, since replacement and original words may not take the same POS tag, Random Placement approach cannot meet one of the requirements mentioned above.

5.2.1.2 POS and Natural Distribution:

Due to the weaknesses of the Random Placement, we introduce a more sophisticated approach that selects the inaccurate words from a set of words with the same tag as the original (or accurate) word. We first extract the POS tags of all the words from all the sentences using Natural Language Toolkit (NLTK) [153], resulting in 32 tags. Let S_r be the set of all the words that takes the tag r ($1 \leq r \leq 32$) at least once in the training sentences. To find a replacement for the annotated blank word w with the tag r in a sentence, we draw a sample from S_r and use it as the inaccurate word. Obviously, some tags are more common than the others in natural language and as a result the incorrect words are similarly the same.

To draw a sample from a set, we use the distribution of the words in all sentences. As a result, the

words with more occurrences in the training set have more chance to be appeared as an inaccurate word. Therefore, the rate of incorrect to correct appearances of different words are close to each other. With this approach, we prevent the rare words to be chosen as the inaccurate word frequently and vice versa.

5.2.2 Results

5.2.2.1 Detection Experiments:

In this subsection, we present our results for detection module and examine our method with various settings. The results are summarized in Table 5.1. Following we explain each experiment in more details.

Random guess is to select one of the words in the sentence randomly as the inaccurate word. In *Text Only Experiments* part of Table 5.1, we compare all the blind experiments, where no visual features are used to detect the inaccuracy. *Vanilla LSTM* uses a simple LSTM to directly produce the $\mathcal{D}_t$(Equation 5.7) out of its hidden state using a fully connected layer.

One-Way Long-Term Dependencies uses just u_l in Equation 5.6. *Long-Term Dependencies* experiment uses Recurrent Neural Networks method explained in Section 5.1.1.2. *Convolutional N-Grams w/o Position Embedding* uses just Convolutional N-Grams, however, without the contribution of the positions of each word explained in Section 5.1.1.1 while *Convolutional N-Grams* is the complete explained module in Section 5.1.1.1. These two experiments show the effectiveness of our proposed words position gating, and finally, *Convolutional N-Grams + Long-Term Dependencies* uses the combination of Convolutiona N-Grams and RNNs as mentioned in Section 5.1.1.3. The last experiment reveals the contribution of both short-term and long-term dependencies of words in a sentence for the TC task.

To further study the strength of our method to detect the wrong words, we compare our method with a *Commercial Web-App*[1]. This application can detect structural or grammatical errors in text. We provide 600 random samples from the test set to the web application and examine if it can detect the inaccuracy. In Table 5.1, we show the comparison between our method and the aforementioned web application. This experiment shows the superiority of our results and also the quality of our generated dataset.

In *Video and Text Experiments* part of the Table 5.1, we show experiments with both video and text. **Visual Gated Bias** experiment shows the capability of our proposed formulation to leverage the visual features in the detection sub-task. To show the superiority of our visual gating method, we conduct *Visual Feature Concatenation* experiment. In this experiment, we combine the visual feature vector $\Omega(\mathcal{V})$ with each of the vectors x_t and $\hat{x}_t$ in Equation 5.7 using concatenation and a fully connected layer. For these experiments, we have used the pre-trained C3D [146] to compute the $\Omega(\mathcal{V})$.

5.2.3 Correction Experiments

In Table 5.2, we provide our results for the correction task. Note that, the correction task is composed of both inaccurate word detection and correct word predictions sub-tasks; thus, a correct answer for a given test sample must have the exact position of the inaccurate word and also the true word prediction ($(t^*,w^*_{t^*})$ in Equation 5.1).

Our Model - Just Text experiment demonstrates our method performance with only textual information. *Our Model With C3D Features* uses both video and text, with C3D [146] features as visual features. Similarly, *Our Model With VGG19 Features* shows the results when VGG19 [150]

[1] www.grammarly.com

features are the visual input. In *Our Pre-trained Detection Model + Pre-Trained FIB* [90] experiment we use our best detection model from Table 5.1 to detect an inaccurate word. We remove the inaccurate word and make an incomplete sentence with one blank. Then, we use one of the pre-trained state of the art FIB methods [90], which uses two staged Bi-LSTMs (LR/RL LSTMs) for text encoding + C3D and VGG19 features + temporal and spatial attentions, to find the missing word of the incomplete sentence. We show the superiority of our method which has been trained End-to-End. In both of detection (Table 5.2) and correction (Table 5.1) tasks, there are accuracy improvements after including visual features. We also report the Mean-Average-Precision (MAP) metric, to have a comprehensive comparison. To measure the MAP, we compute $N \times |\beta|$ scores for all the possible $(t^*, w^*_{t^*})$.

Table 5.1: Detection Experiments Results. For these experiments we just evaluate the ability of different models to localize the inaccurate word.

Method	**Accuracy (%)**
Random	8.3
Text Only Experiments	
Commercial Web-App	18.8
Vanilla LSTM (One LSTM w/o Prop. Detection Formula)	28.0
One-Way Long-Term Dependencies (One LSTM)	58.0
Long-Term Dependencies (BiLSTM)	67.2
Conv N-Grams w/o Position Embedding	66.8
Conv N-Grams	69.0
Conv N-Grams + Long-Term Dependencies	*72.5*
Video and Text Experiments	
Conv N-Grams + Long-Term Dependencies + Visual Feature Concatenation	72.8
Conv N-Grams + Long-Term Dependencies + Visual Gated Bias	**74.5**

Table 5.2: Text Correction Experiments Results. For the correction task, a model needs to successfully locate the inaccurate word and provides the correct substitution.

Method	**Accuracy (%)**	**MAP (%)**
Random	0.04	$\simeq$0
Vanilla LSTM - Just Text	17.2	17.7
Our Model - Just Text	35.2	36.9
Our Pre-trained Detection Model + Pre-Trained FIB [90]	36.0	38.6
Our Model With C3D Features	38.6	39.8
Our Model With VGG19 Features	38.8	40.1
Our Model With VGG19 + C3D Features	**38.9**	**40.7**

5.2.4 Multiple Inaccuracies

Here, we show that our method is capable of to be generalized to sentences with more than one inaccurate words. We conduct a new experiment with multiple inaccuracies in the test sentences and show the results in Table 5.3. In fact, we replace all the annotated blank words in the LSMDC-FIB test sentences with an inaccurate word. We assume that the number of inaccuracies, k, is given for each test sample, but the model needs to locate them. To select the inaccuracies in each sentence, we use the LSMDC-FIB dataset annotations. Note that in training we use sentences that contain just one inaccurate word, similar to previous experiments. During the test time, we modify the Equation 5.11 to $t^*_{i=1,..,k} = arg\,kmax(\mathcal{D}_t)$, where $arg\,kmax$ returns the top k inaccurate word candidates. Number of inaccurate words in our test set sentences reaches up to 10 words. However, in Table 5.3, we show the detection results for sentences with each $k \leq 4$ value separately, and also the overall accuracy for all the k values.

Table 5.3: Detection and Correction results for sentences with multiple inaccuracies. Two types of Accuracy evaluations are provided. (1) Word-Based (WB) Accuracy: All correctly fixed incorrect words are counted independently. (2) Sentence-Based (SB) Accuracy: All inaccurate words in a sentence must be fixed correctly. Similarly, two types of MAP is reported: (1) WB-MAP, in which, one AP per each incorrect word is computed. (2) SB-MAP, in which, one AP per each sentence, including all the k incorrect words, is computed. k represents the number of inaccuracies in each sentence.

k =	1	2	3	4	All	1	2	3	4	All
# Of Test Samples	1805	4856	5961	520	30349	1805	2428	1987	130	9575
Detection	WB-Acc. (%)					SB-Acc. (%)				
Vanilla LSTM - Just Text	59	63	67	68	66	59	37	27	18	36
Our Method - Just Text	80	81	80	80	80	80	65	48	37	59
Our Method - Text + Video	**85**	**83**	**83**	**82**	**83**	**85**	**68**	**54**	**39**	**63**
Correction	WB-Acc. (%)					SB-Acc. (%)				
Our Method - Just Text	19	12	12	11	3	19	2	$\simeq 0$	$\simeq 0$	5
Our Method - Text + Video	**24**	**18**	**17**	**17**	**18**	**24**	**4**	$\simeq 0$	$\simeq 0$	**7**
Correction	WB-MAP (%)					SB-MAP (%)				
Our Method - Just Text	30	14	10	**8**	12	30	15	11	9	17
Our Method - Text + Video	**35**	**17**	**11**	7	**14**	**35**	**18**	**12**	**10**	**19**

5.2.5 Qualitative Results

We show a few VTC examples in Figure 5.2. For each sample, we show frames of a video and corresponding sentence with an inaccuracy. We provide the qualitative results for each example using our "Just Text" and "Text + Video" methods. We show two columns for the detection and correct word prediction. The green and red colors respectively indicate true and false outputs. Note that, for the VTC task, just a good detection or prediction is not enough. Both of these sub-tasks are needed to solve the VTC problem. For example, the left bottom example in Figure 5.2 shows a failure case for both "Just Text", and "Text + Video", although the predicted word is correct using "Text + Video".

