Evolution of A Common Vector Space Approach to Multi-Modal Problems

by

Chi Zhang

B.S. Shandong University, 2009
B.S. Jinan University, 2009
M.S. Rochester Institute of Technology, 2013

A dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy
in the Chester F. Carlson Center for Imaging Science
College of Science
Rochester Institute of Technology

October 18, 2018

Signature of the Author ____

Accepted by ____
Coordinator, Ph.D. Degree Program   Date

CHESTER F. CARLSON CENTER FOR IMAGING SCIENCE
COLLEGE OF SCIENCE
ROCHESTER INSTITUTE OF TECHNOLOGY
ROCHESTER, NEW YORK

CERTIFICATE OF APPROVAL

Ph.D. DEGREE DISSERTATION

The Ph.D. Degree Dissertation of Chi Zhang
has been examined and approved by the
dissertation committee as satisfactory for the
dissertation required for the
Ph.D. degree in Imaging Science

Dr. Carl Salvaggio, Dissertation Advisor	Date
Coordinator Ph.D. Degree Program	Date
Dr. Pengcheng Shi	Date
Dr. Raymond Ptucha	Date
Dr. Alexander Loui	Date

DISSERTATION RELEASE PERMISSION

ROCHESTER INSTITUTE OF TECHNOLOGY

COLLEGE OF SCIENCE

CHESTER F. CARLSON CENTER FOR IMAGING SCIENCE

Title of Dissertation:

Evolution of A Common Vector Space Approach to Multi-Modal Problems

I, Chi Zhang, hereby grant permission to Wallace Memorial Library of R.I.T. to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.

Signature ____

Date

Contents

List of Figures

List of Tables

Evolution of A Common Vector Space Approach to Multi-Modal Problems

by
Chi Zhang

Submitted to the
Chester F. Carlson Center for Imaging Science
in partial fulfillment of the requirements
for the Doctor of Philosophy Degree
at the Rochester Institute of Technology

Abstract

A set of methods to address computer vision problems has been developed. Video understanding is an activate area of research in recent years. If one can accurately identify salient objects in a video sequence, these components can be used in information retrieval and scene analysis. This research started with the development of a course-to-fine framework to extract salient objects in video sequences. Previous work on image and video frame background modeling involved methods that ranged from simple and efficient to accurate but computationally complex. It will be shown in this research that the novel approach to implement object extraction is efficient and effective that outperforms the existing state-of-the-art methods. However, the drawback to this method is the inability to deal with non-rigid motion.

With the rapid development of artificial neural networks, deep learning approaches are explored as a solution to computer vision problems in general. Focusing on image and text, the image (or video frame) understanding can be achieved using CVS. With this concept, modality generation and other relevant applications such as automatic image description, text paraphrasing, can be explored. Specifically, video sequences can be modeled by Recurrent Neural Networks (RNN), the greater depth of the RNN leads to smaller error, but that makes the gradient in the network unstable during training.

To overcome this problem, a Batch-Normalized Recurrent Highway Network (BNRHN) was developed and tested on the image captioning (image-to-text) task. In BNRHN, the highway layers are incorporated with batch normalization which diminish the gradient vanishing and exploding problem. In addition, a sentence to vector encoding framework that is suitable for advanced natural language processing is developed. This semantic text embedding makes use of the encoder-decoder model which is trained on sentence paraphrase pairs (text-to-text). With this scheme, the latent representation of the text is shown to encode sentences with common semantic information with similar vector representations. In addition to image-to-text and text-to-text, an image generation model is developed to generate image from text (text-to-image) or another image (image-to-image) based on the semantics of the content. The developed model, which refers to the Multi-Modal Vector Representation (MMVR), builds and encodes different modalities into a common vector space that achieve the goal of keeping semantics and conversion between text and image bidirectional. The concept of CVS is introduced in this research to deal with multi-modal conversion problems. In theory, this method works not only on text and image, but also can be generalized to other modalities, such as video and audio. The characteristics and performance are supported by both theoretical analysis and experimental results. Interestingly, the MMVR model is one of the many possible ways to build CVS. In the final stages of this research, a simple and straightforward framework to build CVS, which is considered as an alternative to the MMVR model, is presented.

Acknowledgements

I would like to extend my heartfelt thanks to the many people, in many places, who so generously contributed to the work presented in this thesis.

Special mention goes to my enthusiastic advisor, Dr. Carl Salvaggio for his fundamental role in my doctoral work. Carl provided me with every bit of guidance, assistance, and expertise that I needed during my first few semesters; then, when I felt ready to venture into research on my own and branch out into new research areas, Carl gave me the freedom to do whatever I wanted, at the same time continuing to contribute valuable feedback, advice, and encouragement. I quite simply cannot imagine a better advisor.

Similarly, profound gratitude goes to Dr. Raymond Ptucha, who has been a truly dedicated mentor. I am particularly indebted to Raymond for his constant faith in my lab work. I thank Raymond wholeheartedly, not only for his tremendous academic support, advice, and encouragement, but also for making my Ph.D. an amazing experience. In addition to our academic collaboration, I greatly value the close personal rapport that Raymond and I have forged over the years.

I am also hugely appreciative to Dr. Alexander Loui, especially for sharing his imaging expertise so willingly, and for being so dedicated to his role as my supervisor during my internship at Kodak Alaris.

I would like to thank the faculties in Chester F. Carlson Center for Imaging Science at the Rochester Institute of Technology, especially Dr. Derek Walvoord, Dr. Harvey Dhody, Dr. Roger Easton and Dr. John Kerekes, for the substantial influence that their courses have had on my research. I gratefully acknowledge the members of my Ph.D. committee for their time and valuable feedback on a preliminary version of this thesis.

Special mention goes to Shagan Sah, Thang Nguyen, Dheeraj Kumar Peri, Ameya Shringi, Dr. John Hamilton, Dr. Nathan Cahill, and all students in MIL lab who contributed to the work presented in this thesis, for for their infinite patience and our fruitful collaboration there. Among many other things, I am thankful to Jim Bodie and Brett Matzke, for providing me with a fantastic hardware and software support. And to Elizabeth Lockwood, Susan Chan and Joyce French, for handling logistics efficiently and effectively.

Finally, but by no means least, thanks go to my mum and dad for almost unbelievable support, love, and sacrifices. Without them, this thesis would never have been written. They are the most important people in my world and I dedicate this thesis to them. This last word of acknowledgment I have saved for my beloved girlfriend, Meng Wang, who has been with me all these years and has made them the best years of my life.

This thesis is dedicated to my beloved parents and all my friends for endlessly supporting, helping, and standing by me.

Chapter 1

Introduction

This research will start from the general area of multimedia analysis, especially on video background modeling, object extraction and video text summarization. With the rapid development and lower cost of smartphones and new digital capture devices, consumer videos are becoming ever popular as is evident by the large volume of YouTube video uploads, as well as video viewing on the Facebook social network. These large amount of videos pose a great challenge for users to organize, access, and retrieve their content. Hence, the ability to efficiently analyze, index, and summarize consumer videos will enable fast retrieval and intelligent re-purposing of video content for advanced and novel consumer imaging applications. This research will develop and validate a unified video analysis framework for automatically processing, analyzing, segmenting, and summarizing unstructured and unrestricted user-generated videos in the wild. In addition to multimedia analysis, this research is also concerned with the interaction between them, especially on text and image (video frame), using learning approaches. In recent years, deep learning has improved performance on many vision and language tasks individually. However, general vision or language models cannot emerge within a paradigm that focuses on the particularities of a single metric, dataset, and task. Deep learning has enabled dramatic advancement in image, video and text understanding. For example, image classification, object detection, image captioning, localized image description, image and sentence retrieval and visual question answering tasks have witnessed tremendous progress in the past few years. A unified Common Vector Space (CVS) for vision and language will be introduced in this research, which encodes different sources (not limited to image and text) by a latent vector space, where similar inputs from different modalities cluster together and dissimilar ones separate. It is expected that the combination of these resources will facilitate research in multitask learning, transfer learning, general embeddings and encoders, architecture search, zero-shot learning, general purpose question answering, meta-learning, and other related areas of vision and language.

Object level video segments are semantically meaningful spatio-temporal units such as moving people, moving vehicles, a flowing river, etc. Segmentation of a video sequence into a number of component regions would benefit many higher level vision based applications such as scene analysis, object localization and content understanding. However, single target object extraction would be a more demanding task considering consumers' needs. In many cases, a consumer video sequence simply targets capturing a single object's movement in a specific environment such as dancing, skiing, running, etc. In general, motion object detection and extraction for a static video camera is relatively straightforward since the background barely changes and a simple frame differencing would be able to extract a moving foreground object. However, it is still challenging for the object moving on a cluttered and/or dynamic background.

The goal of background modeling and foreground object extraction is to build a model of the background/foreground in an offline manner and extract the object of interest by comparing the estimated model with the frames. The model must be robust enough to cope with background changes in different ways. In recent years, a trend towards modeling spatio-temporal uniform (in terms of either appearance or motion) regions instead of single pixels has been observed [1]. These works rely on superpixels/supervoxels for object segmentation in videos. However, these methods are computationally expensive and group superpixels together according to pure spatio-temporal similarity without exploiting real-world object features. As an improvement, Giordano *et al.* [2] proposed an approach without making any specific assumptions about the videos that relies on how objects are perceived by humans according to Gestalt laws. Khoreva *et al.* [3] proposed an empirical approach to learn both the edge topology and weights of the graph. The most confident edges are selected by the graph structure while the classifiers are learned to combine features and seamlessly integrated by its accuracy. In [4] and [5], Fast Point Feature Histograms (FPFH) and Histogram of Oriented Gradients (HoG) have been used as features to represent superpixels. The high dimension feature space slows down the computation, although some improvements (e.g., [6]) were proposed to provide a better balance of trade off between segmentation quality and run time.

Much research has been devoted to graph models for segmentation, such as [3] and [7]. Fan and Loui [8] proposed a graph-based approach that models the data in a feature space, which emphasizes the correlation between similar pixels while reducing the inter-class connectivity between different objects. In [9], a reduced superpixel graph was re-weighted such that the resulting segmentation was equivalent to the full graph under certain assumptions. However, these approaches still suffer from the expense of computation and low accuracy.

Chapter 2 will investigate inter-relationship between statistical learning, segment or pixel-level classification and fine segmentation on salient objects in the video sequence. The research can be used for other applications based on the developed framework, such as video content retrieval, visual effects generation, video highlight or summarization, and video content understanding, etc. Chapter 2 first develops a novel coarse-to-fine framework and prototype system for automatically segmenting a video sequence and extracting a salient moving object from it. The developed framework is comprised of point tracking and motion clustering of pixels into groups. In parallel, a pixel grouping method is used to generate supervoxels for the corresponding frames of the video sequence. Coarse segmentation is achieved by combining the results of previous steps. Subsequently, a graph-based technique is used to perform fine segmentation and extraction of the salient object.

In addition, recent progress in deep learning using Convolutional Neural Networks (CNNs) has achieved remarkable performance on various computer vision and pattern recognition tasks, including video analysis. A video can be considered as a sequence of frames and the adjacent frames are related by consistent motion and pixel similarity in terms of pixel color and location. Therefore, video sequences can be modeled by Recurrent Neural Networks (RNNs). Typically, increasing the depth of the networks significantly reduces the error on competitive benchmarks [10]. However, training very deep networks is challenging due to the fact that the distribution of each layer's inputs changes during training. When training RNNs, gradients are unstable, and can vanish or explode over time. In some cases the gradients can vanish, the forward flow often diminishes, and the training time can be unbearably slow by requiring lower learning rates and careful parameter initialization. Sometimes the gradient gets much larger in earlier layers, which causes an exploding gradient problem. More generally, it turns out that the gradient in deep neural networks is unstable, tending to either explode or vanish in earlier layers. In such deep architectures the vanishing or exploding gradient problem becomes a key issue.

Several techniques [11, 12, 13] have been proposed to circumvent the vanishing and exploding gradient problem. Batch normalization [14] addresses the internal covariate shift problem by normalizing the layer inputs per mini-batch statistics. This speeds up training by allowing the usage of more aggressive learning rates, creates more stable models which are not as susceptible to parameter initialization, and has been shown to minimize vanishing and exploding gradients. While batch normalization has been found to be very effective for feedforward CNNs, the technique has not been as prevalent on RNNs. Laurent *et al.* [15] reported that applying batch normalization to the input-to-hidden transitions of RNNs leads to faster convergence but does not seem to improve the generalization performance on sequence modeling tasks. Cooijmans *et al.* [16] found that it is both possible and beneficial to batch normalize both the input-to-hidden and hidden-to-hidden transition, thereby reducing internal covariate shift between time steps.

In addition to batch normalization, much attention has been paid to controlling gradient behavior by changing the network structure. For instance, networks with stochastic depth [17] enable the seemingly contradictory setup to train short networks and use deep networks at test time. This approach complements the recent success of residual networks. It reduces training time substantially and improves the test error significantly on almost all data sets. Recent evidence also indicates that CNNs could benefit from an interface to explicitly construct memory mechanisms interacting with a CNN feature processing hierarchy. Correspondingly, the convolutional residual memory network [18] was proposed as a memory mechanism which enhances CNN architecture based on augmenting convolutional residual networks with a Long Short-Term Memory (LSTM) [11] mechanism. Weight normalization [19] was reported to be better suited for recurrent models such as LSTMs compared to batch normalization. It improves the conditioning of the optimization problem and speeds up convergence of stochastic gradient descent without introducing any dependencies between the examples in a mini-batch. Similarly, layer normalization [20] normalizes across the inputs on a layer-by-layer basis at each time step. This stabilizes the dynamics of the hidden layers in the network and accelerates training, without the limitation of being tied to a batched implementation.

Chapter 3 develops a novel recurrent framework based on Recurrent Highway Networks (RHNs) for sequence modeling using batch normalization. This chapter explores the differences of several state-of-the-art techniques in terms of gradient control in data propagation within recurrent networks and compares the performance between them. The developed technique relaxes the constraint in RHNs such that they have a better chance to avoid the gradient from vanishing or exploding by normalizing the recurrent transition units in highway layers. Since this work is able to deal with applications with RNN structure, such as captioning, it turns out that this technique can be used in Multi-Modal Vector Representation (MMVR) as illustrated in Chapter 5.

Exploring the new RNN framework brings about benefits for multimedia conversion, for example, captioning, which is also known as image-to-text conversion. Text-to-text conversion also needs to be paid attention by using a sequence-to-sequence model. Modeling temporal sequences of patterns requires the embedding of each pattern into a vector space. For example, by passing each frame of a video through a CNN, a sequence of vectors can be obtained. These vectors are fed into a RNN to form a powerful descriptor for video annotation [21, 22, 23]. Similar, techniques such as word2vec [24] and GloVe [25] have been used to form vector representations of words. Using such embeddings, sentences become a sequence of word vectors. When these vector sequences are fed into an RNN, it generates a powerful descriptor of a sentence [26].

Given vector representations of a sentence and video, the mapping between these vector spaces can be solved, forming a connection between visual and textual spaces. This enables tasks such as captioning, summarizing, and searching of images and video to become more intuitive for humans. By vectorizing paragraphs [27], similar methods can be used for richer textual descriptions.

Recent advances at vectorizing sentences represent exact sentences faithfully [28, 27, 29], or pair a current sentence with prior and next sentence [30]. Similar to word2vec and GloVe map words of similar meaning close to one another, the desired method is to map sentences of similar meaning close to one another. For example, the sentences "A man jumped over the stream" and "A person hurdled the creek" have similar meaning to humans, but are not close in traditional sentence vector representations. Just like the words flower, rose, and tulip are close in good sentence to vector representations, the example sentences must lie close in the introduced embedded vector space.

Inspired by the METEOR [31] captioning benchmark which allows substitution of similar words, it is desired to map similar sentences as close as possible. Both paraphrase datasets and ground truth captions from multi-human captioning data sets are utilized. For example, the MS-COCO data set [32] has over 120K images, each with five captions from five different evaluators. On average, each of these five captions from each image should convey the same semantic meaning. Chapter 4 presents an encoder-decoder framework for sentence paraphrases, and generate the vector representation of sentences from this framework which maps sentences of similar semantic meaning nearby in the vector encoding space.

The main contributions of this semantic sentence embedding method are 1) the usage of sentences from widely available image and video captioning data sets to form sentence paraphrase pairs, whereby these pairs are used to train the encoder-decoder model; 2) the demonstration of the application of the sentence embeddings for paragraph summarization and sentence paraphrasing, whereby evaluations are performed using metrics, vector visualizations and qualitative human evaluation; and 3) the extention of the vectorized sentence approach to a hierarchical architecture, enabling the encoding of more complex structures such as paragraphs for applications such as text summarization. Chapter 4 presents the developed encoder-decoder framework for sentence and paragraph paraphrasing.

Looking at the exploration of the relationship between multimedia and vectors, it is reasonable to extend the goal of this dissertation to find the common vector representation for different types of sources. In other words, given a(n) video sequence/image/audio/sentence/paragraph, the goal is to extract an embedded vector containing the semantics of the source, and decode it to any type of the multimedia. The embedded vectors from different types of sources lie in a common space so that there is no need to align the vectors in the generative models. As an example, image captioning converts an image into a vector, and then the vector is used to generate text. In the other direction, the vector representation should be able to generate an image with related content. The common vectors form a space referred to as common vector space (CVS).

For simplicity, the study is focusing on images and text. Specifically, the common vector space deals with four source-target conversion: text-text (text2text, which can be described as a sentence paraphrasing model detailed in Chapter 4), image-text (im2text, as known as image captioning model as illustrated in Chapter 3), text-image (text2im, as illustrated in Chapter 5), and image-image (im2im, as illustrated in Chapter 5).

Recent success in image captioning [33, 34, 35, 36] has shown that deep networks are capable of providing apt textual descriptions of visual data. In parallel, advances in conditioned image generation [37, 38, 39, 40] provide diverse images from a text based prior. An ambitious goal for machine learning in the vision and language domain is to be able to represent different modalities of data that have the same meaning with a common latent representation. For example, words like "baseball" and "batter", a sentence describing a baseball game, or image representations of a baseball game all refer to similar concepts. Generally, concepts that are semantically similar would lie close together in the descriptor's space while dissimilar concepts would lie farther apart. A sufficiently powerful model should be able to store similar concepts in a similar representation or produce any of these realizations from the same latent space. Successfully mapping visual and textual modalities in and out of this latent space would significantly impact the broad task of information retrieval.

Chapter 5 develops a cross-domain model, based on Plug & Play Generative Networks (PPGN) [37] architecture, capable of converting between text and image. The networks used in these domains are combined by merging the latent representations obtained during transition. The goal of the latent vector representation is to encourage similar patches to have descriptors that are closer to each other in the descriptors' space than dissimilar ones. The contributions of this model are as follows: 1) The formulation of a latent representation based model that merges inputs across multiple modalities; 2) Development of an n-gram based cost function that generalizes better to a text prior; 3) Improvements on image quality while using multiple semantically similar sentences for conditioning image generation on generalized text; and 4) To advance qualitative measurement of text-to-visual models, an object detector based metric is introduced, and the human evaluations are conducted which compare the metric to the standard inception score [41].

In addition to the PPGN based model, the concept of a unified Common Vector Space (CVS) for vision and language is introduceed, that spans across five broad tasks: classification, captioning, object detection, retrieval and visual question answering. Chapter 6 first summarizes the work we have done, then introduces the idea of learning a Common Vector Space (CVS) where similar inputs from different modalities cluster together. It is expected that the combination of these resources will facilitate research in multi-task learning, transfer learning, general embeddings and encoders, architecture search, zero-shot learning, general purpose question answering, meta-learning, and other related areas of vision and language. The contribution of the CVS model would be: 1) Formulating an efficient vector space based model using neural embeddings that act as a bridge between vision and language modalities and is easily expandable to new modalities; 2) Introducing a multi-modal loss function that includes metric loss, category loss and adversarial loss terms. The adversarial framework includes within-modality and across modality discriminators; and 3) It is capable to applying this model to new tasks, such as sentence/image retrieval, object localization and captioning, etc.

Chapter 2

Spatio-Temporal Video Segmentation

Object level segmentation of the video sequence would benefit many higher level vision based applications such as scene analysis, object localization and content understanding. However, single target object extraction would be a more demanding task considering consumers' needs. In many cases, a consumer video sequence simply targets capturing a single object's movement in a specific environment. In general, it is challenging for the object moving on a cluttered and/or dynamic background, since simple frame differencing would not be able to extract a moving foreground object.

The goal of background modeling and foreground object extraction is to build a model of the background/foreground in an offline manner and extract the object of interest by comparing the estimated model with the frames. Recently, modeling on spatio-temporal uniform regions [1] becomes popular approaches. Moreover, researchers have focused on graph models for segmentation, such as [3] and [7]. However, these approaches still suffer from the expense of computation and low accuracy.

In this chapter, a discussion that focuses on the coarse-to-fine framework and prototype for automatic video sequence segmentation and salient moving object extraction are presented. Effectively, voxel grouping techniques are often used to generate supervoxels for video sequences and describe how similar the adjacent voxels are in terms of appearance and motion. At a top-level, segmentation of video sequences becomes a problem of classification of supervoxels into foreground or background. With the segmentation results, some interesting and pleasing visual effects can be created easily for consumers. This is known as video object recomposition.

The developed framework is comprised of point tracking algorithms and motion clustering of pixels into groups. In parallel, a pixel grouping method is used to generate supervoxels for the corresponding frames of the video sequence. Coarse segmentation is achieved by combining the results of previous steps. Subsequently, a graph-based segmentation technique is used to perform fine segmentation and extraction of the salient object. Section 2.1 outlines some related work and approaches on video modeling and moving object segmentation. Section 2.2 first gives an overview of the developed coarse-to-fine video segmentation framework and the system workflow, and then describes the details of the key algorithms and components of the developed framework. Section 2.3 discusses the performance evaluations and experimental results of the developed framework and algorithms. Finally, some discussions and future work are presented in Section 2.4.

2.1 Related Work

The goal of background modeling and foreground object extraction is to build a model of the background/foreground in an off-line manner and extract the object of interest by comparing the estimated model with the frames. The model must be robust enough to cope with background changes in different ways. This section reviews some of the relevant state-of-the-art methods on video segmentation in terms of the following aspects:

Superpixel/Supervoxel Based Approaches. In recent years, a trend towards modeling spatio-temporal uniform (in terms of either appearance or motion) regions instead of single pixels has been observed. These works rely on superpixels/supervoxels for object segmentation in videos. The core idea is that in superpixels appearance and motion are more or less uniform, thus estimated density functions are likely to be quite accurate. However, these methods 1) need to compute the motion field through optical flow, which is computationally expensive; 2) group superpixels together according to pure spatio-temporal similarity (in terms of appearance) without exploiting real-world object features; and 3) produce segmentation through global minimization of an energy function, thus considering video object segmentation as a single objective optimization problem, while, in fact, it is intrinsically multi-objective. As an improvement, Giordano *et al.* [2] proposed an approach without making any specific assumptions about the videos and it relies on how objects are perceived by humans according to Gestalt laws. This methods is able to segment objects in crowded scenes and accurately segments complex articulated objects. From another point of view, meaningful features are necessary in frame partitions for good video segmentation. Much literature [42, 43] has proposed features for appearance, motion or shape similarities among the graph nodes. Most works are currently limited in the number of features they can leverage, as often the researchers hand-design the feature combination to measure similarity between pixels or superpixels. Khoreva *et al.* [3] proposed an empirical approach to learn both the edge topology and weights of the graph. The most confident edges are selected by the graph structure while the classifiers are learned to combine features and seamlessly integrated by its accuracy. In [5] and [4], HoG and FPFH have been used as features to represent superpixels. The high dimension feature space slows down the computation, although some improvements (for example, [6]) were proposed to provide a better balance of trade off between segmentation quality and runtime.

Graph Based Approaches. Much research has been devoted to graph partitioning models [44, 45]. While measurable differences have been observed, Khoreva *et al.* [3] intentionally focused on the graph construction problem instead, and adopted the recent and successful graph partitioning model [7] based on spectral clustering. Fan and Loui [8] proposed a graph-based approach that effectively models the data in a high-dimensional feature space, which emphasizes the correlation between similar pixels while reducing the inter-class connectivity between different objects. The graph model fuses appearance' spatial and temporal information to break a volumetric video sequence into semantic spatiotemporal key-segments. In the work of Li *et al.* [9], the reduced superpixel graph was reweighted such that the resulting segmentation was equivalent to the full graph under certain assumptions. Among this type of algorithms, constructing the graph is a vital step for ensuring the performance of clustering methods. Although graph-based methods have been extensively studied, there have been limited efforts for building effective graphs. The most popular method for constructing a sparse graph is the nearest neighbor approach, including different variants such as k-nearest neighbor and ϵ-nearest neighbor methods.

Learning Based Approaches. Learning based models consist one of the major themes in image and video segmentation. They learn the appearance of semantic categories, under various transformations, and the relations among them using parametric models. Conditional Random Field (CRF) based image models have been quite successful in jointly modeling the appearance and structure of an image. [46] used CRFs to combine unary potentials obtained from the visual features of superpixels with the neighborhood constraints. Multi-scale convolution neural networks were used in [47] to learn visual feature extractors from raw-image/label training pairs. It achieved impressive results on various datasets using gPb, purity-cover and CRF on top of the learned features. It was extended in [48] by feeding in the per-pixel predicted labels using a Convolutional Neural Network (CNN) classifier to the next stage of the same CNN classifier. Long *et al.* [49] showed that convolutional networks by themselves, trained end-to-end, pixels-to-pixels, exceed the state-of-the-art in semantic segmentation. The key insight is to build "fully convolutional" networks that take input of arbitrary size and produce correspondingly-sized output with efficient inference and learning. Cimpoi *et al.* [50] conducted a study of material and describable texture attributes recognition in clutter, and proposed a texture descriptor, Fisher Vector pooling of a CNN filter bank. FV-CNN substantially improves the state-of-the-art in texture, material and scene recognition. Sharma *et al.* [51] proposed a learning-based approach to scene parsing inspired by the deep Recursive Context Propagation Network (RCPN). RCPN is a deep feed-forward neural network that utilizes the contextual information from the entire image, through bottom-up followed by top-down context propagation via random binary parse trees. This improved the feature representation of every superpixel in the image for better classification into semantic categories.

Supervised vs. Unsupervised Approaches. Video object segmentation approaches in the current literature can be grouped into supervised or unsupervised categories. Supervised (and semi-supervised) approaches typically act through training label classifiers [50, 51] or propagating user-annotated labels over time [52, 53]. Supervised learning is a way to improve the performance of segmentation for specific tasks. Teney *et al.* [52] improved hierarchical video segmentation with supervised learning. They optimized a metric between segment descriptors over labeled training data, using a large-margin formulation suitable for hierarchical segmentation. Although being well studied for a long period, such methods are limited to a small range of applications due to the extreme dependence on labor-intensive pixel annotations to train suitable models. Unsupervised approaches generally focus on segmenting the most primal object in a single video and co-segmenting the common object among a video collection [54, 55, 56]. Dong *et al.* [57] proposed to densely extract object segments with high objectness and smooth evolvement based on directed acyclic graph. Papazoglou *et al.* [58] developed a fast object segmentation approach that quickly estimates rough object configurations through the use of inside-outside maps. In addition, weakly supervised approaches have received growing attention for their convenience in gathering video-level labels and the prospect in analyzing web-scale data. Existing algorithms employed variants on the learning techniques to predict the confidence of each pixel belonging to a given concept.

Zhang *et al.* [59] proposed a segmentation approach based on semantic objects in weakly labeled videos via object detection. Zhang *et al.* [60] learned a weakly supervised semantic segmentation model from social images whose labels might be noisy and are not pixel level but image-level.

2.2 System Framework and Algorithms

The developed coarse-to-fine framework [61] is illustrated in Figure 2.1, and consists of several stages: 1) The point tracking algorithms are applied to the consecutive frames of the input video, and then 2) these tracking points are clustered into groups; in parallel, 3) a pixel grouping method is used to generate supervoxels for the corresponding frame of the video sequence; 4) the coarse segmentation is achieved by combining the results of previous steps; 5) the graph-based segmentation technique is used to perform fine segmentation and generate a mask of the most salient object, and finally, 6) visual effects are created based on the segmentation results.

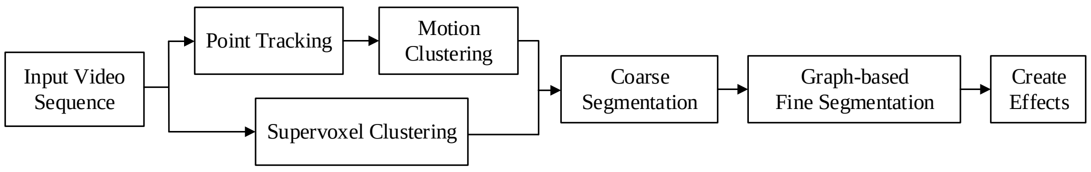


Figure 2.1: The overall coarse-to-fine framework for video segmentation and recomposition.