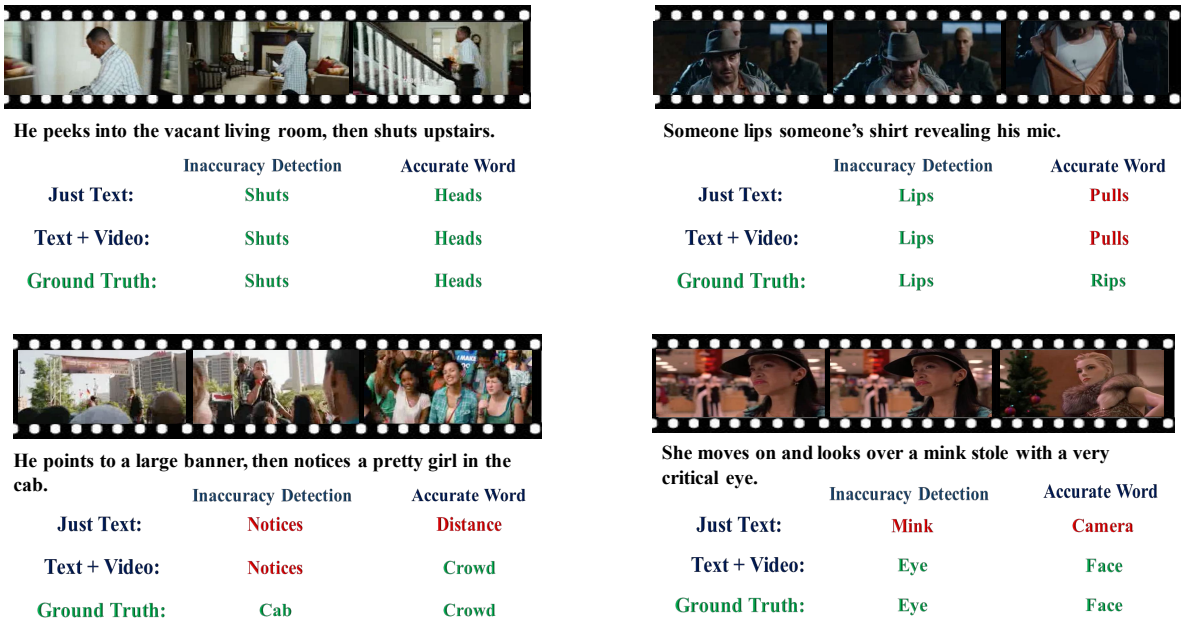


Figure 5.2: Here we show four samples of our test results. For each sample, we show a video and an inaccurate sentence, the detected inaccurate word, and the predicted accurate word for substitution. The green color indicates a correct result while the red color shows a wrong result.

5.3 Summary

We have presented a new formulation of text correction problem, where the goal is to find an inaccuracy in a video description, and fix it by replacing the inaccurate word. We propose a novel approach to leverage both textual and visual features to detect and fix the inaccurate sentences, and we show the superior results are obtained our approach. Moreover, we introduce an approach to generate a suitable dataset for VTC problem. Our proposed method provides a strong baseline for inaccuracy detection and correction tasks for sentences with one or multiple inaccuracies. We believe that our work is a step forward in the research related to intersection of Natural Language Processing and Computer Vision.

CHAPTER 6: VIDEO GENERATION IN THE WILD FROM TEXT EMPLOYING LATENT PATH CONSTRUCTION FOR TEMPORAL MODELING

The work in this Chapter is currently under review for ECCV 2020:

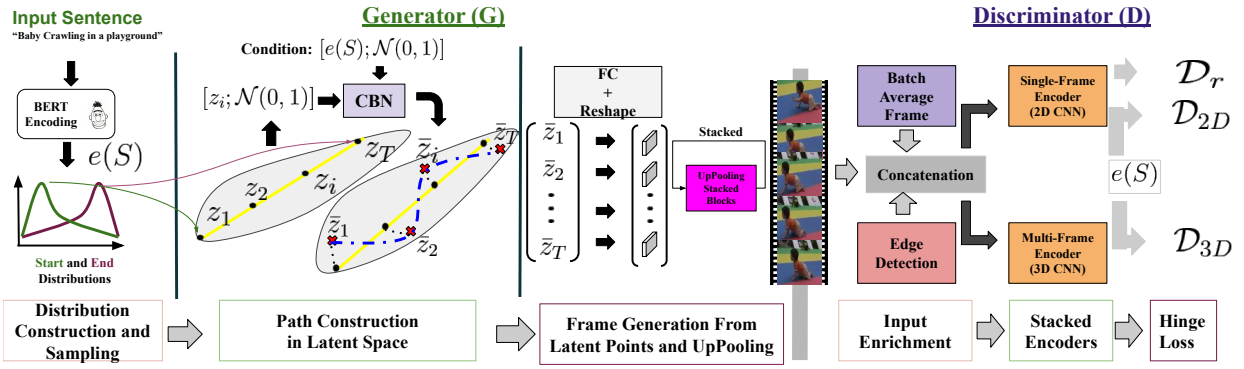


Figure 6.1: In this figure, we show a block-diagram of different steps of our proposed method. We encode the sentence using pre-trained BERT model and some trainable layers, and represent it by $e(S)$ (see Section 6.1.1 for details). Given $e(S)$, we construct two distributions and draw one sample from each corresponding to latent representations of start (z_1) and end (z_T) frames, respectively. We then determine T latent representations, $[\bar{z_1}, \bar{z_2}, \ldots, \bar{z_T}]$, corresponding to T frames, employing a context-aware interpolation in the latent space. We use Conditional Batch-Normalization (CBN, Section 6.1.3) with $e(S)$ and noise as the condition. Subsequently, we transform each $\bar{z}_i$ into a spatial representation using an FC and reshape layers, and increase its size to the desired resolution through stacked "UpPooling" blocks (Section 6.1.2.1). Inputs to the Discriminator are encoded sentence $e(S)$, and the video (real or fake). We augment the video input to the Discriminator by concatenating it with an average frame from the whole batch and edge maps of the frames. The discriminator employs a single and multi-frame based videos encoders along with $e(S)$, to measure if each frame and the video ($\mathcal{D}_{2D}$ and $\mathcal{D}_{3D}$) are relevant to the input sentence and if each spatial regions of each frame are naturally looking ($\mathcal{D}_r$). Finally, we train the proposed network with GAN Hinge-loss (Equations 6.6 and 6.5).

In this chapter, we address the task of video content generation using natural language sentences (as text). Our proposed method to solve the text to video generation follows the Generative-Adversarial framework, which includes a generator and a discriminator sub-modules. In Figure 6.1, we show our method diagram, including all the steps in both Generator(G) and Discrimi-

nator(D). In the rest of this Chapter, we present the details of our proposed model.

6.1 Methodology

In this section, we describe our proposed method in more details. Our method consists of Text Encoder (Section 6.1.1), Generator (Section 6.1.2.1) and Discriminator (Section 6.1.4) sub-modules.

6.1.1 Text Encoder

Given a sentence as the sequence of words $S = [w_1, w_2, \ldots, w_N]$, the purpose of the text encoder is to represent S as a vector of real numbers. Ideally, one can train a neural network from scratch or end-to-end with the rest of the system, similar to [113]. However, in this work, we target realistic datasets, i.e., A2D and UCF101, and due to the complex nature of such target datasets and annotations, we do not have enough number of examples for each of the words in the dataset. A large portion of the words in our target datasets are rare words. In addition, more than 500 verbs, adjectives, and nouns in the A2D dataset appear only once. Also, this amount of rare words makes models like [116] impractical. Therefore, in this research, We employ the BERT (Bidirectional Encoder Representations from Transformers) [154] sentence encoder, pre-trained on English Wikipedia[1]. BERT provides us a rich representation of all the sentences even if they contain rare words. We transform the 1024 dimensional output of BERT encoding into 256D using two blocks of Fully-Connected, Batch-normalization, and Leaky-ReLU layers. We denote the encoded sentence by $e(S) \in \mathcal{R}^{256}$.

[1] https://github.com/hanxiao/bert-as-service

6.1.2 Video Generator

Let $e(S)$ be the encoded sentence. We estimate two Gaussian distributions $\mathcal{N}_s(\mu_s, \sigma_s) \in \mathcal{R}^{256}$, and $\mathcal{N}_e(\mu_e, \sigma_e) \in \mathcal{R}^{256}$ for the starting and ending frames:

$$\mu_s, \mu_e, \sigma_s, \sigma_e = \mathcal{F}([e(S); \mathcal{N}(0, 1)]). \tag{6.1}$$

Here, $\mathcal{F}$ is a Multilayer perceptron (MLP). Concretely, we split the output of the $\mathcal{F}$ into four equally length vectors, and we use sigmoid non-linearity on top σ_s and σ_e. Note that, $[;]$ denotes concatenation operation throughout this chapter. We draw one vector from each of the distributions $\mathcal{N}(\mu_s, \sigma_s)$ and $\mathcal{N}(\mu_e, \sigma_e)$, and denote them by z_1 and z_T. To generate a video with T frames, we employ an interpolation to extract the latent representation for frame i:

$$z_i = \frac{T - i}{T} z_1 + \frac{i}{T} z_T. \tag{6.2}$$

Note that, there exist many interpolation methods like bi-linear, bi-cubic, and etc. We choose the linear interpolation for this step as the most simple option. We concatenate each of z_i vectors with a normal noise vector $\mathcal{N}(0, 1) \in \mathcal{R}^{32}$ and pass it through a Conditional Batch-Normalization (CBN) [155] (see Section 6.1.3), where the condition is $[e(S); \mathcal{N}(0, 1)]$. We denote normalized latent representations by $\bar{z}_i$. We briefly explain the CBN module and its effect on the training in Section 6.1.3. The added random noise $\mathcal{N}(0, 1)$ brings in the needed variability to the final motion. In addition, the CBN provides a stochastic context-aware transformation on each latent representation, to finally produce $\bar{z}_1 \dots \bar{z}_T$.