This video segmentation scheme exhibits state-of-the-art boundary adherence, improves the performance of segmentation algorithms, and produces interesting and pleasing visual outputs, with reduced memory consumption. This approach is a major enhancement to the previous graph-based framework [8], with the following distinctions and advantages:

- This scheme deals with the video sequence with any resolution and any length, *i.e.*, there is no restriction on the size of the video. For a long video sequence, it is segmented into small clips that are processed by the system one by one.

- The parallel approach combines the spatial and temporal information and takes advantages of both graph-based algorithms and pixel grouping methods. Consequently it provides a remarkable improvement on accuracy and speed.
- It is an unsupervised scheme, *i.e.*, there is no user interaction required to generate the accurate object mask. The desired output visual effects can be easily created by the predefined parameters.

This section contains the details of the system framework. Given the full accessibility of consumer videos, the point tracking and clustering can be processed all at once instead of one frame at a time. In parallel, the 3D Simple Linear Iterative Clustering (SLIC) supervoxels are generated in each segment of the entire video sequence. After combining the results of point clustering and supervoxel grouping, the graph-based approach, typically GrabCut, is used to provide a fine segmentation.

2.2.1 Points Tracking

There are a lot of widely-used point tracking algorithms, and each of them has its own characteristics. Particle filtering [62] is good at seeking the global optimal solution, but this algorithm is not fast enough. In addition, the color histogram based calculation leads to the weakness of distinguishing the objects with similar color. Mean shift tracking [63] overcomes the difficulty on computing speed, however, it is easy to run into local optimum.

Another popular and well-performed video object tracking algorithm is the Kanade-Lucas-Tomasi (KLT) point tracker. This algorithm is based on the early work of Lucas and Kanade [64], and later was developed fully by Tomasi and Kanade [55]. The algorithm basically provides the trajectories of a bundle of points. The KLT points tracker requires some prerequisites: 1) the luminance between two adjacent frames could be considered as constant; 2) the object moves under a continuous time slot, otherwise the movement should be "small" enough; 3) a point and its neighborhood have similar motion vector, *i.e.*, spatially consistent. In theory, if the window w in frame I is the same as that in the adjacent frame J, we have $I(x, y, t) = J(x', y', t+\tau)$. The constant-luminance hypothesis holds the equality and gets rid of the effect of luminance changes. The second premise ensures the existence of the tracking points. The points in the same window that have the same offset are guaranteed by the third premise.

Mathematically, in the window w, all the points (x, y) move to (x', y') by the offset (dx, dy), *i.e.*, the point (x, y) at time t corresponds the point $(x + dx, y + dy)$ at time $t + \tau$. Based on this fact, the point matching problem can be described by looking for the minimum of the equation below

$$\varepsilon(\mathbf{d}) = \varepsilon(d_x, d_y) = \sum_{x\in w}\sum_{y\in w}[J(x + d_x, y + d_y) - I(x, y)]^2 \tag{2.1}$$

In continuous representation,

$$\varepsilon(\mathbf{d}) = \iint_W \left[J\left(\mathbf{x} + \frac{\mathbf{d}}{2}\right) - I\left(\mathbf{x} - \frac{\mathbf{d}}{2}\right)\right]^2 w(\mathbf{x})d\mathbf{x} \tag{2.2}$$

implies the difference in the window with center $\mathbf{x} - \dfrac{\mathbf{d}}{2}$ in the frame I and $\mathbf{x} + \dfrac{\mathbf{d}}{2}$ in the frame J, and side length $w/2$. In order to find the minimum, let

$$\frac{\partial\varepsilon(\mathbf{d})}{\partial\mathbf{d}} = 0 \tag{2.3}$$

where

$$\frac{\partial\varepsilon(\mathbf{d})}{\partial\mathbf{d}} = 2\iint_W \left[J\left(\mathbf{x} + \frac{\mathbf{d}}{2}\right) - I\left(\mathbf{x} - \frac{\mathbf{d}}{2}\right)\right]\left[\frac{\partial J\left(\mathbf{x} + \frac{\mathbf{d}}{2}\right)}{\partial\mathbf{d}} - \frac{\partial J\left(\mathbf{x} - \frac{\mathbf{d}}{2}\right)}{\partial\mathbf{d}}\right] w(\mathbf{x})d\mathbf{x} \tag{2.4}$$

Applying a Taylor-series expansion, we have

$$J\left(\mathbf{x}+\frac{\mathbf{d}}{2}\right) \approx J(\mathbf{x}) + \frac{d_x}{2}\frac{\partial J}{\partial x}(\mathbf{x}) + \frac{d_y}{2}\frac{\partial J}{\partial y}(\mathbf{x}) \tag{2.5}$$

$$I\left(\mathbf{x}-\frac{\mathbf{d}}{2}\right) \approx I(\mathbf{x}) - \frac{d_x}{2}\frac{\partial I}{\partial x}(\mathbf{x}) - \frac{d_y}{2}\frac{\partial I}{\partial y}(\mathbf{x}) \tag{2.6}$$

The problem becomes

$$\frac{\partial \varepsilon(\mathbf{d})}{\partial \mathbf{d}} \approx \iint_W [J(\mathbf{x}) - I(\mathbf{x}) + \mathbf{g}^T(\mathbf{x})\mathbf{d}]\mathbf{g}(\mathbf{x})w(\mathbf{x})d\mathbf{x} \tag{2.7}$$

where

$$\mathbf{g} = \left[\frac{\partial}{\partial x}\left(\frac{I+J}{2}\right) \frac{\partial}{\partial y}\left(\frac{I+J}{2}\right)\right]^T \tag{2.8}$$

Rearranging the equation above yields

$$\iint_W [J(\mathbf{x}) - I(\mathbf{x})]\mathbf{g}(\mathbf{x})w(\mathbf{x})d\mathbf{x} = -\iint_W \mathbf{g}^T(\mathbf{x})\mathbf{d}\mathbf{g}(\mathbf{x})w(\mathbf{x})d\mathbf{x} \tag{2.9}$$

$$= -\left[\iint_W \mathbf{g}(\mathbf{x})\mathbf{g}^T(\mathbf{x})w(\mathbf{x})d\mathbf{x}\right] d \tag{2.10}$$

Now this equation has the form of

$$Z\mathbf{d} = \mathbf{e} \tag{2.11}$$

where

$$Z = \iint_W \mathbf{g}(\mathbf{x})\mathbf{g}^T(\mathbf{x})w(\mathbf{x})d\mathbf{x} \tag{2.12}$$

is a 2×2 matrix and

$$\mathbf{e} = \iint_W [I(\mathbf{x}) - J(\mathbf{x})]\mathbf{g}(\mathbf{x})w(\mathbf{x})d\mathbf{x} \tag{2.13}$$

is a 2×1 vector. The matrix ZZ^T has to be invertible to ensure the existence of the solution. Generally, the corner points have such property.

In this work, the points to be tracked are selected in a grid-based manner in order to make the initial points distributed uniformly in the entire frame, as shown by the red dots in Figure 2.2(a). As the point tracking algorithm progresses over time, points can be lost due to lighting variation, out of plane rotation, or articulated motion as shown in Figure 2.2(b) and Figure 2.2(c). To track an object over a long period of time, it is better to reacquire points periodically.

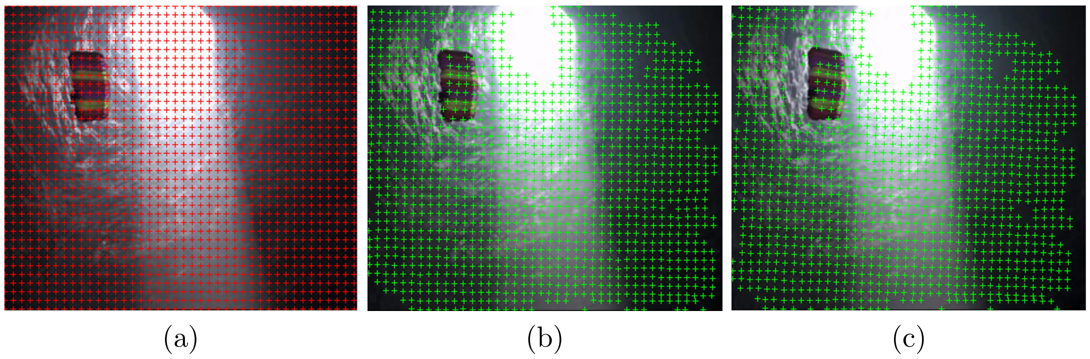


Figure 2.2: KLT point tracking. (a) Selected tracking points in the 1st frame; (b) and (c) Tracking points in the 3rd and 5th frame.

There are some algorithms proposed to improve the accuracy of KLT points tracking. One practical includes TLD algorithm proposed by Kalal [65].

Alternatively, tasks such as object tracking can be done by making use of optical flow methods. In the case of a continuous video sequence, the tracking points in a frame are extracted by feature point detector, such as SIFT or Harris corner detector, or manually selected depending on the application of the system. Optical flow methods look for the optimal location of these tracking points in any specific frame after that. The point tracking is achieved by running this process iteratively. However, this method is time-consuming.

2.2.2 Motion Clustering

In the video segmentation problems, the collection of points in a video sequence is located in the high-dimensional space. Often, high-dimensional data lie close to low-dimensional structures corresponding to several classes or categories the data belongs to. The Sparse Subspace Clustering (SSC) algorithm clusters tracking points that lie in a union of low-dimensional subspaces [66]. The key idea is that, among infinitely many possible representations of a data in terms of other points, a sparse representation corresponds to selecting a few points from the same subspace. This motivates solving a sparse optimization program whose solution is used in a spectral clustering framework to infer the clustering of data into subspaces. Since solving the sparse optimization program is in general NP-hard, a convex relaxation is considered, and it is shown that under appropriate conditions, on the arrangement of subspaces and the distribution of data, the minimization program succeeds in recovering the desired sparse representations. The algorithm can be solved efficiently and can handle data points near the intersections of subspaces. Another key advantage of this algorithm with respect to the state-of-the-art is that it can deal with data nuisances, such as noise, sparse outlying entries, and missing entries, directly by incorporating the model of the data into the sparse optimization program.

The underlying idea behind the SSC algorithm is the "self-expressiveness" property, meaning that each data point in a union of subspaces can be efficiently represented as a linear or affine combination of other points in the dataset. Based on this fact, let $\{S_\ell\}_{\ell=1}^n$ be an arrangement of n linear subspaces of $\mathbb{R}^D$ of dimensions $\{d_\ell\}_{\ell=1}^n$. Consider a given collection of N tracking points $\{\mathbf{y}_i\}_{i=1}^N$ that lie in the union of the n subspaces. Denote the matrix containing all data point as

$$\mathbf{Y} \triangleq [\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_N] = [\mathbf{Y}_1, \mathbf{Y}_2, \cdots, \mathbf{Y}_n]\mathbf{\Gamma} \tag{2.14}$$

where $\mathbf{Y}_\ell \in \mathbb{R}^{D \times N_\ell}$ is a rank-d_ℓ matrix of $N_\ell > d_\ell$ points that lie in S_ℓ, and $\mathbf{\Gamma} \in \mathbb{R}^{N \times N}$ is an unknown permutation matrix. The subspace clustering problem refers to the problem of finding the number of subspaces, their dimensions, a basis of each subspace, and the segmentation of the data from $\mathbf{Y}$. Each data point can be written as

$$\mathbf{y}_i = \mathbf{Y}\mathbf{c}_i, \; c_{ii} = 0 \tag{2.15}$$

where $\mathbf{c}_i \triangleq [c_{i1}, c_{i2}, \cdots, c_{iN}]^T$ and the constraint $c_{ii} = 0$ eliminates the trivial solution of writing a point as a linear combination of itself. In other words, the matrix of data points $\mathbf{Y}$ is a self-expressive dictionary in which each point can be written as a linear combination of other points. However, the representation of $\mathbf{y}_i$ in the dictionary $\mathbf{Y}$ is not unique in general. This comes from the fact that the number of data points in a subspace is often greater than its dimension, *i.e.*, $N_\ell \leq d_\ell$. As a result, each $\mathbf{Y}_\ell$, and consequently $\mathbf{Y}$, has a non-trivial null space giving rise to infinitely many representations of each data point.

A data point $\mathbf{y}_i$ that lies in the d_ℓ-dimensional subspace S_ℓ can be written as a linear combination of d_ℓ other points in general directions from S_ℓ. As a result, ideally, a sparse representation of a data point finds points from the same subspace where the number of the non-zero elements corresponds to the dimension of the underlying subspace.

For a system equation, such as (2.15), with infinitely many solutions, one can restrict the set of solutions by minimizing an objective function such as the ℓ_q-norm of the solution as

$$\min \|\mathbf{c}_i\|_q \tag{2.16}$$

$$\text{s.t. } \mathbf{y}_i = \mathbf{Y}\mathbf{c}_i, \; c_{ii} = 0 \tag{2.17}$$

where the ℓ_q-norm is defined as

$$\|\mathbf{c}_i\|_q \triangleq \left(\sum_{j=1}^{N} |c_{ij}|^q \right)^{\frac{1}{q}} \tag{2.18}$$

Typically, by decreasing the value of q from infinity toward zero, the sparsity of the solution increases. The extreme case of $q \to 0$ corresponds to the general NP-hard problem of finding the sparsest representation of the given point, as the ℓ_0-norm counts the number of non-zero elements of the solution. Since this research is interested in efficiently finding a non-trivial sparse representation of $\mathbf{y}_i$ in the dictionary $\mathbf{Y}$, minimizing the tightest convex relaxation of the ℓ_0-norm is considered, which can be solved efficiently using convex programming tools and is known to prefer sparse solutions. The sparse optimization problem for all data points $i = 1, \cdots, N$ can be rewritten in matrix form as

$$\min \|\mathbf{C}\|_1 \tag{2.19}$$

$$\text{s.t. } \mathbf{Y} = \mathbf{Y}\mathbf{C}, \ \operatorname{diag}(\mathbf{C}) = 0 \tag{2.20}$$

where $\mathbf{C} \triangleq [\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_N] \in \mathbb{R}^{N\times N}$ is the matrix whose i-th column corresponds to the sparse representation of $\mathbf{y}_i$, $\mathbf{c}_i$, and $\operatorname{diag}(\mathbf{C}) \in \mathbb{R}^N$ is the vector of the diagonal elements of $\mathbf{C}$.

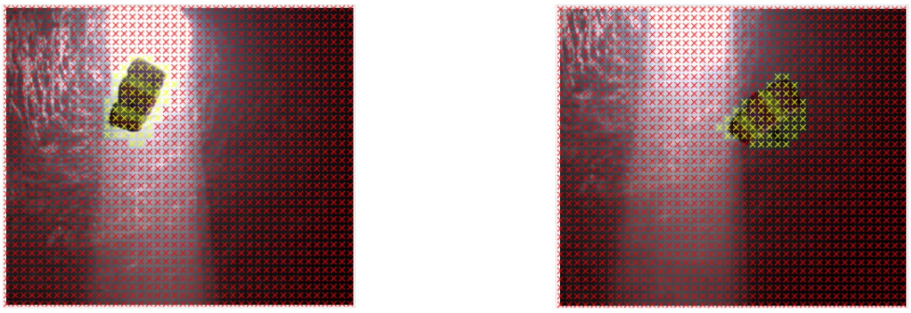
The next step is to infer the segmentation of the data into different subspaces using the sparse coefficients. To address this problem, a weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, where $\mathcal{V}$ denotes the set of N nodes of the graph corresponding to N data points and $\mathcal{E} \in \mathcal{V} \times \mathcal{V}$ denotes the set of edges between nodes. $\mathbf{W} \in \mathbb{R}^{N\times N}$ is a symmetric non-negative similarity matrix representing the weights of the edges, *i.e.*, node i is connected to node j by an edge whose weight is equal to w_{ij}. An ideal similarity matrix $\mathbf{W}$, hence an ideal similarity graph $\mathcal{G}$, is one in which nodes that correspond to points from the same subspace are connected to each other and there are no edges between nodes that correspond to points in different subspaces. Note that the sparse optimization problem ideally reverts to a subspace-sparse representation of each point, *i.e.*, a representation whose non-zero elements correspond to points from the same subspace of the given data point. This provides an immediate choice of the similarity matrix as $\mathbf{W} = |\mathbf{C}| + |\mathbf{C}|^T$. In other words, each node i connects itself to a node j by an edge whose weight is equal to $|c_{ij}|+|c_{ji}|$. The reason for the symmetrization is that, in general, a data point $\mathbf{y}_i \in S_\ell$ can write itself as a linear combination of some points including $\mathbf{y}_j \in S_\ell$. However, $\mathbf{y}_j$ may not necessarily choose $\mathbf{y}_i$ in its sparse representation. This particular choice of the weight ensures that nodes i and j get connected to each other if either $\mathbf{y}_i$ or $\mathbf{y}_j$ is in the sparse representation of the other.

The similarity graph built this way has, ideally, n connected components corresponding to the n subspaces, *i.e.*,

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}_1 & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{W}_n \end{bmatrix} \mathbf{\Gamma} \tag{2.21}$$

where $\mathbf{W}$ is the similarity matrix of data points in S_ℓ. Clustering of data into subspaces then follows by applying spectral clustering to the graph $\mathcal{G}$. More specifically, the clustering of data is obtained by applying the k-means algorithm to the normalized rows of a matrix whose columns are the n bottom eigenvectors of the symmetric normalized Laplacian matrix of the graph.

The point trajectories acquired by the KLT point tracker are grouped into two clusters using SSC algorithm. Figure 2.3 shows the clustering results on two frames. Due to the fact that the object moves in a different way than the background does, the tracking points on the object are separated from the points on the background.



(a) (b)

Figure 2.3: Demonstrations of point trajectory clustering using SSC algorithm on two frames. The yellow and red markers represent two clusters, foreground and background respectively.

2.2.3 Supervoxel Clustering

Simple linear iterative clustering (SLIC) is an accurate, fast and memory efficient algorithm proposed by Achanta *et al.* [67] to generate superpixels. This approach can be extended to 3D space for dealing with 3D data clustering problem such as video sequence segmentation.

Considering the aspect of computational efficiency, the entire video sequence is cut into clips and each chip contains a fixed number of frames which is determined and adjusted by the computing ability of the processor. Each clip can then be processed individually. Since the consumer videos are usually taken by users' portable video cameras or even cellphones, the resolution of consumer videos is sometimes comparable or higher than 720p HD videos. Those kinds of videos contain too much details in each frame and cause undesired effects and redundant computations on 3D SLIC performance. In order to solve this problem, a bilateral filter [68] can be used on each frame in the clip. The intensity value of each pixel is replaced by the weighted average intensity values from neighboring pixels so that the edges around the objects are preserved and the other regions are smoothed. Also, bilateral filtering reduces the noise in each channel.

Suppose that the desired number of supervoxels on each frame is n and the depth of each supervoxel is D along the temporal axis. Assuming that the supervoxels are initially square in each frame and approximately equal-sized. All cluster centers are initialized by sampling the clip on a regular grid spaced S pixel apart inside each frame and t pixel between frames (along temporal axis). Therefore, the actual total number of supervoxel is determined by

$$k = k_x \times k_y \times k_z \tag{2.22}$$

where k_x, k_y and k_z are the number of supervoxels along the x, y and z (temporal) directions:

$$k_x = \frac{\text{total number of rows}}{S} \tag{2.23}$$

$$k_y = \frac{\text{total number of columns}}{S} \tag{2.24}$$

$$k_z = \frac{\text{number of frames in each clip}}{S} \tag{2.25}$$

Without considering the accuracy for small color differences, the video sequence is converted into CIELAB space. Each cluster is then represented by the vector

$$C = \begin{bmatrix} x & y & z & L^* & a^* & b^* & u & v \end{bmatrix} \tag{2.26}$$

where x and y represent the special location and z carries the temporal information, L^*, a^* and b^* represent the spectral information and u, v represent the motion information extracted by optical flow.

In the assignment, the cluster of each pixel is determined by calculating the distance between the pixel itself and the cluster center in the search region with size $2S \times 2S \times 2D$, as shown in Figure 2.4. The problem arises when the distance is measured. In this case, the distances in each domain are calculated separately and then combined after multiplying by the appropriate weights, *i.e.*, the distance d is defined by the pixel location, the CIELAB color space and motion vector in the image as follows:

$$d = \sqrt{\frac{d_l^2}{2S^2 + D^2} + \frac{d_c^2}{m} + \frac{w_m d_m^2}{RS}} \tag{2.27}$$

where m is the regularity that controls the compactness of the supervoxel, w_m is a weight

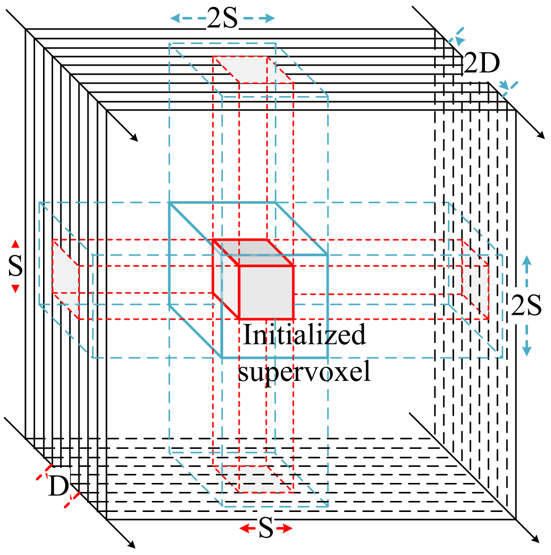


Figure 2.4: Initialization and the search region of the supervoxel. Red box shows the initialized supervoxel along D consecutive frames. Blue box is the searching area for this cluster. Each pixel is calculated eight times since it enclosed by eight cluster search region.

on motion information, R is frame rate, and

$$d_l = \sqrt{\Delta x^2 + \Delta y^2 + w_z \cdot \Delta z^2} \tag{2.28}$$

$$d_c = \sqrt{w_{L^*} \cdot \Delta L^{*2} + \Delta a^{*2} + \Delta b^{*2}} \tag{2.29}$$

$$d_m = \sqrt{\Delta u^2 + \Delta v^2} = \sqrt{\Delta \dot{x}^2 + \Delta \dot{y}^2} \tag{2.30}$$

where w_z and w_{L^*} are the weights for temporal distance and L^* channel. In the distance measure, the location is normalized by the maximum distance in the 3D lattice $2S^2 + D^2$ according to Figure 2.4. The weight for the depth component w_z is introduced since the inter-frame (lateral) position distance should be treated differently as in-frame (transverse) distance. Considering two adjacent supervoxels with depth D in the temporal axis, these two supervoxels would shrink transversely and expand up to 2D in lateral direction during the iterations if the region surrounded is relatively uniform and the weight w_z is small. This causes the increased number of clusters on a single frame, which is unexpected for some applications.

Similar to some graph-based algorithms [69], 3D SLIC does not explicitly enforce connectivity. For some clusters, the pixels are merged by the adjacent supervoxels and only a small group of pixels (sometimes only one pixel) is retained in the cluster to keep the total number of clusters unchanged. To deal with this problem, the adjacency matrix is generated and the clusters with number of pixels under a threshold are reassigned to the nearest neighbor cluster using connected component analysis. Figure 2.5 shows the results of 3D SLIC algorithm after the connected component analysis.

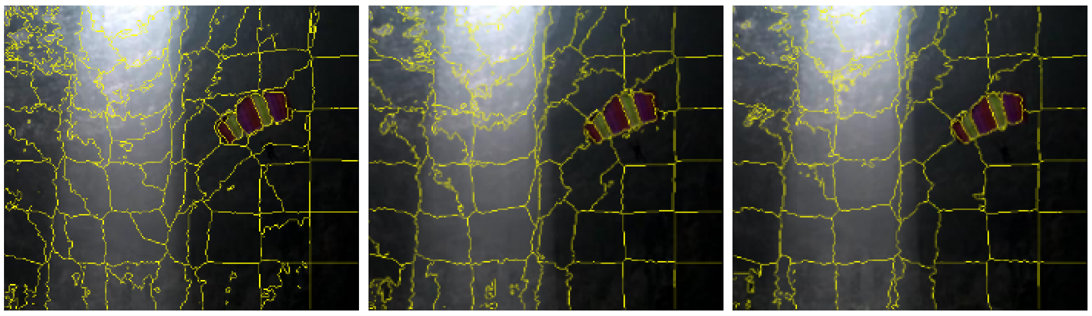


Figure 2.5: Results of 3D SLIC voxel grouping on three consecutive frames. The boundaries of each supervoxel are shown in yellow. The block enclosed by the yellow boundaries in the corresponding position between frames has the same label.

Note that for some HD videos that contain too much redundant details on the background, the SLIC pixel grouping generates some tiny clusters which are too fine and increase the computation and processing time. To solve this problem, it is recommended to cluster videos of this kind after the bilateral filtering. The fine edges can be removed and the main boundaries of the object and background would be retained.

2.2.4 Coarse Segmentation

For each supervoxel, the coarse segmentation is performed by combining the SSC approach on tracking points. As shown in Figure 2.6(a), the SSC algorithm provides an approximate region containing the object of interest. Based on that, a strategy is developed with the following rules: for each supervoxel in the video clip as shown in Figure 2.6(b), if all the tracking points in it are marked red, this supervoxel is considered as background (black region in Figure 2.6(c)); similarly, if all the tracking points in a supervoxel are marked yellow, this supervoxel is labelled as foreground (white region in Figure 2.6(c)); otherwise, for the supervoxels containing both colored markers, they are considered as undetermined regions, as shown by the gray region in Figure 2.6(c).

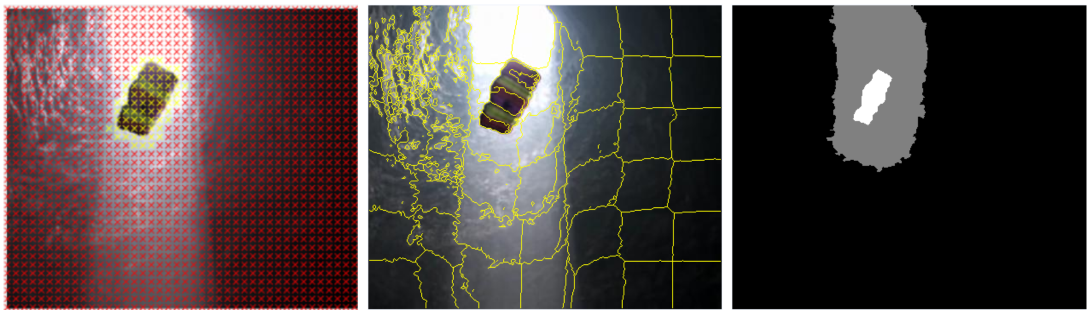

(a) (b) (c)

Figure 2.6: Coarse segmentation by combining the results of SSC and 3D SLIC algorithms. (a) Tracking points generated by KLT and SSC. The yellow and red markers represent the foreground and background region respectively; (b) The 3D SLIC supervoxels on the same frame; and (c) The mask generated by combining (a) and (b). The black, gray and white regions denote determined background, undetermined region and determined foreground respectively.

2.2.5 Graph-based Fine Segmentation

For fine segmentation, the GrabCut [70] algorithm is adopted since it requires a set of pixels for background, *i.e.*, it allows incomplete labeling. Also, GrabCut looks for the minimum iteratively rather than in an one-time manner. Each iteration improves the parameters of the Gaussian Mixture Models (GMMs) to generate a better segmentation.

For the video frames in RGB color space, the object and background are modeled by a full-covariance Gaussian mixture with K components (typically $K = 5$). In order to deal with the GMM tractably, in the optimization framework, an additional vector $\boldsymbol{k} = [k_1, k_2, \cdots, k_n, \cdots, k_N]$ is introduced, with $k_n \in 1, \cdots, K$, assigning, to each pixel, a unique GMM component, with one component either from the background or the foreground model. Using the mask generated by the coarse segmentation, the black, white and gray regions are flagged with background, foreground and undetermined, or simply marked as 0, 1 or 2 for the image. Applying k-means clustering, the pixels belonging to either object or background are clustered into K groups (GMMs). The mean and covariance of the GMM can be estimated by the RGB values of pixels in each cluster, and the weight can be determined by the ratio of the number of pixels in the cluster to the number of overall pixels. Finally, texture (color) and boundary (contrast) information is used to improve the GMM and get a reliable segmentation result within a few iterations, as illustrated in Figure 2.7.

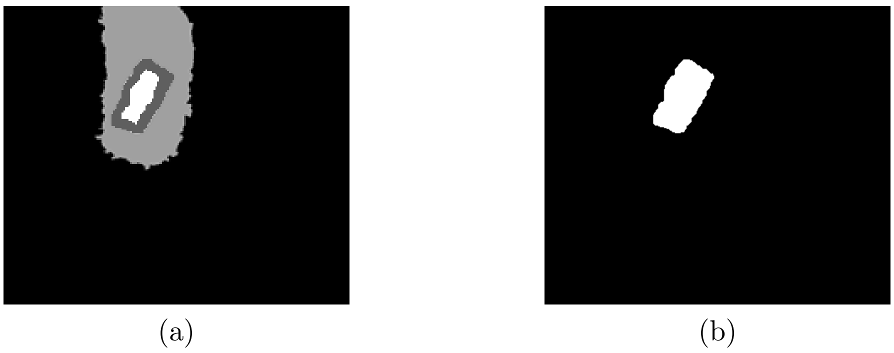


Figure 2.7: Result of fine segmentation using GrabCut method. (a) The algorithm segment the undetermined region to light and dark gray regions; and (b) The light and dark gray regions are merged to the background and foreground respectively to form the final mask.