6.1.2.1 Frame Generator

In the second part of our Generator network, we propose a CNN based network to transform each of z_i latent vectors into a RGB frame. First, we transform the latent vectors into a spatial representation using a linear transformation and reshape. Basically, we map each $\bar{z}_i$ into a $h_1 \times w_1 \times c_1$ vector using a Fully-Connected layers, and reshape it into a spatial tensor $\in \mathcal{R}^{h_1 \times w_1 \times c_1}$. In our experiments, we choose $h_1 = 4$, $w_1 = 4$, and $c_1 = 2048$.

To build the frames of desired resolution, we employ a CNN based module to increase (up-pooling) the resolution of spatial features (see Figure 6.2). The proposed module increases the resolution of the given input via two paths, a short path with only one convolution layer, and a longer path with two convolution layers with Conditional Batch-Normalization (Section 6.1.3) and ReLU activation in between . The short path plays a role of skip-connection that facilitates the training process. However, the longer path increases the capacity of the model by adding non-linearity and normalization. We use Nearest-Neighbour (NN) interpolation to increase the spatial size of each tensor. We tried PixelShuffle [156], and 2D-Deconvolutions as other design choices, however, NN-interpolation consistently produced better results in all experiments.

We stack the "UpPooling" block (as explained in Figure 6.2) to reach the desired output resolution. In our experiments, our generated frames are 64×64; thus, we need four blocks of UpPooling Blocks. Finally, we apply a 3D convolution on the output of the final layer with 3 (RGB) filters and $tanh$ non-linearity to build the final RGB frame.

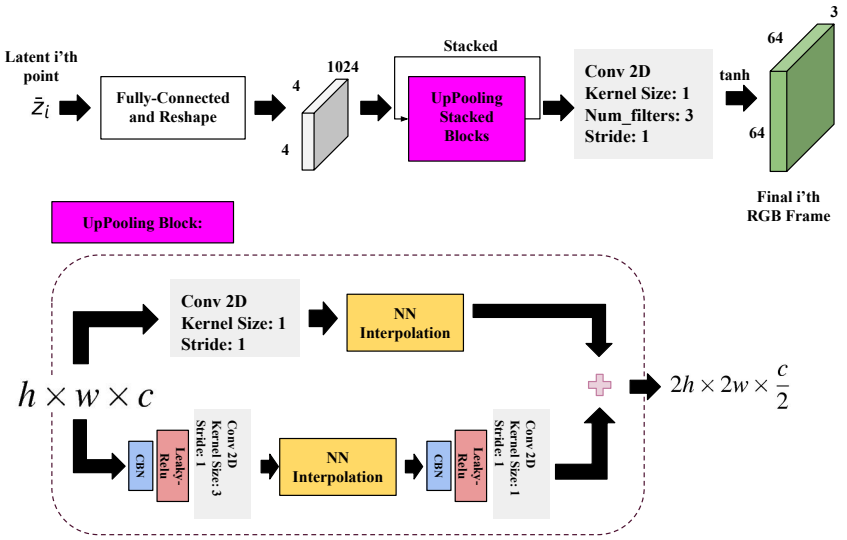


Figure 6.2: Here we present the frame generator sub-module. We transform each latent point, $\bar{z}_i$, which corresponds to i'th frame of the video, into a spatial representation using a Fully-Connected layer and reshape. We increase the resolution of the spatial representation by a factor of two in each "UpPooling Block". The UpPooling block has a short and a long path. The short path has only a linear 1×1 Convolution and the long path, which has two 2D CNNs from Conditional Batch Normalization (CBN) and Leaky-ReLU. We increase the resolution in both paths by a Nearest Neighbour(NN) interpolation. Note that, the concatenation of the encoded text and a 32 dimensional noise ($[e(S); \mathcal{N}(0, 1)]$) are used as a condition to all CBNs. Finally, we build the RGB frame by a 1×1 Convolution and tanh non-linearity.

6.1.3 Conditional Batch-Normalization (CBN)

Here, we briefly explain the conditional Batch-normalization we employ in our generator. Given an input x and condition c we compute $\bar{x}$ as follows:

$$\bar{x} = \gamma(c)_{\mu=1} \frac{x - \mu_x}{\sigma_x} + \beta(c)_{\mu=0}, \tag{6.3}$$

where $\gamma(.)$ and $\beta(.)$ are neural networks that have the same output shape as shape of x. In our case, we use a single linear FC layer to implement each of them. Also, $\gamma(c)_{\mu=b} = \gamma(c) - \mu_{\gamma(c)} + b$ (same for $\beta(c)$, that simply means to shift the mean value of the batch, $\mu_{\gamma(c)}$, to b), and we compute the mean μ, and variance σ over the batch. Conditional Batch-Normalization, in fact, normalizes the

mean and variance of each sample with respect to the statistical data of whole batch, and applies a context-aware affine transformation (scale $\gamma(c)$, and shift $\beta(c)$), where the context is represented as condition c, on the normalized input.

6.1.4 Discriminator

Our proposed Discriminator (D) consists of a frame-based and a video-based encoder sub-modules. The frame-based sub-module encodes each frame globally and locally using 2D CNN blocks. It encodes each frame into one vector (global frame encoding), and estimates its relevance to the input text, while it uses spatial features, extracted before the global average pooling (see Figure 6.3), to compute one score for each region of the frame. It helps the discriminator not only to determine if the global context of the video is related to the text, but also each spatial region of the frames is locally natural looking. Similarly, the multi-frame (video-based) sub-module D, leverages 3D CNN blocks to encode all frames of a video, as a sequence, into a vector. To compute the relevance between the encoded video $v \in \mathcal{R}^{d_v}$, which can be a single or multi-frame based encoded vector, and the encoded sentence $e(S) \in \mathcal{R}^{d_e}$, we compute the discriminator score by:

$$\mathcal{D}(v, e(s)) = W_D \times (\sigma(W_e \times e(S)) \odot v), \tag{6.4}$$

where $\odot$ represents element-wise multiplication, $W_D \in \mathcal{R}^{d_v \times 1}$ and $W_e \in \mathcal{R}^{d_e \times d_v}$. We denote the discriminator scores from 3D CNN multi-frame video-based encoder by $\mathcal{D}_{3D}$, and from 2D CNN frame-based encoder by $\mathcal{D}_{2D}$. Also, we use $\mathcal{D}_r$ for the spatial regions' scores which is computed along with the single-frame encoder. Note that, $\mathcal{D}_r$ is independent of $e(S)$. Finally, we take an average of all the scores from all the frames to compute the final $\mathcal{D}_{2D}$ and $\mathcal{D}_r$.

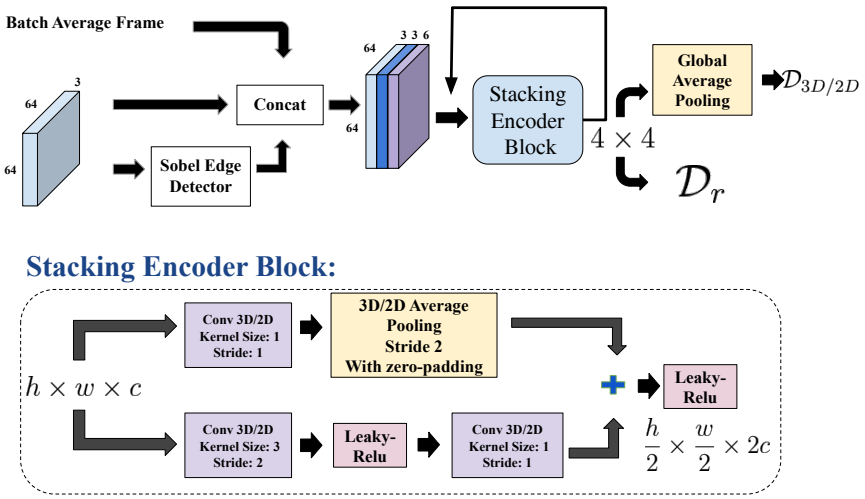


Figure 6.3: Our proposed Discriminator building block. First, we compute the average frame of a batch and each frames' edge map using the Sobel edge detector and concatenate them to the input frames. To reduce the input resolution, we implement a stacking encoder block that consists of a short path of a 1×1 convolution followed by average pooling, and in parallel, a long path with a 3×3 convolution followed by average pooling and a 1×1 Convolution. We sum the outputs of short and long paths, resulting in half of the resolution of the input. We stack several blocks like this until we reach a 4×4 spatial resolution. Note that, for the multi/single-frame based discriminator, we use 3D/2D convolutions and average pooling layers.

6.1.4.1 Discriminator Input Enrichment

We observe that using Batch-Normalization or Conditional Batch-Normalization in Discriminator (D) architecture do not facilitate the training process. In our experiments, D containing BN dominates the G in early iterations, and results in severe mode-collapse. To utilize the stochastic batch information, we propose to concatenate each RGB frame with an average RGB frame of all the frames in a batch. In this scenario, D benefits from information in both single sample and batch statistics. This technique reduces the mode-collapse in our experiments; without this technique, we observe that there is a high chance that the model collapses into one or two modes during training, and we need to reload an earlier checkpoint to continue training. Additionally, as shown in previous studies [157], edge information is useful; therefore, we augment each RGB frame with

its Sobel edge map. See Figure 6.3 for more details.