Admittedly, GrabCut algorithm has its drawbacks: if the background is too complicated and contains too much fine detail, or there is a lack of similarity between the background and foreground, the segmentation results would be affected. In addition, the speed of computation cannot be guaranteed. However, compared to traditional Graph Cut, GrabCut improves the foreground and background model as GMM of RGB channel, instead of using gray scale histograms. Moreover, GrabCut learns the parameters and segments the video frames iteratively, whereas traditional Graph Cut is a one-time segmentation. Most importantly, Graph Cut needs some user-defined "seeds" for both foreground and background, whereas GrabCut only requires a set of pixels for background, *i.e.*, it allows incomplete labeling.

2.3 Experimental Results

The experiments are conducted on a variety of video content. The developed algorithm is applied on multiple types of data, and generate a mask of the extracted object for each frame. The comparison of the segmentation results to those produced by other state-of-the-art methods [19, 20, 21, 22, 23] is made in this section. Both qualitative and quantitative results will be presented to support the effectiveness and robustness of the developed method. Some demonstrations of interesting and pleasing effects generated based on the developed algorithm will be shown as well.

2.3.1 Parameters Settings

The parameters used in the experiments are as follows. In the point tracking and clustering process, the initial point sampling interval is set to 10 pixels and the tracking points are reset every 5 frames. The number of clustering groups depends on the application. Typically, it is set to 5. To group pixels, the 3D SLIC algorithm is performed every 30 frames (clip size). For demonstration, the desired number of supervoxels in one frame is set to 100; the desired depth of supervoxels is $D = 5$ frames; the regularity $m = 22$; depth of supervoxel $D = 5$; the weights for temporal distance and L^* channel are $w_Z = 50$ and $w_L = 1$ respectively. On average, the 3D SLIC algorithm runs 5 iterations to get a reliable result. To construct the visual effects, the brightness, size, transparency and location of the extracted object and the background image/video could be adjusted and controlled by the user control.

2.3.2 Evaluation on SegTrack Dataset

First the video sequences from the SegTrack [71] dataset are considered since a pixel-level segmentation ground truth for each video is available. The ground truth provided with the original data is used to quantitatively evaluate the segmentation performance. The developed method is compared with five state-of-the-art methods as shown in Table 2.1. The "penguin" video sequence is not available for this segmentation application since the ground truth for the "penguin" sequence is designed for object tracking in a weakly supervised setting, in which only one penguin is manually annotated by the original user at the each frame. Note that this method is an unsupervised methods, whereas [72] and [71] are supervised method which needs an initial annotation for the first frame. One can see that this algorithm outperforms the other unsupervised methods except for the "parachute" and "birdfall2" video where it is still comparable to the best one. As mentioned before, for the "parachute" video sequence, this result is based on the fact that the person under the parachute should be a part of the object and extracted. However, the person in the ground truth of "parachute" sequence was removed in the original dataset, which leads to a slightly inaccurate error calculation. Due to the small size of moving object and the complex background in the scene, the predefined density of tracking points may not be high enough to extract the foreground in "birdfall2" video sequence, which leads to the pixel error a little higher than the best one. However, this can be improved by making the density of the tracking points self-adjustable. The results in Table 2.1 take the average of the difference between pixel error and the ground truth, *i.e.*,

$$\text{error} = \frac{xor(\text{result, ground truth})}{\text{number of frames}} \tag{2.31}$$

where xor is an exclusive OR operation.

Figure 2.8 shows an example of the qualitative results of parachute video sequence in SegTrack dataset. In this video sequence, the foreground and background regions move in different ways. Compared to the last column of Figure 2.8(b) and (c), the person

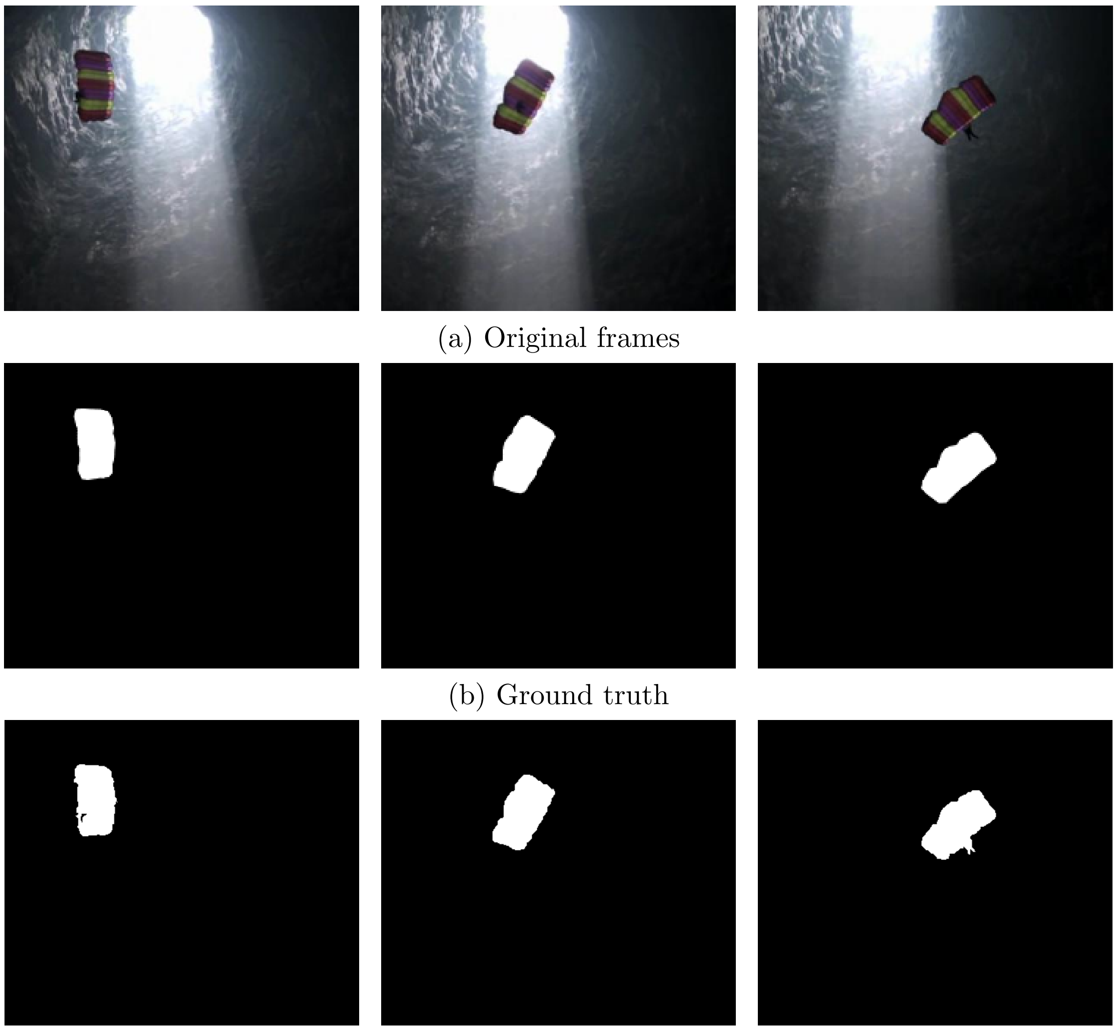
Table 2.1: Quantitative pixel-level errors and comparison with the state-of-the-art methods on SegTrack dataset.

Video Sequence Name	[57]	[72]	[73]	[74]	[71]	Ours
parachute	220	502	201	221	235	219
girl	1488	1755	1785	1698	1304	1471
monkeydog	365	683	521	472	563	345
birdfall2	155	454	288	189	252	232
cheetah	633	1217	905	806	1142	621
penguin*	NA	NA	NA	NA	NA	NA

* The video sequence "penguin" is not applicable to this evaluation.

under the parachute is segmented into the foreground in this results instead of merged into background as shown in ground truth. This makes the segmentation result more reasonable, although it leads to the error increase in Table 2.1. Figure 2.9 compares this results with the ground truth on "girl" video sequence. In this video sequence, the foreground is always at the center of the frame, the background keeps moving from left to right. Also, this video suffers from low resolution and severe motion blur which increases the difficulty. The point tracking and supervoxel generation are affected by the motion blur (the third column of Figure 2.9). This becomes the main source of the pixel-level error. In addition, it has been proved that the developed algorithm performs well on rigid body motion, however, the motion in this video sequence is non-rigid. Specifically, different parts of the body behave differently during the movement. This makes motion clustering module fail to work properly. In order to solve this problem, each part of the moving object, *i.e.*, body, arms, legs, can be considered as a rigid-body, the SSC algorithm clusters them into different classes and they are combined afterwards.

With the segmentation results, some interesting and pleasing visual affect can be created easily for consumers, as shown in Figure 2.10 and Figure 2.11. In these demonstrations, the extracted moving object in the video sequence can be adjusted in terms of size, location, brightness, transparency, etc. according to user's demand. Also, the



(a) Original frames

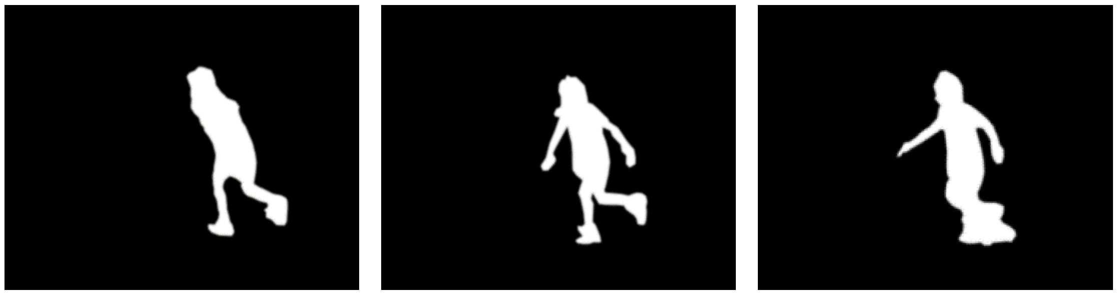
(b) Ground truth

(c) Our results

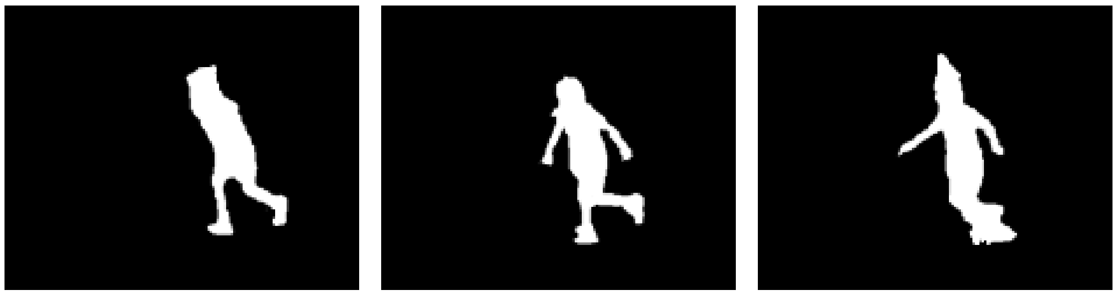
Figure 2.8: Qualitative results of SegTrack "parachute" video sequence.

number of extracted objects and the attributes mentioned above can be adjusted individually. The objects from different frames can be synthesized into a single image to create a static effect, or a moving background to create a dynamic effect. To achieve the results in 2.10 and Figure 2.11, the extracted objects from the original video sequences and the background images/videos are converted into CIELAB color space, and then histogram matching is performed on L^* channel only to make the inserted objects look

(a) Original frames



(b) Ground truth



(c) Our results

Figure 2.9: Qualitative results of SegTrack "girl" video sequence.

natural. This results in a good visual effects by comparing the brightness of the objects in Figure 2.10(b) and Figure 2.10(d)(e)(f), Figure 2.11(b) and Figure 2.11(d)(e)(f). Note that all these effects can be easily produced by the predefined user parameters.

All the experiments are performed on an Intel® Core™i5-4590 CPU at 3.30GHz with 16GB memory. Before extensive code and data structure optimization, the processing time per frame is around 0.52s, 15.86s, and 7.62s for points clustering, supervoxel generation and final segmentation, respectively.

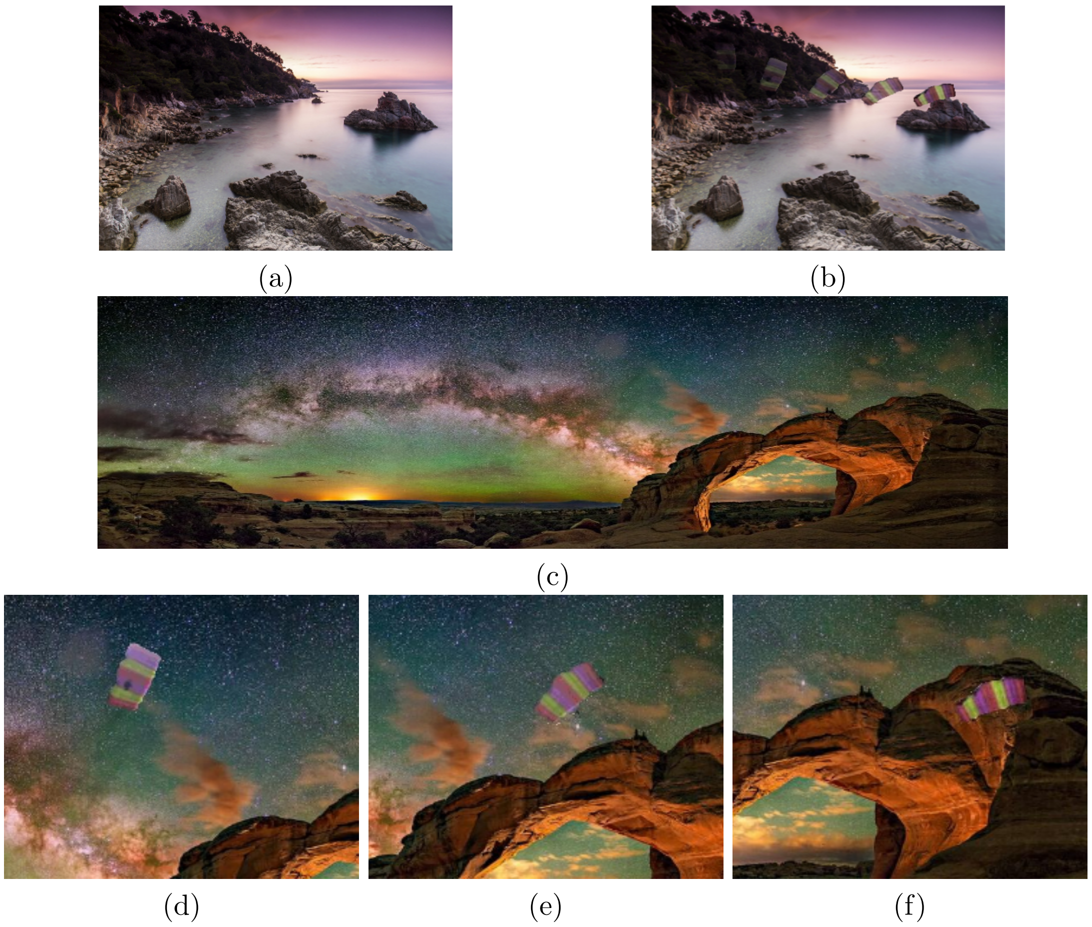


Figure 2.10: Visual effects created using segmentation result of parachute video sequence. (a) The background used to create static effect in (b); (b) A static image synthesized by the segmentation result and (a); (c) A panorama used to create dynamic effects as shown in (d) to (f); and (d) to (f) The video frames generated by synthesizing the segmentation result and (c).

2.3.3 Evaluation on Kodak Alaris Consumer Video Dataset

With the rapid development and lower cost of smartphones and new digital capture devices, consumer videos are becoming ever popular as is evident by the large volume of YouTube video upload, as well as video viewing in the Facebook social network. These large amount of videos also pose a challenge for organizing and retrieving videos for consumers. In addition to the SegTrack dataset, evaluations of the developed approach

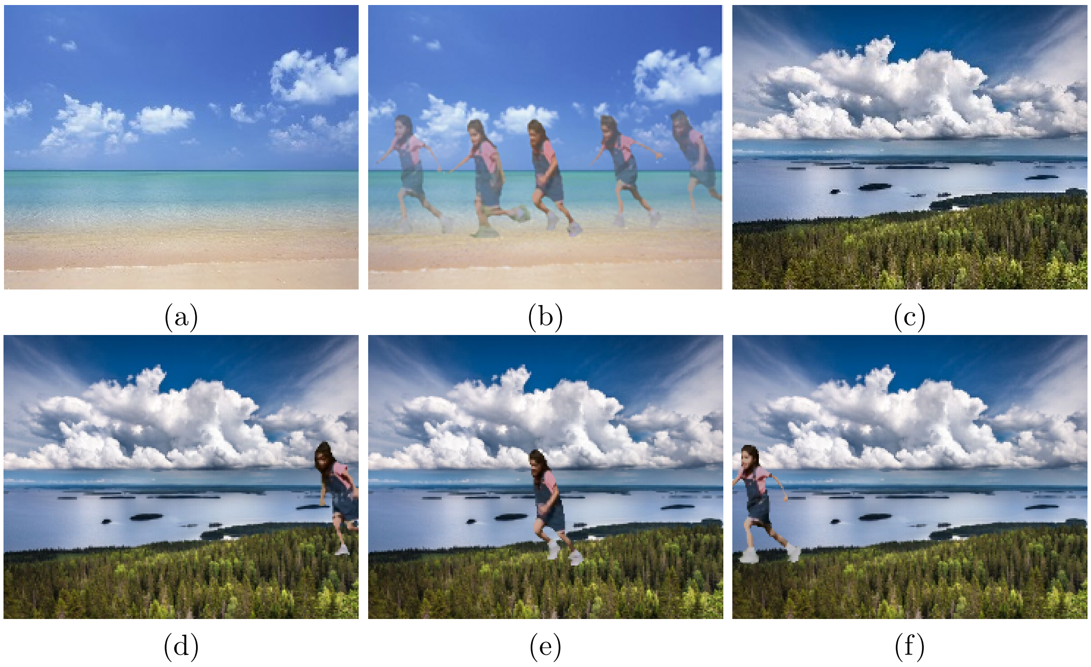
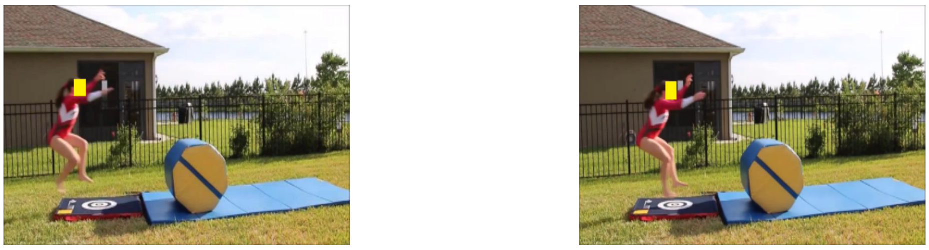


Figure 2.11: Visual effects created using segmentation result of girl video sequence. (a) The background used to create static effect in (b); (b) A static image synthesized by the segmentation result and (a); (c) The background used to create dynamic effect in (d) to (f); and (d) to (f) The video frames generated by synthesizing the segmentation result and (c).

are conducted on some of the videos from Kodak Alaris consumer video dataset. The videos in the dataset are mostly captured in standard HD format with high frame resolution. Figure 2.12 shows the qualitative results of "gymnast1" video sequence in this dataset. Because of the high resolution of the video, bilateral filtering is applied on the original frame to remove some fine details of the background and keep the main edges. The bilateral filtering does not affect the performance of either SSC or 3D SLIC algorithm, but rather saves the computation. Another example is shown in Figure 2.13. In this video, some parts of the moving object (dog) is similar to the background trees in color, and the other parts are as white as the background sky. It turns out that this algorithm produces reasonably good results for this difficult task.


(a) Original frames in the video sequence.

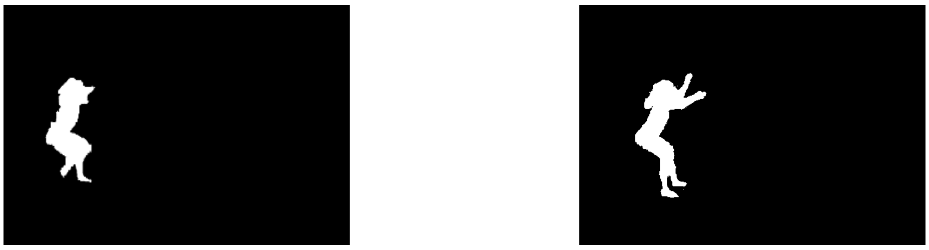
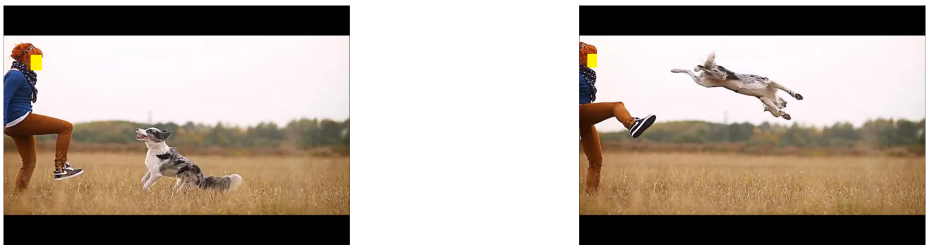

(b) Mask representing the extracted object in the sequence.

Figure 2.12: Object segmentation results on "gymnast1" video sequence in Kodak Alaris consumer video dataset.


(a) Original frames in the video sequence.

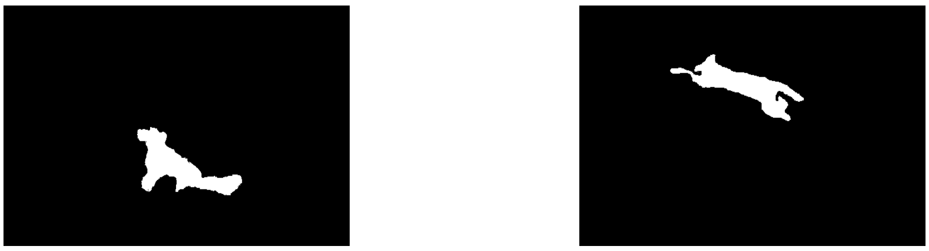

(b) Mask representing the extracted object in the sequence.

Figure 2.13: Object segmentation results on "dog" video sequence in Kodak Alaris consumer video dataset.

2.4 Discussion

This chapter has developed a novel and accurate coarse-to-fine approach to segment the salient object in video sequences and generate interesting and pleasing visual outputs. This approach involves a parallel scheme which consists of KLT, SSC and 3D SLIC algorithms to identify the approximate location of the most salient object and, as a result, an unsupervised graph-based method can be used in subsequent steps.

Admittedly, there exists some drawbacks in the developed scheme. For example, the SSC algorithm divides the motions into different groups according to the point tracking results, but sometimes the object with non-rigid motion leads to inconsistent motion on different part of the object. This results in the failure of segmentation. To solve this problem, an easy way is to consider each part on the object as an independent component, and then separate these components from the background. Finally these components can be combined using image classification approaches such as described in [75] and [76]. In addition, motion blur decreases the chance of success of object detection and makes the created effects not looking natural. To fix this problem, Kim and Lee [77] proposed an algorithm that deals with general blurs inherent in dynamic scenes. Also, some consumer videos are taken by the hand-held device, Su and Heidrich [78] developed an approach to handle rolling shutter wobble. The intra-frame deblurring algorithm proposed by Zhang and Yang [79] is also a good choice in terms of reducing motion blur.

However, since the coarse segmentation determines the location of the moving object rather than the exact boundaries, the accuracy of KLT, SSC and 3D SLIC does not affect the final segmentation result too much, i.e., the robustness of this approach can be guaranteed. It is also worth mentioning that this algorithm can be easily extended to multiple objects segmentation by controlling the number of classes in the SSC stage. Due to the novel approach involving motion tracking and clustering, the objects with unconnected bodies and objects with transparent holes can be extracted as well. Compared to the previous work [8], it has stronger ability to segment video sequences accurately in any resolution and any length within a shorter time. The experimental results validate the effectiveness and performance of the developed method.

Chapter 3

Batch-Normalized Recurrent Highway Networks

Deep artificial neural networks have been applied to various computer vision and pattern recognition tasks, including video analysis. The adjacent frames in a video sequence are related by consistent motion and pixel similarity in terms of pixel color and location. Therefore, video sequences can be modeled by Recurrent Neural Networks (RNNs). Generally, greater depth of the networks leads to the small error, but training such a deep networks is challenging due to the fact that the distribution of each layer's inputs changes during training. Moreover, it turns out that the gradient in deep neural networks is unstable, tending to either explode or vanish in earlier layers. In such deep architectures the vanishing or exploding gradient problem becomes a key issue.

Several techniques have been proposed to circumvent the vanishing and exploding gradient problem. For example, batch normalization [14], which addresses the internal covariate shift problem by normalizing the layer inputs per mini-batch statistics. While batch normalization has been found to be very effective for feedforward CNNs, the technique has not been as prevalent on RNNs.

In this chapter, Batch-Normalized Recurrent Highway Networks (BNRHN) are developed to control the gradient flow in an improved way for network convergence in RNNs. The framework will be performed using batch normalization on recurrent highway layers. This architecture will reasonably control the gradient flow and effectively strengthen the ability of processing the data as sequences by increasing the depth of the network. The developed model is tested on an image captioning task using MSCOCO dataset. Experimental results indicate that the batch normalized recurrent highway networks converge faster and perform better compared with the traditional LSTM and RHN based models.

This chapter is organized as follows: Section 3.1 reviews some related techniques on gradient control in deep RNNs. Section 3.2 presents the developed sequence modeling framework and the details of the key component of the framework. Section 3.3 discusses the experimental setup, performance evaluations, and experimental results. Finally, some concluding remarks are presented in Section 3.4.

3.1 Related Work

Much theoretical and empirical evidence indicates that the depth of neural networks plays an important role as a powerful machine learning paradigm. Deep CNNs have been proven successful on modern computer vision tasks, as illustrated in [10]. However, increasing depth in RNNs which are deep in the time domain typically does not take advantage of depth since the state update modeled by certain internal function mapping in modern RNNs is usually represented by non-linear activations [80]. Also, increasing depth in time tends to make RNNs suffer from vanishing or exploding gradient problems.

Recently, researchers made great efforts on gradient control. The highway layers [12], based on the LSTM unit, relax the limitation of training deep RNNs. Specifically, a highway network additionally defines two nonlinear transforms- the transform gate and carry gate. These gates express how much of the output is produced by transforming the input and carrying it, respectively. By coupling the transform gate and carrying gate, a highway layer can smoothly vary its behavior between that of a plain layer and that of a layer which simply passes its inputs through. Due to this gating mechanism, a neural network can have paths along which information can flow across several layers without attenuation. Thus, highway networks, even with hundreds of layers, can be trained directly using stochastic gradient descent. These networks, when used with a variety of activation functions, have been shown to avoid the vanishing or exploding gradient problem. Highway layers have achieved success in the fields of speech recognition [81] and language modeling [82].

Based on the insights of highway layers, Zilly *et al.* [13] introduced Recurrent Highway Networks (RHNs) that have long credit assignment paths, not just in time, but also long in space (per time step). By replacing the LSTM cell in the recurrent loop, the RHN layer instead stacks the highway layers inside the recurrent units. By increasing recurrence depth, additional non-linearity strengthens the ability of the recurrent network without slowing down the processing. Compared to regular RNNs, RHNs provide more versatile ways to deal with data flow in terms of transforming and carrying information. It has been theoretically proven that coupling a carrying and transforming gate effectively controls the gradient. However, such a constraint may limit the power of the network to some extent. In the next sections, the focus will be on this problem and a new scheme is developed which relaxes the constraint in RHNs, by incorporating batch normalization. The developed method simultaneously improves network performance while avoiding the vanishing and exploding gradient problem.