6.1.5 Loss Function

We use the hinge-based loss to train our GAN architecture. We compose the Generator loss as:

$$\mathcal{L}_G = -\mathcal{D}_{3D}(G(S), e(S)) - \mathcal{D}_{2D}(G(S), e(S)) - \mathcal{D}_r(G(S)), \tag{6.5}$$

and the discriminator loss as:

$$\begin{aligned}\mathcal{L}_D = [1 - \mathcal{D}_{3D}(\mathcal{V}, e(s))]_+ + [\mathcal{D}_{3D}(G(S), e(S)) + 1]_+ \\ [1 - \mathcal{D}_{2D}(\mathcal{V}, e(S))]_+ + [\mathcal{D}_{2D}(G(S), e(S)) + 1]_+ \\ [1 - \mathcal{D}_r(\mathcal{V})]_+ + [\mathcal{D}_r(G(S)) + 1]_+,\end{aligned} \tag{6.6}$$

where $[x]_+ = max(0, x)$ and $\mathcal{V}$ is a real video from training set with the text annotation S, and $G(S)$ is a generated video given S.

6.2 Experimental Results

In this section, we describe the experimental settings, including the datasets, the metrics we use to quantify the performance of our approach, a brief description of the baselines, quantitative and qualitative results. Finally, we provide more details about the hardware and implementations.

6.2.1 Dataset

Actor and Action (A2D) The A2D dataset is a popular dataset for the actor and action video segmentation task. The authors in [120] provide sentence annotations corresponding to video actor segmentation, and authors in [158] provide frame level bounding box for each actor. A2D contains 3,782 videos with 6,656 sentences corresponding to actor segmentation, including 811 different nouns, 225 verbs, and 189 adjectives. Each sentence corresponds to one actor throughout the whole video. Therefore, there can be more than one sentence annotated for each video, corresponding to multiple actors. We crop the video for each sentence by constructing a maximal bounding box that covers all the instances of the object in all the frames. This way we get one video sequence for each sentence; hence, we have 6,656 cropped video sequences with one sentence for each.

UCF101 is one of the popular datasets for the task of human action recognition. However, to the best of our knowledge, there has been no video level captioning annotations for UCF101. We have annotated 9 classes of UCF101. The selected classes are: "Fencing", "Basketball", "Basketball Dunk", "Biking", "Golf Swing", "Gymnastics", "Cricket Bowling", and "Cliff Diving". We asked the annotators to describe each video by a short sentence. Note that some of videos in each of UCF101 classes are very similar, and we let the annotators use identical annotations based on their judgment. The corpus of video captions have 182 unique words, and the maximum sentence length is 22 words.

Robot Object Manipulation: Authors in [159] provide an object manipulation robotic dataset containing videos and corresponding user-to-robot textual command. This dataset contains "push" and "pick-up" tasks for multiple objects. Sentences are in form of "task + object description". For example, "pick-up the blue box". Each video is about 20 seconds, and we randomly pick 16 frames to train the system.

6.2.2 Evaluation Metrics

Inception Score (IS) [160]is widely used in quality assessment of generative models. Inception Score is computed based on a pre-trained classifier on the dataset. Ultimately, any generated sample must belong to a specific class (high probability output on a single activation of the classifier), and the model must generate outputs from all the available categories (diversity on the classifier output). A higher IS is better. To compute the Inception Score, we fine-tune the I3D model [161] pre-trained on Kinetics [162] and imagenet [61] on each of datasets with the same number of classes (in our case, 43,9 and 11 classes for A2D, UCF101, and Robotic datasets, respectively), and other settings like the frame size, frame rate (fps), etc. We fine-tune the pre-trained I3D model [161] on Kinetics [162] and imagenet [61] on each of datasets.

Fréchet Inception Distance (FID) [163] compares the statistics of two sets of samples, namely real and fake. We use the same fine-tuned I3D classifier used for the IS score and extract 1024 dimensional features. The lower FID is better. This quantitative measurement for video synthesis is also known as FVD [111].

R-Precision: Following [116], we employ R-Precision, which is a retrieval evaluation metric. To compute the R-Precision, we first train a CNN based retrieval network (again based on pre-trained I3D), that can rank a set of sentences, given a video. The retrieval network architecture consists of a video encoder and a text encoder, and we train it with a hinge ranking loss and cosine similarity until it fully converges on the training data. This network achieves "top-1 accuracy" of 80% and 60% for UCF101 and A2D training data, respectively. Later, given a sentence, we generate a video and using the retrieval network, we rank a set of 100 sentences, including unseen and seen. Assuming that there are R related sentences in the 100 sentences, and r of them are in top R ranked sentences, the R-precision score is: $\frac{r}{R}$. Note that, in contrast to [116], our datasets do not have multiple sentences per video sequence. Or simply, $R = 1$. To overcome this issue, we slightly alter each sentence by randomly dropping/replacing some words. Using this technique, we generate

between 6 to 12 related sentences for each video. We believe that if a sentence is slightly changed, it must be still ranked above totally unrelated sentences. We use this metric for A2D and UCF101 datasets.

Accuracy: Since there are only 11 unique sentence in the Robotic dataset [159], and some of them have only one word difference, the R-Precision is not a good option for evaluation. Instead, we train a classification network that given a video, classifies which of the 11 classes (unique sentence) the video belongs to. Later, we use this classification network and test it on the generated videos. A higher accuracy is better.

Table 6.1: A2D experimental results. All-FID: The FID on all the videos from all classes. Intra-FID: Mean of FID within videos of each class. R-P: R-Precision. IS: Inception Score.

	IS ↑	All-FID ↓	Intra-FID ↓	R-P↑
Temporal Modeling Baselines				
Deconvolution	3.84 ± 0.12	31.56	108.04	0.31
SLERP + LSTM	4.04 ± 0.17	**25.74**	104.13	0.34
ConvGRU	3.97 ± 0.33	34.78	109.78	0.30
Ablation Study on Textual Encoding				
Only Class Labels	3.90 ± 0.12	49.35	119.37	N/A
BiLSTM Sentence Encoder	3.09 ± 0.13	54.12	131.58	0.05
Ablation Study on Discriminator				
Frame Based Discriminator ($\mathcal{D}_{2D}$)	3.52 ± 0.11	169.59	49.17	0.06
Video Based Discriminator ($\mathcal{D}_{3D}$)	3.17 ± 0.09	39.75	117.92	0.08
Region-Based Discriminator ($\mathcal{D}_r$)	3.77 ± 0.20	38.14	100.18	0.05
Ours - Full Model	$\mathbf{4.85 \pm 0.16}$	25.91	**94.26**	**0.39**
Real Data	*9.93 ± 1.18*	*11.55*	*71.56*	*0.45*

6.2.3 Quantitative Results

We evaluate our trained model on UCF101 and A2D dataset using the explained metrics in Section 6.2.2. For a comprehensive study, we include some baselines in which we use other design choices from previous works. We provide the following baselines for all A2D, UCF101, and Robotics datasets. **Only Class Labels:** We train the model merely with video class labels. By

comparing the results of this method with our final method, we show that sentences are more compelling conditions for the task of video generations; and our generative model benefits from additional information contained in a sentence compared to employing only labels.

SLERP + LSTM: We follow the design of [102] which construct the temporal domain of a video by a Spherical Linear Interpolation (SLERP), and estimates each latent point representation $\bar{z}_i$ using an LSTM.

Deconvolution: In this baseline, similar to [113], we expand the number of generated frames by Stacking Deconvolution layers (also known as Convolution Transpose).

Conv-RNN: Similar to [111], we estimate a distribution out of the input text, and transform it into a spatial representation using a linear transformation and reshape. The resulting spatial representation is repeated T times and is passed to a Convolutional Recurrent Neural Network. We observe that a Convolutional Gated Recurrent Unit (ConvGRU) with layer Normalization is the best choice for this baseline.

Real Data: Evaluation on the "Real Data" gives us a better understanding of what would be a realistic expected value for each of the Inception Score, Fréchet Inception Distance, and R-Precision. We do not expect even on real data, we will get the best possible scores, since neither of I3D or our retrieval network is perfect. Note that, the FID value would be ideally zero on the real data itself; however, we split the set of all the real videos in half and compute the FID between these two sets.

Furthermore, for the A2D dataset, which is the most challenging dataset, we provide more ablation studies (Table 6.1) to show the contribution of the proposed components in our method. In **BiLSTM Sentence Encoder** experiment, we replace the pre-trained BERT encoder with a simple BiLSTM that trains from the scratch, and the performance of the method drops drastically. This is due to the reasons mentioned in Section 6.1.1. Moreover, we provide ablation study on the Discriminator. We isolate each of the discriminator terms, namely $\mathcal{D}_r$, $\mathcal{D}_{2D}$, and $\mathcal{D}_{3D}$. By comparing the performance of these ablation studies with the full model, we show that the terms in

Equations 6.5, and 6.6 are complementary.

For the sake of fairness, we keep the implementation of all the baselines and our proposed method as similar as possible. For example, the discriminator architecture, hardware, and etc. In Tables 6.1, 6.2, and 6.3 we show the results of our proposed method respectively on A2D, UCF101, and Robot-Object-Manipulation datasets. Our proposed method is competitive to the baselines based on all the evaluation metrics.

Table 6.2: UCF 101 Quantitative Results. Here we report the Inception Score (IS), R-Precision (R-P), Fréchet Inception Distance (FID). For the FID score, all the videos from all the classes of the dataset, are used to compute the FID score. And in another experiment (Intra-Classes FID) we compute the FID score for the videos within each class.

All-Videos				Intra-Classes FID									
	IS ↑	R-P ↑	FID ↓	Fencing	Basketball	B. Dunk	Biking	Diving	Golf	Gymnastics	Crick. Bowl.	CliffDiving	*Mean*
Only Class Labels	3.69 ± 0.19	N/A	60.38	163.97	77.93	131.31	144.57	49.37	48.52	121.26	77.89	39.86	94.96
SLERP + LSTM	1.02 ± 0.00	0.04	127.05	181.59	132.14	132.03	190.87	222.27	101.90	134.91	174.22	94.22	151.57
Deconvolution	3.95 ± 0.19	0.19	51.64	126.85	116.87	53.06	98.28	85.80	59.81	105.05	103.95	49.91	88.84
ConvGRU	5.93 ± 0.18	0.35	30.24	63.31	54.99	70.03	66.61	68.52	23.01	90.65	40.53	35.89	57.06
Ours	$\mathbf{7.01 \pm 0.36}$	**0.43**	**17.12**	**29.20**	**28.08**	**54.69**	**46.48**	**48.54**	**19.44**	**46.24**	**31.40**	**35.44**	**37.72**
Real Data	*8.24 ± 0.20*	*0.56*	*6.92*	*14.51*	*18.71*	*16.21*	*11.19*	*16.76*	*3.87*	*18.75*	*12.64*	*15.78*	*14.27*

Table 6.3: Robotic experimental results. All-FID: The FID on all the videos from all classes. Intra-FID: Mean of FID within videos of each class. IS: Inception Score.

	IS ↑	All-FID ↓	Intra-FID ↓	Accuracy (%) ↑
Only Class Labels	1.99 ± 0.19	20.39	73.2	10.4
Deconvolution	2.97 ± 0.21	6.59	18.49	70.4
SLERP + LSTM	$\mathbf{3.47 \pm 0.16}$	4.60	18.22	73.7
ConvGRU	3.17 ± 0.26	6.65	25.74	50.4
Ours	3.36 ± 0.15	**3.79**	**16.45**	**76.6**
Real Data	*3.64 ± 0.3*	*3.4*	*11.8*	*100*

6.2.3.1 *Qualitative Results*

In Figures 6.4 and 6.5 we provide qualitative results for A2D and UCF101 datasets. Each figure comes with multiple sentences and generated videos corresponding to each of them. In Figure 6.6, we show generated videos that contain 16 frames. Note that, for the Robot dataset, each video represents a full task performance, which usually has around 200 frames in the original dataset. These results show that our method can handle datasets with a higher skip frame rate (lower fps). In more realistic and wild datasets like A2D, videos can have various ranges of motion. A video can have minimal motion (static video) or jumpy consecutive frames. We observe that our model can successfully cover various motions. For example, in Figure 6.4, the top left example ("The bird is climbing the chair") has much less motion than the bottom left example ("A bird is flying").

6.2.4 *Experimental Setup Details*

For both of UCF101 and A2D dataset, we randomly select 5 to 9 frames, with skip rate of 1 frame ($\simeq$ 15 fps); meaning that the training clips can be from the beginning, middle or end of a video sequence. For the Robotic dataset, we sample 16 frames from a full length demonstration of the robot that can be up to 20 seconds. Thus, the videos shown in Figures 6.4, and 6.5 represent about 0.5 of an actual video, and the videos shown in Figure 6.6 covers a longer range (up to 20 seconds) of time. We train the models on different datasets in slightly different manners. We use 1 Titan X Pascal GPU to train the experiments of UCF101, and 4 GPUs to train the A2D dataset. Due to the higher variance of videos in A2D dataset, it takes more time for our model to start generating meaningful videos. The model takes 1 day to train on UCF101 and Robotic, and 3 days on A2D. We employ Spectral Normalization [164] on both of Generator and Discriminator modules in all training iterations. We train the Generator and Discriminator equally, i.e., training Generator and Discriminator alternatively, with one iteration for each. We use Adam optimizer with learning rate

0.0001 for both of G and D.

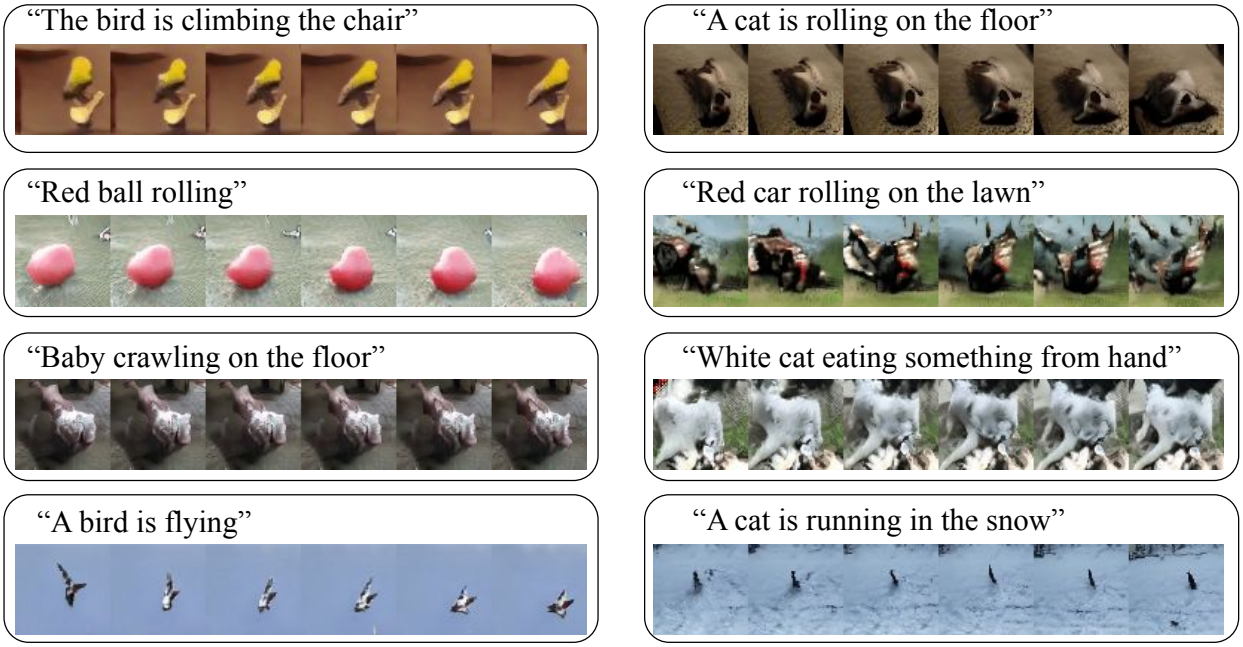


Figure 6.4: Qualitative Results on A2D dataset. Corresponding to each sentence we show the frames of generated videos. All samples are 6 frames with 64×64 resolution. Our proposed model can successfully produce diverse videos with different amount of motions, backgrounds and objects.

Figure 6.5: UCF101 qualitative results. Corresponding to each sentence we show the frames of generated videos. All samples are 6 frames with 64×64 resolution.

Figure 6.6: Robotic dataset qualitative results. Corresponding to each user command (sentence) we show the frames of generated videos. All samples are 16 frames with 64×64 resolution.

6.3 Summary

In this chapter, we tackle the problem of text to video generation on realistic datasets with free-form sentences. Our proposed method models the temporal dynamics of a video by constructing a path in the latent space. Each point on this path will transform into an RGB frame through a set of stacking "upPooling" layers. Our method provides superior results compared to well-known approaches like Deconvolution and ConvLSTM. To the best of our knowledge, we are the first to solve this problem on challenging datasets like A2D and UCF101. We believe that solving the video content generation using text has a lot of research value and has many real-world use cases.

CHAPTER 7: CONCLUSION AND FUTURE WORK

The research and methods outlined in the preceding sections have been motivated by the unprecedented surge of unanalyzed video and text data in our modern world. As humans, we are incapable of comprehensively processing and analyzing this wealth of data and in turn must harness the power of Artificial Intelligence to extract meaningful information.

7.1 Summary

In this dissertation, we have studied the joint comprehension of video and text data. We have proposed methods to solve real-world problems involving natural language and video data, such as Video Retrieval, Question Answering, Text Correction, and Video Creation. The power of the problem-specific solutions we have provided rely on the nascent field of Artificial Intelligence, and specifically on the sub-field of Machine Learning. With the introduction of each problem-solution pair, we have illustrated our competitive results in conjunction with ablation studies on the respective domains.

In Chapter 3, we studied the problem of Video Retrieval in large video datasets. We proposed a framework that can improve the traditional video retrieval models, which we refer to as video-level methods in Chapter 3, in which the user query contains more than one concept, like a sentence. We proposed a ranking based model, motivated by a ranking SVM, that captures both inter and intra-shot concept correlations. This model also utilized a 0-1 loss to ensure robustness against outlier samples in the training data. Finally, we illustrated the superior results of the proposed model compared to traditional video-level models.