3.2 Developed Framework

A plain RNN consists of L layers and T time states. Denoting the input sequence as $\{\mathbf{x}^1, ..., \mathbf{x}^{t-1}, \mathbf{x}^t, \mathbf{x}^{t+1}, ..., \mathbf{x}^T\}$, in general, each node in the layer $l \in \{1, 2, ..., L\}$ and time state $t \in \{1, 2, ..., T\}$ takes input $\mathbf{x}_l^t$ and output $\mathbf{h}_l^t$, respectively, with a non-linear transformation H. Omitting the bias term for simplicity, the output can be represented as

$$\mathbf{h} = H(\mathbf{x}, \mathbf{W}_H) \tag{3.1}$$

where the non-linear activation H is typically specified by hyperbolic tangent function, $tanh$, and W_H is the associated weight matrix. In highway networks [12], the training process is facilitated by using adaptive computation. The additional defined transform gate and carry gate determine how much information is transformed and carried to the output, *i.e.*,

$$\mathbf{t} = T(\mathbf{x}, \mathbf{W}_T) \tag{3.2}$$

$$\mathbf{c} = C(\mathbf{x}, \mathbf{W}_C) \tag{3.3}$$

where t, c are the output of the transform and carry gate, respectively, T, C are defined as a sigmoid function $\sigma(x) = 1/(1+e^{-x})$, and $\mathbf{W}_T, \mathbf{W}_C$ are corresponding weights. The RHN layer with recurrence depth D is defined as

$$\mathbf{s}_d^t = \mathbf{h}_d^t \odot \mathbf{t}_d^t + \mathbf{s}_{d-1}^t \odot \mathbf{c}_d^t \tag{3.4}$$

where $\odot$ implies the element-wise product.

RHN uses highway layers instead of LSTM units in regular recurrent networks as shown in the dotted box in Figure 3.1. Note that each recurrent loop takes the output of the last recurrent unit in the previous loop ($\mathbf{s}_{D-1}^t$) as input, and the time-varying data $\mathbf{x}^t$ is only fed into the recurrent loop to the recurrence depth, $d = 1$. According to Geršgorin circle theorem [83], all eigenvalues of the temporal Jacobian are preferably set to 1 across time steps in order to keep the gradient flow steady. In this case, the Geršgorin circle radius is reduced to 0 and each diagonal entry of temporal Jacobian is set to 1. Zilly *et al.* [13] states that it can be accomplished by coupling the carry gate to the transform gate by setting $C = 1 - T$, as a constraint, in order to prevent an unbounded "blow-up" of state values which leads to more stable training. However,

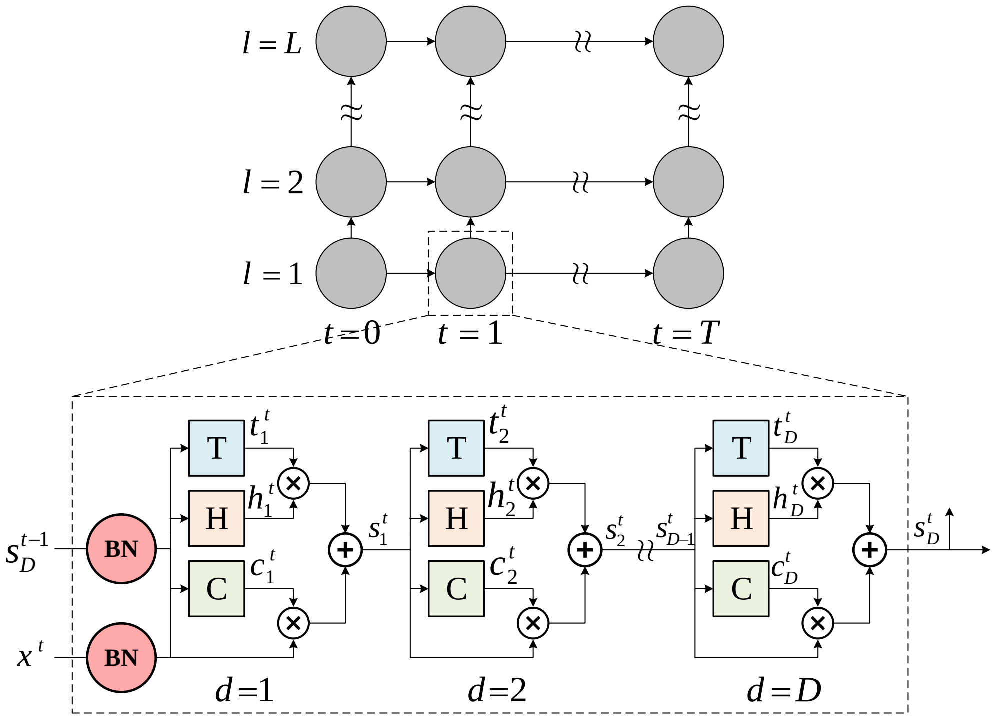


Figure 3.1: The architecture of batch normalized recurrent neural networks. T and C denote transform and carry gates specified in (3.2) and (3.3) respectively, H is a nonlinear transform specified by (3.1), and BN represents batch normalization operation.

this constraint may limit the ability of the gates to freely learn parameter values and imposes a modeling bias which may be suboptimal for certain tasks [84, 85].

Figure 3.1 shows the layout of the developed recurrent network architecture. Because of its ability to control the gradient during back propagation, batch normalization is incorporated to the inputs of each recurrent loop. This allows us to relax the $C = 1 - T$ constraint, while simultaneously making gradients less prone to vanishing or exploding. Specifically, in batch normalization, the mean and variance are extracted across each channel and spatial locations. Each individual in the batch is normalized by subtracting the mean value and dividing by variance, and the data are recovered by shifting and scaling the normalized value during training.

3.3 Experimental Results

Experiments are performed on an image captioning task. The first part of this section describes the implementation details and experimental setup. Next, the evaluation and analysis of the developed framework are discussed from different perspectives.

3.3.1 Experimental Setup

Dataset The evaluation is carried out on the popular MSCOCO captioning dataset [32]. This dataset contains ∼80k training images, ∼40k validation images and ∼40k test images. Note that ground truth captions are only available for training and validation sets. In order to efficiently use the available data, the validation set is split into three parts: 85% of the images are merged into the training set, 10% are used for testing, and the remaining 5% are used as a validation set for hyperparamter tuning. All the experimental results are evaluated using the MSCOCO caption evaluation server [86].

Metrics The following metrics are employed for evaluation: 1) *BLEU* [87] is a metric for precision of word n-grams between predicted and ground truth sentences; 2) *ROUGE-L* [88] takes into account sentence level structure similarity naturally and identifies the longest co-occurring sequence in n-grams automatically; 3) *METEOR* [31] was designed to fix some of the problems found in the more popular BLEU metric, and also produce good correlation with human judgment at the sentence or segment level. It has several features not found in other metrics, such as stemming and synonymy matching, along with the standard exact word matching; and 4) *CIDEr* [89] computes the average cosine similarity between n-grams found in the generated caption and those found in reference sentences, weighting them using TF-IDF. METEOR is more semantically preferred than BLEU and ROUGE-L [90].

Training Details In the training phase, the <START> token is added at the beginning of the sentence and the <END> token is attached at the end of the sentence so that the model can generate captions of varying lengths. In inference mode, the caption generation is started with <START> and the word combination with highest probability will be selected. In practice, beam search was used to prune the word combinations with small joint probabilities to avoid the number of considered word combinations growing exponentially. Specifically, in the first step, the fist three candidates with highest probability were selected. Moving to the second step, three more candidates were selected based upon each of the selected candidates in the previous step. This process was stopped until the network output the <END> token. The combinations with highest joint probability was considered as the final output sentence. The word embedding size and number of RHN neurons per layer are empirically set to 512. Based on empirical results, the recurrence depth $D = 3$ is adopted. Stochastic gradient descent is employed for optimization, where the initial learning rate and decay factor are set to 0.1 and 0.5, respectively, and the learning rate decays exponentially every 8 epochs. The initial time state vector is extracted from the Inception_v3 model [91] and all the other weight matrices are initialized with a random uniform distribution. The training process minimizes a softmax loss function. The developed network is implemented using TensorFlow [92] and trained on a server with dual GeForce GTX 1080 graphics cards.

Table 3.1: Evaluation metrics on MSCOCO dataset. LSTM: regular RNN model with LSTM cell; RHN: model with original RHN cell; BN_RHN: the developed model with RHN constrain relaxed and batch normalization applied instead.

Model	LSTM	RHN	BN_RHN
BLEU-1	0.706	0.688	0.710
BLEU-2	0.533	0.512	0.541
BLEU-3	0.397	0.377	0.408
BLEU-4	0.298	0.281	0.311
ROUGE-L	0.524	0.511	0.533
METEOR	0.248	0.241	0.254
CIDEr	0.917	0.864	0.955

3.3.2 Image Captioning Results

The developed model is evaluated on MSCOCO image captioning dataset. The results are reported in Table 3.1. To make a fair comparison, an image feature vector is extracted as initialization of the hidden state using the same Inception_v3 model [91], and the parameters are locked in it (without fine-tuning) in all test models. Three test models are compared: LSTM denotes the im2txt model using regular LSTM cells implemented by [93]; RHN denotes the image captioning generation performed by original RHNs [13]; and BNRHN is the developed method with batch normalization instead of the $C = 1 - T$ constraint in RHN cell. The results show that the BNRHN is the best performing model. METEOR and CIDEr are generally considered the most robust scores for captioning. The higher BLEU-4 and METEOR scores, due to fluency of language in the image captions, can be attributed to the RHN depth, because more depth increases the complexity that helps learn the grammatical rules and language semantics. The LSTM employs a mechanism with input, output, and forget gates to generate complex captions. The developed model shows better performance than LSTM, which may indicate that simplifying the gate mechanism and increasing depth do not affect performance for image captioning. The test model with RHN cells benefits from having less parameters during training, and good gradient control, in a simple way. The BNRHN achieves better result than original RHN, because the gate value model biases are more flexible, and batch normalization guarantees the steady gradient flow in back propagation.

Additionally the developed model is compared based on the speed of convergence. Figure 3.2 shows the loss change during training. The BNRHN model achieves the steady loss fastest among all three models. It turns out that adding batch normalization allows a more aggressive learning rate and achieves faster convergence. It is worth mentioning that during back propagation in the original LSTM and RHN models, a gradient norm clipping strategy is adopted to deal with exploding gradients and a soft constraint for the vanishing gradients problem to generate reasonable captions. For

BNRHN, this restriction can be relaxed. This confirms that the developed model is effective on gradient control, as presupposed in Section 3.2.

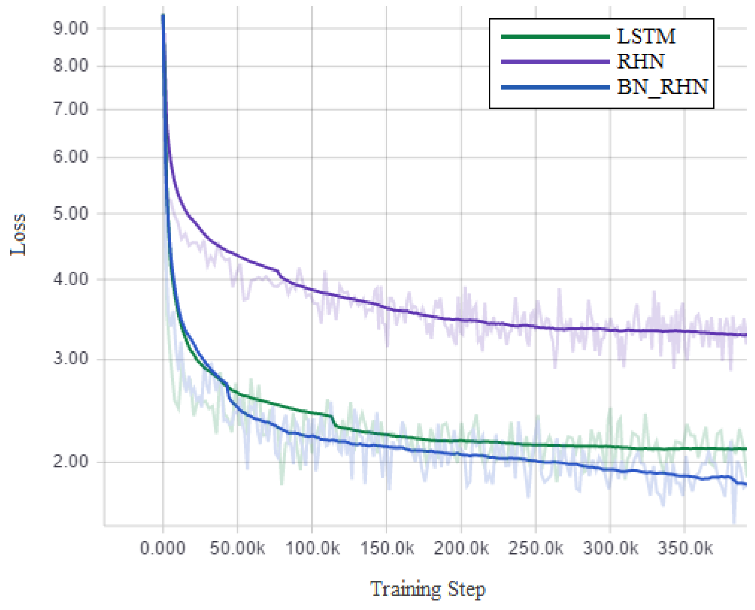
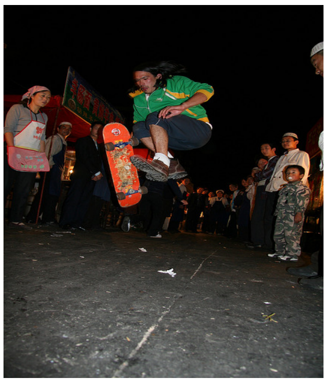
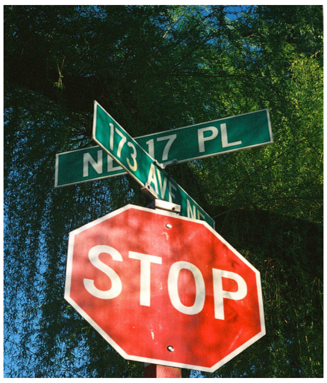


Figure 3.2: The total loss change vs. training steps. All dark curves are smoothed by a factor of 0.8. The light curves are not smoothed.

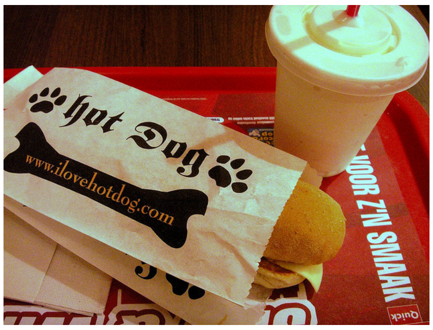
Figure 3.3 lists a few examples of the descriptions generated by the developed model, compared with the captions obtained by the original LSTM and RHN model. The sample images are picked randomly. It is clear that the overall quality of the captions generated by the developed model have improved significantly compared to RHN model. Notice that the BNRHN model describes the object in the image accurately and can generate better descriptions of the image, even for very complex images, such as middle left and middle right in Figure 3.4. Additionally, the captions generated by the developed model have better grammar and language semantics due to the increased depth of recurrent network. Figure 3.4 shows more examples including negative results.



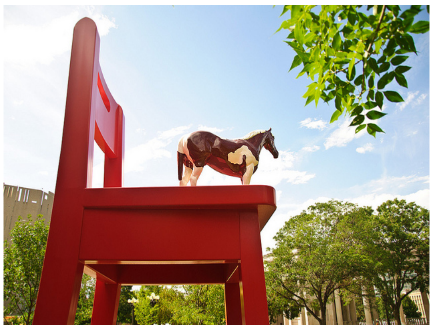
(**LSTM**) a group of people standing around a parking meter .
(**RHN**) a group of people standing next to each other .
(**BNRHN**) a young man riding a skateboard down a street .
(**G.T.**) a person is doing a trick on a skateboard



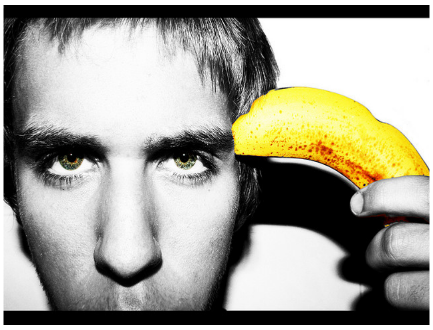
(**LSTM**) a red stop sign sitting on top of a metal pole .
(**RHN**) a red stop sign sitting on the side of a road .
(**BNRHN**) a stop sign with a street sign attached to it .
(**G.T.**) Street corner signs above a red stop sign.