In Chapter 4, we studied the Video Fill-In-the-Blank (VFIB) problem that is a novel form of Vi-

sual Question Answering (VQA). Compared to Video Retrieval, a VQA model can extract more detailed knowledge from a video by enabling a user to ask questions about a video or image. In our proposed approach, we leverage spatial and temporal attention to extract relevant information from the video to answer a given fill-in-the-blank question. We also presented a new text encoder, lr/rl LSTMs, that encodes the segmented FIB question better than traditional BiLSTMs and vanilla LSTMs. We use the Large-Scale-Movie-Description-Challenge (LSMDC) to train and test our method. We also extend the VFIB problem from only one blank to multiple blanks.

In Chapter 5, we follow the problem outlined in Chapter 4, as well as introduce a real-world application of VFIB, which is Visual Text Correction (VTC). In this chapter, we present the previously unformulated problem of detecting an inaccuracy in a video caption and suggesting correct words to replace the inaccurate word. To solve the VTC problem, we proposed to leverage LSTMs and 1D-CNNs to capture the long-term and short-term dependencies of a sentence, respectively. To handle the words without a specific visual pattern, like "love", we propose a novel method named "Visual Gated Bia". The proposed Visual Gated Bias approach enables the network to transform the visual features in a specific way for each word in a sentence through a gating process. We present the accuracy and Mean Average Precision (MAP) quantitative results for the inaccuracy detection and correct word prediction tasks.

In Chapter 6, we tackled the challenging problem of text-conditioned video generation. Our method proposed the novel approach of modelling the temporal dynamics of a video with a straight line in the latent space. We transform each point in the latent space into an RGB frame; hence, the line in the latent representation transforms into a video. We show the superior results on realistic datasets like Actor-Action Dataset (A2D), UCF101, and Robotics.

7.2 Future Work

Our work in this field has led us to believe that solutions to the joint video-text understanding problem have far-reaching, real-world applications. In light of this conclusion, we hope our work herein will encourage other researchers to continue to define and study new problems in this area. While our work has advanced numerous problems in the field, there are still many great strides to conquering the problem in its entirety. In addition, we believe that our work has laid the foundation for many extensions which will continue the overall goal of a comprehensive solution. In the following, we outline several specific areas in which we feel immediate contributions may precipitate.

We postulate that retrieval models can be further improved, specifically for rare concepts that do not have many training examples. In our research, we observed that our methods suffered from word-concepts with few positive training examples. We believe that deep learning has a lot to offer to improve the retrieval problem results, including rare concepts. We suggest Active Learning for the re-sampling and annotation process to create a balanced dataset. Generative models have the potential to generate synthetic data to compensate for rare concept samples. We also suggest studying new forms of loss functions and regularization techniques for the ranking problem that takes concepts with limited positive samples into account.

As mentioned above, we have addressed the Video Fill In the Blank (VFIB) and Visual Text Correction (VTC) problems. These problems provide tools to learn better features for text and video and to ultimately improve other tasks in Computer Vision and Natural Language Processing. VFIB and VTC are great options for self-supervised learning. One can remove or replace words of sentences and learn rich representations by solving VTC and VFIB tasks, similar to [154]. Furthermore, we address the VTC problem by constructing a synthetic dataset. It is our belief that a valuable next step in future works will be to find a way to collect human-made video description errors (inac-

curacies) effectively, as well as to train and examine machine learning models on more realistic data.

At present, a solution to the Video Generation problem is highly coveted in the Computer Vision community. For future work, we suggest collecting larger video datasets that include captions and atomic actions, actors, and objects. These larger datasets will allow future researchers to better evaluate their methods on real-world data. Furthermore, our observations have led us to the opinion that the process of training video generation systems is an unstable process. With said instability, models can easily face mode-collapse. Thus, it is our belief that further research into methods and techniques which provide stability to the training process will be of great significance.

LIST OF REFERENCES

[1] A. Rohrbach, M. Rohrbach, N. Tandon, and B. Schiele, "A dataset for movie description," in *CVPR*, 2015.

[2] C. V. N. Index, "Global mobile data traffic forecast," *http://www. cisco. com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862. html*, 2014.

[3] ——, "The zettabyte era–trends and analysis," *Cisco white paper*, 2013.

[4] V. Cisco, "The zettabyte era: trends and analysis. updated (07/06/2017)," 2017.

[5] Y. A. Aslandogan and C. T. Yu, "Techniques and systems for image and video retrieval," *Knowledge and Data Engineering*, vol. 11, no. 1, pp. 56–63, 1999.

[6] W. Hu, N. Xie, L. Li, X. Zeng, and S. Maybank, "A survey on visual content-based video indexing and retrieval," *Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 41, no. 6, pp. 797–819, 2011.

[7] C. G. Snoek and M. Worring, "Concept-based video retrieval," *Foundations and Trends in Information Retrieval*, vol. 2, no. 4, pp. 215–322, 2008.

[8] A. Hauptmann, R. Yan, and W.-H. Lin, "How many high-level concepts will fill the semantic gap in news video retrieval?" in *ACM international conference on Image and video retrieval*, 2007.

[9] S.-F. Chang, D. Ellis, W. Jiang, K. Lee, A. Yanagawa, A. C. Loui, and J. Luo, "Large-scale multimodal semantic concept detection for consumer video," in *Workshop on multimedia information retrieval*. ACM, 2007.

[10] G.-J. Qi, X.-S. Hua, Y. Rui, J. Tang, T. Mei, and H.-J. Zhang, "Correlative multi-label video annotation," in *ACM Multimedia*, 2007.

[11] Y. Yang and M. Shah, "Complex events detection using data-driven concepts," in *ECCV*, 2012.

[12] C. Snoek, K. van de Sande, D. Fontijne, A. Habibian, M. Jain, S. Kordumova, Z. Li, M. Mazloom, S. Pintea, R. Tao *et al.*, "Mediamill at trecvid 2013: Searching concepts, objects, instances and events in video," in *NIST TRECVID Workshop*, 2013.

[13] A. Dehghan, M. M. Kalayeh, Y. Zhang, H. Idrees, Y. Tian, A. Mazaheri, M. Shah, J. Liu, and H. Cheng, "Ucf-crcv at trecvid 2014: Semantic indexing," in *NIST TRECVID Workshop*, 2014.

[14] G. Ye, Y. Li, H. Xu, D. Liu, and S.-F. Chang, "Eventnet: A large scale structured concept library for complex event detection in video," in *ACM Multimedia*, 2015.

[15] C. G. Snoek, B. Huurnink, L. Hollink, M. De Rijke, G. Schreiber, and M. Worring, "Adding semantics to detectors for video retrieval," *Multimedia*, vol. 9, no. 5, pp. 975–986, 2007.

[16] S.-Y. Neo, J. Zhao, M.-Y. Kan, and T.-S. Chua, "Video retrieval using high level features: Exploiting query matching and confidence-based weighting," in *Image and Video Retrieval*. Springer, 2006, pp. 143–152.

[17] D. Wang, X. Li, J. Li, and B. Zhang, "The importance of query-concept-mapping for automatic video retrieval," in *ACM Multimedia*, 2007.

[18] A. P. Natsev, A. Haubold, J. Tešić, L. Xie, and R. Yan, "Semantic concept-based query expansion and re-ranking for multimedia retrieval," in *ACM Multimedia*, 2007.

[19] A. Haubold and A. Natsev, "Web-based information content and its application to concept-based video retrieval," in *International conference on Content-based image and video retrieval*. ACM, 2008.

[20] X.-Y. Wei, C.-W. Ngo, and Y.-G. Jiang, "Selection of concept detectors for video search by ontology-enriched semantic spaces," *Multimedia*, vol. 10, no. 6, pp. 1085–1096, 2008.

[21] L. Kennedy, S.-F. Chang, and A. Natsev, "Query-adaptive fusion for multimodal search," *Proceedings of the IEEE*, vol. 96, no. 4, pp. 567–588, 2008.

[22] K. Mc Donald and A. F. Smeaton, "A comparison of score, rank and probability-based fusion methods for video shot retrieval," in *Image and video retrieval*. Springer, 2005, pp. 61–70.

[23] Y. Aytar, M. Shah, and J. Luo, "Utilizing semantic word similarity measures for video retrieval," in *CVPR*, 2008.

[24] T. Lan, W. Yang, Y. Wang, and G. Mori, "Image retrieval with structured object queries using latent ranking svm," in *ECCV*, 2012.

[25] D. Grangier and S. Bengio, "A discriminative kernel-based approach to rank images from text queries," *Pattern Analysis and Machine Intelligence*, vol. 30, no. 8, pp. 1371–1384, 2008.

[26] B. Siddiquie, R. S. Feris, and L. S. Davis, "Image ranking and retrieval based on multi-attribute queries," in *CVPR*, 2011.

[27] F. X. Yu, R. Ji, M.-H. Tsai, G. Ye, and S.-F. Chang, "Weak attributes for large-scale image retrieval," in *CVPR*, 2012.

[28] M. Malinowski and M. Fritz, "A multi-world approach to question answering about real-world scenes based on uncertain input," in *NIPS*, 2014.

[29] M. Ren, R. Kiros, and R. Zemel, "Exploring models and data for image question answering," in *NIPS*, 2015.

[30] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, 1997.

[31] A. F. Smeaton, "Techniques used and open challenges to the analysis, indexing and retrieval of digital video," *Information Systems*, vol. 32, no. 4, pp. 545–559, 2007.

[32] P. Over, G. Awad, M. Michel, J. Fiscus, W. Kraaij, A. F. Smeaton, G. Quéenot, and R. Ordelman, "Trecvid 2015 – an overview of the goals, tasks, data, evaluation mechanisms and metrics," in *NIST TRECVID Workshop*. NIST, USA, 2015.

[33] L. Jiang, S.-I. Yu, D. Meng, T. Mitamura, and A. G. Hauptmann, "Bridging the ultimate semantic gap: A semantic search engine for internet videos," in *ACM Multimedia*, 2015.

[34] S.-F. Chang, W. Hsu, L. Kennedy, L. Xie, A. Yanagawa, E. Zavesky, and D.-Q. Zhang, "Columbia university trecvid-2005 video search and high-level feature extraction," in *NIST TRECVID Workshop*, 2005.

[35] M. Campbell, A. Haubold, M. Liu, A. Natsev, J. R. Smith, J. Tesic, L. Xie, R. Yan, and J. Yang, "Ibm research trecvid-2007 video retrieval system." in *NIST TRECVID Workshop*, 2007.

[36] R. Yan and A. G. Hauptmann, "The combination limit in multimedia retrieval," in *ACM Multimedia*, 2003.

[37] X. Li, D. Wang, J. Li, and B. Zhang, "Video search in concept subspace: a text-like paradigm," in *ACM international conference on Image and video retrieval*, 2007.

[38] S. Assari, A. Zamir, and M. Shah, "Video classification using semantic concept co-occurrences," in *CVPR*, 2014.

[39] A. F. Smeaton, P. Over, and W. Kraaij, "High-level feature detection from video in trecvid: a 5-year retrospective of achievements," in *Multimedia content analysis*. Springer, 2009, pp. 1–24.

[40] G. Iyengar and H. J. Nock, "Discriminative model fusion for semantic concept detection and annotation in video," in *ACM Multimedia*, 2003.

[41] C. G. Snoek, M. Worring, J. C. Van Gemert, J.-M. Geusebroek, and A. W. Smeulders, "The challenge problem for automated detection of 101 semantic concepts in multimedia," in *ACM Multimedia*, 2006.

[42] T. Chen, D. Borth, T. Darrell, and S.-F. Chang, "Deepsentibank: Visual sentiment concept classification with deep convolutional neural networks," *arXiv preprint arXiv:1410.8586*, 2014.

[43] L.-J. Li, H. Su, L. Fei-Fei, and E. P. Xing, "Object bank: A high-level image representation for scene classification & semantic feature sparsification," in *Advances in neural information processing systems*, 2010, pp. 1378–1386.

[44] S. Sadanand and J. J. Corso, "Action bank: A high-level representation of activity in video," in *CVPR*, 2012.

[45] T. Althoff, H. O. Song, and T. Darrell, "Detection bank: an object detection based video representation for multimedia event recognition," in *ACM Multimedia*, 2012.

[46] L. Torresani, M. Szummer, and A. Fitzgibbon, "Efficient object category recognition using classemes," in *ECCV*, 2010.

[47] C. H. Lampert, H. Nickisch, and S. Harmeling, "Attribute-based classification for zero-shot visual object categorization," *Pattern Analysis and Machine Intelligence*, vol. 36, no. 3, pp. 453–465, 2014.

[48] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth, "Describing objects by their attributes," in *CVPR*, 2009.

[49] Q. Li, J. Wu, and Z. Tu, "Harvesting mid-level visual concepts from large-scale internet images," in *CVPR*, 2013.

[50] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, "Support vector machine learning for interdependent and structured output spaces," in *International conference on Machine learning*. ACM, 2004.

[51] C.-N. J. Yu and T. Joachims, "Learning structural svms with latent variables," in *International Conference on Machine Learning*. ACM, 2009.

[52] J. Wu and M. Worring, "Efficient genre-specific semantic video indexing," *IEEE Transactions on Multimedia*, vol. 14, no. 2, pp. 291–302, 2012.

[53] V. Vapnik, *The nature of statistical learning theory*. Springer Science & Business Media, 2013.

[54] R. Herbrich, T. Graepel, and K. Obermayer, "Large margin rank boundaries for ordinal regression," *Advances in neural information processing systems*, pp. 115–132, 1999.

[55] T. Joachims, "Optimizing search engines using clickthrough data," in *ACM SIGKDD*, 2002.

[56] A. Shashua and A. Levin, "Ranking with large margin principle: Two approaches," in *Advances in neural information processing systems*, 2002, pp. 937–944.

[57] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

[58] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[59] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015.

[60] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[61] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.

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

[63] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, 1997.

[64] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[65] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *ICML*, 2015.

[66] S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. Mooney, and K. Saenko, "Translating videos to natural language using deep recurrent neural networks," *arXiv preprint arXiv:1412.4729*, 2014.

[67] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *CVPR*, 2015.

[68] J. Johnson, A. Karpathy, and L. Fei-Fei, "Densecap: Fully convolutional localization networks for dense captioning," *arXiv preprint arXiv:1511.07571*, 2015.

[69] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Lawrence Zitnick, and D. Parikh, "Vqa: Visual question answering," in *ICCV*, 2015.

[70] A. Agrawal, J. Lu, S. Antol, M. Mitchell, C. L. Zitnick, D. Batra, and D. Parikh, "Vqa: Visual question answering," *arXiv preprint arXiv:1505.00468*, 2015.

[71] C. Xiong, S. Merity, and R. Socher, "Dynamic memory networks for visual and textual question answering," *arXiv preprint arXiv:1603.01417*, 2016.

[72] P. Zhang, Y. Goyal, D. Summers-Stay, D. Batra, and D. Parikh, "Yin and yang: Balancing and answering binary visual questions," *arXiv preprint arXiv:1511.05099*, 2015.

[73] A. Bordes, N. Usunier, S. Chopra, and J. Weston, "Large-scale simple question answering with memory networks," *arXiv preprint arXiv:1506.02075*, 2015.

[74] A. Kumar, O. Irsoy, J. Su, J. Bradbury, R. English, B. Pierce, P. Ondruska, I. Gulrajani, and R. Socher, "Ask me anything: Dynamic memory networks for natural language processing," *arXiv preprint arXiv:1506.07285*, 2015.

[75] J. Weston, S. Chopra, and A. Bordes, "Memory networks," *arXiv preprint arXiv:1410.3916*, 2014.

[76] M. Malinowski, M. Rohrbach, and M. Fritz, "Ask your neurons: A neural-based approach to answering questions about images," in *CVPR*, 2015.

[77] F. Sadeghi, S. K. Divvala, and A. Farhadi, "Viske: Visual knowledge extraction and question answering by visual verification of relation phrases," in *CVPR*, 2015.

[78] M. Tapaswi, Y. Zhu, R. Stiefelhagen, A. Torralba, R. Urtasun, and S. Fidler, "Movieqa: Understanding stories in movies through question-answering," in *CVPR*, 2016.

[79] H. Zhang and D. Chiang, "Kneser-ney smoothing on expected counts."

[80] S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," in *Proceedings of the 34th annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 1996, pp. 310–318.

[81] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," *arXiv preprint arXiv:1404.2188*, 2014.

[82] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," *arXiv preprint arXiv:1705.03122*, 2017.

[83] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," *arXiv preprint arXiv:1612.08083*, 2016.

[84] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[85] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013, pp. 3111–3119.

[86] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014, pp. 1188–1196.

[87] A. M. Dai, C. Olah, and Q. V. Le, "Document embedding with paragraph vectors," *arXiv preprint arXiv:1507.07998*, 2015.

[88] T. Maharaj, N. Ballas, A. Courville, and C. Pal, "A dataset and exploration of models for understanding video data through fill-in-the-blank question-answering," *arXiv preprint arXiv:1611.07810*, 2016.

[89] Y. Yu, H. Ko, J. Choi, and G. Kim, "End-to-end concept word detection for video captioning, retrieval, and question answering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3165–3173.

[90] A. Mazaheri, D. Zhang, and M. Shah, "Video fill in the blank using lr/rl lstms with spatial-temporal attentions," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1407–1416.

[91] E. Mays, F. J. Damerau, and R. L. Mercer, "Context based spelling correction," *Information Processing & Management*, vol. 27, no. 5, pp. 517–522, 1991.

[92] C.-H. Wu, C.-H. Liu, M. Harris, and L.-C. Yu, "Sentence correction incorporating relative position and parse template language models," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1170–1181, 2010.

[93] R. A. Wagner, "Order-n correction for regular languages," *Communications of the ACM*, vol. 17, no. 5, pp. 265–268, 1974.

[94] B. Suhm, B. Myers, and A. Waibel, "Multimodal error correction for speech user interfaces," *ACM transactions on computer-human interaction (TOCHI)*, vol. 8, no. 1, pp. 60–98, 2001.

[95] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[96] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[97] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196*, 2017.

[98] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," *arXiv preprint arXiv:1805.08318*, 2018.

[99] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[100] C. Chan, S. Ginosar, T. Zhou, and A. A. Efros, "Everybody dance now," *arXiv preprint arXiv:1808.07371*, 2018.

[101] A. Siarohin, S. Lathuilière, S. Tulyakov, E. Ricci, and N. Sebe, "Animating arbitrary objects via deep motion transfer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2377–2386.

[102] C. Spampinato, S. Palazzo, P. D'Oro, F. Murabito, D. Giordano, and M. Shah, "Vos-gan: Adversarial learning of visual-temporal dynamics for unsupervised dense prediction in videos," *arXiv preprint arXiv:1803.09092*, 2018.

[103] J. Pan, C. Wang, X. Jia, J. Shao, L. Sheng, J. Yan, and X. Wang, "Video generation from single semantic label map," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[104] C. Finn, I. Goodfellow, and S. Levine, "Unsupervised learning for physical interaction through video prediction," in *Advances in neural information processing systems*, 2016, pp. 64–72.

[105] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, G. Liu, A. Tao, J. Kautz, and B. Catanzaro, "Video-to-video synthesis," *arXiv preprint arXiv:1808.06601*, 2018.

[106] A. X. Lee, R. Zhang, F. Ebert, P. Abbeel, C. Finn, and S. Levine, "Stochastic adversarial video prediction," *arXiv preprint arXiv:1804.01523*, 2018.

[107] W. Lotter, G. Kreiman, and D. Cox, "Deep predictive coding networks for video prediction and unsupervised learning," *arXiv preprint arXiv:1605.08104*, 2016.

[108] R. Villegas, J. Yang, Y. Zou, S. Sohn, X. Lin, and H. Lee, "Learning to generate long-term future via hierarchical prediction," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 3560–3569.

[109] D. Kim, S. Woo, J.-Y. Lee, and I. S. Kweon, "Deep video inpainting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5792–5801.

[110] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Generative image inpainting with contextual attention," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5505–5514.

[111] A. Clark, J. Donahue, and K. Simonyan, "Efficient video generation on complex datasets," *arXiv preprint arXiv:1907.06571*, 2019.

[112] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz, "Mocogan: Decomposing motion and content for video generation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1526–1535.

[113] Y. Pan, Z. Qiu, T. Yao, H. Li, and T. Mei, "To create what you tell: Generating videos from captions," in *Proceedings of the 25th ACM international conference on Multimedia*. ACM, 2017, pp. 1789–1798.

[114] T. Marwah, G. Mittal, and V. N. Balasubramanian, "Attentive semantic video generation using captions," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[115] Y. Li, M. R. Min, D. Shen, D. Carlson, and L. Carin, "Video generation from text," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[116] T. Xu, P. Zhang, Q. Huang, H. Zhang, Z. Gan, X. Huang, and X. He, "Attngan: Fine-grained text to image generation with attentional generative adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1316–1324.

[117] T. Hinz, S. Heinrich, and S. Wermter, "Semantic object accuracy for generative text-to-image synthesis," *arXiv preprint arXiv:1910.13321*, 2019.

[118] S. Ö. Arik, M. Chrzanowski, A. Coates, G. Diamos, A. Gibiansky, Y. Kang, X. Li, J. Miller, A. Ng, J. Raiman *et al.*, "Deep voice: Real-time neural text-to-speech," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 195–204.

[119] X. Huang, A. Acero, J. Adcock, H.-W. Hon, J. Goldsmith, J. Liu, and M. Plumpe, "Whistler: A trainable text-to-speech system," in *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP'96*, vol. 4. IEEE, 1996, pp. 2387–2390.

[120] K. Gavrilyuk, A. Ghodrati, Z. Li, and C. G. Snoek, "Actor and action video segmentation from a sentence," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5958–5966.

[121] C. Xu, S.-H. Hsieh, C. Xiong, and J. J. Corso, "Can humans fly? Action understanding with multiple classes of actors," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2015. [Online]. Available: http://web.eecs.umich.edu/~jjcorso/pubs/xu_corso_CVPR2015_A2D.pdf

[122] N. Srivastava, E. Mansimov, and R. Salakhudinov, "Unsupervised learning of video representations using lstms," in *International conference on machine learning*, 2015, pp. 843–852.

[123] C. Schuldt, I. Laptev, and B. Caputo, "Recognizing human actions: a local svm approach," in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, vol. 3. IEEE, 2004, pp. 32–36.

[124] Y. Liu, X. Wang, Y. Yuan, and W. Zhu, "Cross-modal dual learning for sentence-to-video generation," in *Proceedings of the 27th ACM International Conference on Multimedia*. ACM, 2019, pp. 1239–1247.

[125] A. Mazaheri, B. Gong, and M. Shah, "Learning a multi-concept video retrieval model with multiple latent variables," in *2016 IEEE International Symposium on Multimedia (ISM)*. IEEE, 2016, pp. 615–620.

[126] ——, "Learning a multi-concept video retrieval model with multiple latent variables," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 14, no. 2, p. 46, 2018.

[127] J. Petterson and T. S. Caetano, "Reverse multi-label learning," in *Advances in Neural Information Processing Systems*, 2010, pp. 1912–1920.

[128] O. Paul, G. Awad, M. Michel, J. Fiscus, W. Kraaij, A. Smeaton, and G. Quéenot, "Trecvid 2011-an overview of the goals, tasks, data, evaluation mechanisms and metrics," *NIST TRECVID Workshop*, 2011.

[129] W. Ping, Q. Liu, and A. Ihler, "Marginal structured svm with hidden variables," *arXiv preprint*, 2014.

[130] P. Over, J. Fiscus, G. Sanders, D. Joy, M. Michel, G. Awad, A. Smeaton, W. Kraaij, and G. Quénot, "Trecvid 2014–an overview of the goals, tasks, data, evaluation mechanisms and metrics," in *TRECVID*, 2014.

[131] K. Järvelin, S. L. Price, L. M. Delcambre, and M. L. Nielsen, "Discounted cumulated gain based evaluation of multiple-query ir sessions," in *Advances in Information Retrieval*. Springer, 2008, pp. 4–15.

[132] A. Mazaheri, M. Kalayeh, H. Idrees, and M. Shah, "Ucf-crcv at trecvid2015: Semantic indexing," 2015.

[133] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[134] A. Vedaldi and B. Fulkerson, "Vlfeat: An open and portable library of computer vision algorithms," in *ACM Multimedia*, 2010.

[135] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," *ECCV*, 2010.

[136] Y. Zhang, B. Gong, and M. Shah, "Fast zero-shot image tagging," in *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*. IEEE, 2016, pp. 5985–5994.

[137] M. Guillaumin, T. Mensink, J. Verbeek, and C. Schmid, "Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation," in *ICCV*, 2009.

[138] O. Chapelle and S. S. Keerthi, "Efficient algorithms for ranking with svms," *Information Retrieval*, vol. 13, no. 3, pp. 201–215, 2010.

[139] S. Ishikawa, M. Koskela, M. Sjöberg, J. Laaksonen, E. Oja, E. Amid, K. Palomäki, A. Mesaros, and M. Kurimo, "Picsom experiments in trecvid 2013," in *NIST TRECVID Workshop*, 2013.

[140] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[141] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *ICCV*, 2013.

[142] R. Hou, A. R. Zamir, R. Sukthankar, and M. Shah, "Damn–discriminative and mutually nearest: Exploiting pairwise category proximity for video action recognition," in *ECCV*, 2014.

[143] E. Yilmaz and J. A. Aslam, "Inferred ap: Estimating average precision with incomplete judgments," in *Fifteenth ACM International Conference on Information and Knowledge Management (CIKM)*, 2006, pp. 102–111.

[144] M. Berglund, T. Raiko, M. Honkala, L. Kärkkäinen, A. Vetek, and J. T. Karhunen, "Bidirectional recurrent neural networks as generative models," in *NIPS*, 2015.

[145] Z. Yang, X. He, J. Gao, L. Deng, and A. Smola, "Stacked attention networks for image question answering," *arXiv preprint arXiv:1511.02274*, 2015.

[146] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *ICCV*, 2015.

[147] A. Torabi, C. Pal, H. Larochelle, and A. Courville, "Using descriptive video services to create a large data source for video annotation research," *arXiv preprint*, 2015. [Online]. Available: http://arxiv.org/pdf/1503.01070v1.pdf

[148] Y. Yu, H. Ko, J. Choi, and G. Kim, "Video captioning and retrieval models with semantic attention," *arXiv preprint arXiv:1610.02947*, 2016.

[149] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *Journal of Artificial Intelligence Research*, vol. 11, 1999.

[150] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[151] 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.

[152] A. Mazaheri and M. Shah, "Visual text correction," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 155–171.

[153] E. Loper and S. Bird, "Nltk: The natural language toolkit," in *Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics-Volume 1*. Association for Computational Linguistics, 2002, pp. 63–70.

[154] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[155] H. De Vries, F. Strub, J. Mary, H. Larochelle, O. Pietquin, and A. C. Courville, "Modulating early visual processing by language," in *Advances in Neural Information Processing Systems*, 2017, pp. 6594–6604.

[156] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1874–1883.

[157] Z. Fu, Y. Zheng, H. Ye, Y. Kong, J. Yang, and L. He, "Edge-aware deep image deblurring," *arXiv preprint arXiv:1907.02282*, 2019.

[158] B. McIntosh, K. Duarte, Y. S. Rawat, and M. Shah, "Multi-modal capsule routing for actor and action video segmentation conditioned on natural language queries," *arXiv preprint arXiv:1812.00303*, 2018.

[159] P. Abolghasemi, A. Mazaheri, M. Shah, and L. Boloni, "Pay attention!-robustifying a deep visuomotor policy through task-focused visual attention," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4254–4262.

[160] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Advances in neural information processing systems*, 2016, pp. 2234–2242.

[161] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6299–6308.

[162] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev *et al.*, "The kinetics human action video dataset," *arXiv preprint arXiv:1705.06950*, 2017.

[163] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Advances in Neural Information Processing Systems*, 2017, pp. 6626–6637.

[164] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," *arXiv preprint arXiv:1802.05957*, 2018.