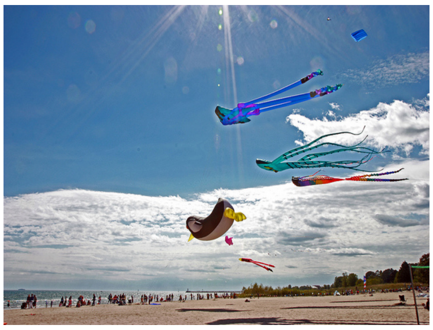
(**LSTM**) a box with a donut and a cup of coffee .
(**RHN**) a birthday cake with a picture of a dog on it .
(**BNRHN**) a plate with a doughnut and a cup of coffee .
(**G.T.**) A bag with a hot dog inside of it.



(**LSTM**) a large brown dog sitting on top of a wooden bench .
(**RHN**) a statue of a cow with a bird on top of it .
(**BNRHN**) a statue of a cow standing on top of a wooden bench .
(**G.T.**) A giant chair with a horse statue on it

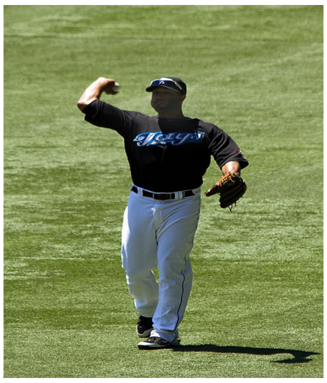


(**LSTM**) a man holding a banana in his hand .
(**RHN**) a woman is holding a banana in her hand .
(**BNRHN**) a man holding a banana up to his ear .
(**G.T.**) A man is holding a banana up to his temple.

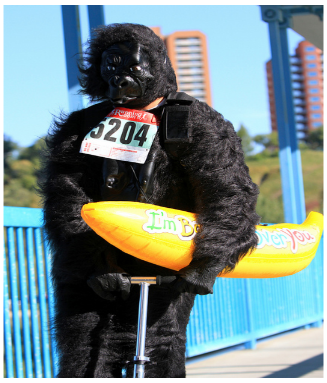


(**LSTM**) a group of people flying kites in the sky .
(**RHN**) a group of people standing on top of a sandy beach .
(**BNRHN**) a group of people flying kites on a beach .
(**G.T.**) A group of people standing on a bitch flying large kites.

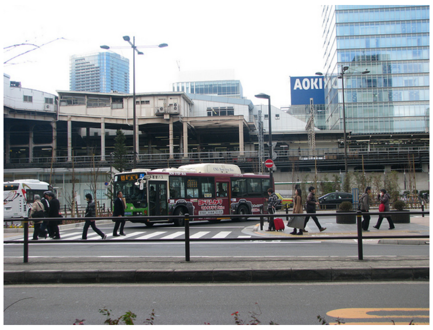
Figure 3.3: Example results on MSCOCO captioning dataset.



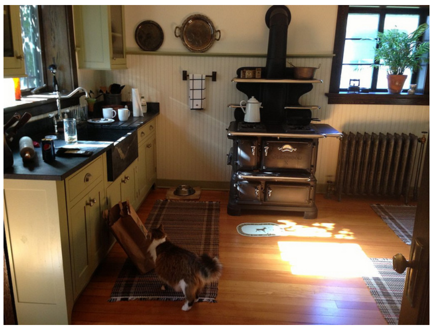
(**LSTM**) a baseball player standing on top of a field .
(**RHN**) a baseball player standing on a field holding a bat .
(**BNRHN**) a man in a baseball uniform throwing a ball .
(**G.T.**) A man in a black shirt is throwing a baseball.



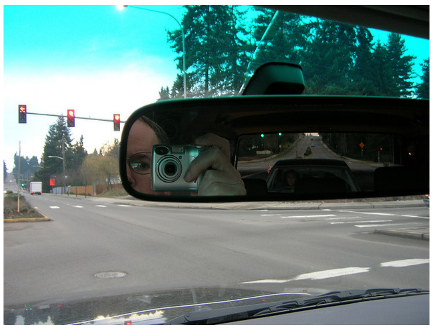
(**LSTM**) a teddy bear sitting on top of a skateboard .
(**RHN**) a man holding a giant pair of scissors .
(**BNRHN**) a person wearing a hat and holding a banana .
(**G.T.**) A man holding a giant banana while riding a scooter.



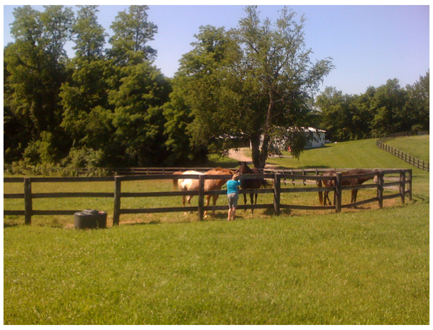
(**LSTM**) a bus driving down a street next to a tall building .
(**RHN**) a group of people riding bikes down a street .
(**BNRHN**) a city street filled with lots of traffic .
(**G.T.**) A group of people walking down a sidewalk near a bus.



(**LSTM**) a cat sitting on a chair in a kitchen .
(**RHN**) a cat sitting on a chair in a room .
(**BNRHN**) a black and white dog standing in a kitchen .
(**G.T.**) A puppy is looking at a paper bag in the kitchen.



(**LSTM**) a rear view mirror of a car in the side view mirror .
(**RHN**) a rear view mirror on the side of a car .
(**BNRHN**) a rear view mirror with a dog in the side mirror .
(**G.T.**) A guy takes a picture of his car's rear view mirror.



(**LSTM**) a person sitting on a bench in a park .
(**RHN**) a wooden bench sitting on top of a lush green field .
(**BNRHN**) a person sitting on a bench in a park .
(**G.T.**) A woman standing next to a group of horses on a field.

Figure 3.4: More results on MSCOCO captioning dataset. The bottom two are negative examples.

3.4 Discussion

This chapter introduces a novel recurrent neural network model that is based on batch normalization and recurrent highway networks. The analyses provide insight into the ability of the batch normalized recurrent highway model to dynamically control the gradient flow across time steps. Additionally, this model takes advantages of faster convergence compared to the original RHN, and keeps the feature of increasing depth in the recurrent transitions while retaining the ease of training. These theoretical advantages were supported by the results and analysis. Experimental results on image captioning task reveals that the developed model achieves high METEOR and BLEU scores compared to previous models on a modern dataset. Since this work is able to deal with applications with RNN structure, such as captioning, it turns out that this technique can be used in Multi-Modal Vector Representation (MMVR) as we will see in Chapter 5.

Chapter 4

Semantic Sentence Embeddings

We have already seen the success in image-to-text conversion using developed BNRHN. If we look at the image-to-text model closely as shown in Figure 4.1, one can see that the hidden state between the feature extractor (CNN) and the captioner (RNN) is represented by a vector h. This vector encodes the image and can be decoded in to a sentence as caption. Intuitively, other types of multimedia can also be encoded as a vector in similar ways, for example, machine translation uses a sequence-to-sequence model to encode a sentence into a vector, then the vector is decoded as another language.

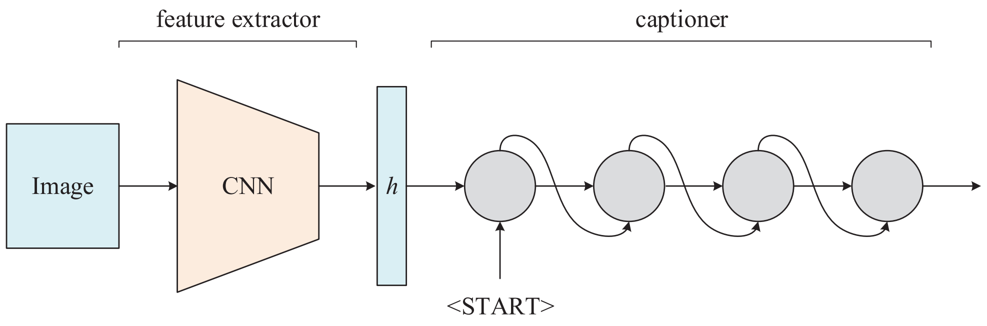


Figure 4.1: The typical image-to-text (image captioning) inference model. The CNN encodes the image into a feature vector h, which is decoded by the following RNN.

To further explore the embedding, this chapter introduces a sentence to vector encoding framework suitable for advanced natural language processing. The latent representation is shown to encode sentences with common semantic information with similar vector representations. The vector representation is extracted from an encoder-decoder model which is trained on sentence paraphrase pairs[1]. This chapter demonstrates the application of the sentence representations for two different tasks – sentence paraphrasing and paragraph summarization, making it attractive for commonly used recurrent frameworks that process text. Experimental results help gain insight into how vector representations are suitable for advanced language embedding.

The rest of this section is organized as follows: Section 4.1 reviews some related techniques. Section 4.2 presents the developed encoder-decoder framework for sentence and paragraph paraphrasing. Section 4.3 discusses the experimental results. Concluding remarks are presented in Section 4.4.

4.1 Related Work

Most machine learning algorithms require inputs to be represented by fixed-length feature vectors. This is a challenging task when the inputs are text sentences and paragraphs. Many studies have addressed this problem in both supervised and unsupervised approaches. For example, [30] presented a sentence vector representation while [27] created a paragraph vector representation. An application of such representations is shown by [94], that has used individual sentence embeddings from a paragraph to search for relevant video segments.

An alternate approach uses an encoder-decoder [95] framework that first encodes f inputs, one at a time to the first layer of a two layer Long Short-Term Memory (LSTM), where f can be of variable length. Such an approach is shown for video captioning tasks by S2VT [96] that encodes the entire video, then decodes one word at a time.

There are numerous recent works on generating long textual paragraph summaries from videos. For example, [97] present a hierarchical recurrent network that comprise of

[1]For example, the sentence "the young boys are playing outdoors and the man is smiling nearby" and "the kids are playing outdoors near a man with a smile" form a paraphrase pair.

a paragraph generator that is built on top of a sentence. The sentence generator encodes sentences into compact representations and the paragraph generator captures inter-sentence dependencies. [98] performed similar narratives from long videos by combining sentences using connective words at appropriate transitions learned using unsupervised learning.

4.2 Methodology

The vector representation of a sentence is extracted from an encoder-decoder model on sentence paraphrasing, and then tested on a text summarizer.

4.2.1 Vector Representation of Sentences

The sentence paraphrasing framework is considered as an encoder-decoder model, as shown in Figure 4.2. Given a sentence, the encoder maps the sentence into a vector (sent2vec) and this vector is fed into the decoder to produce a paraphrase sentence. The paraphrase sentence pairs are represented as (S_x, S_y). Let x_i denote the word embedding for sentence S_x; and y_j denote the word embedding for sentence S_y, $i \in \{1...T_x\}, j \in \{1...T_y\}$ where T_x and T_y are the length of the paraphrase sentences.

Several choices for encoder have been explored, including LSTM, GRU [99] and BNRHN [100]. In the developed model, an RNN encoder with LSTM cells is used since it was easy to implement and performs well on this model. Specifically, the words in S_x are converted into token IDs and then embedded using GloVe [25]. To encode a sentence, the embedded words are iteratively processed by the LSTM cell [95]. To

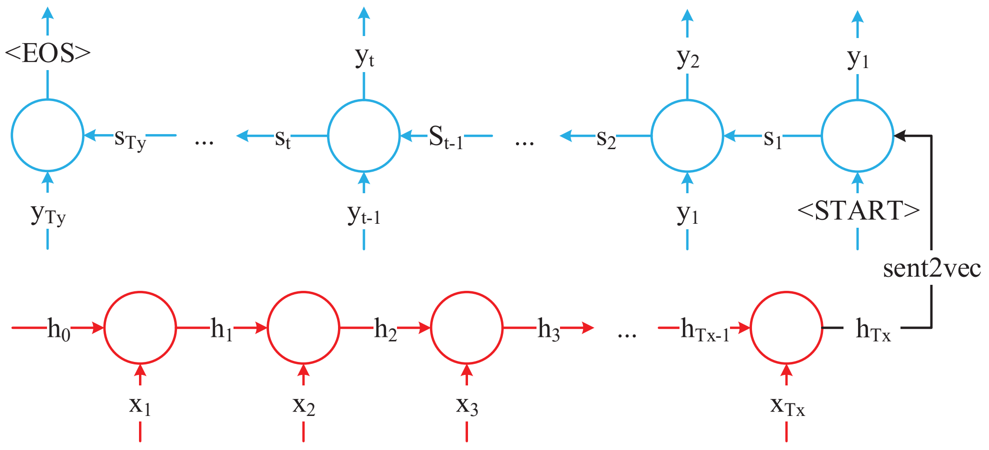


Figure 4.2: The sentence paraphrasing model. The red and blue cells represent encoder and decoder respectively. The intermediate vector in black (sent2vec) is vector encoded sentence.

encode a sentence, we iterate the following sequence of equations:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{4.1}$$

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{4.2}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{4.3}$$

$$g_t = \tanh(W_g \cdot [h_{t-1}, x_t] + b_g) \tag{4.4}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{4.5}$$

$$h_t = o_t \odot \tanh(c_t) \tag{4.6}$$

where h is the hidden state, W and b are weights and biases. $\odot$ denotes a component-wise product.

The decoder is a natural language model which conditions on the encoder output h_{T_x}. The computation is similar to that of the encoder. The vector h_{T_x} encodes the input sentence into a vector and is known as the vector representation of the input sentence, or “sent2vec” in this paper. Note that there is no attention between encoder and decoder. This ensures that all the information extracted from the input sentence by encoder goes through sent2vec. In other words, attention is not adopted in order to avoid information leakage.

In full softmax training, for every training example in the word level, we would need to compute logits for all classes L. However, this can get expensive when the vocabulary is very large. Given the predicted sentence and ground truth sentence, the sampled softmax [101] is used as a candidate sampling algorithm. In details, each training sample (x_i, t_i) consists of a context and one target class, and we write $P(y|x)$ for the probability of that the one target class is y given that the context is x. A small set $s_i \subset L$ of sampled classes is picked according to a chosen sampling function $Q(y|x)$ and a set of candidates C_i is created containing the union of the target class and the sampled classes.

$$C_i = S_i \bigcup t_i \tag{4.7}$$

Applying softmax, the training task figures out, given this set C_i, which of the classes in C_i is the target class.

$$P(t_i = y|x_i, C_i) = \frac{exp(W_{t_i}^T \phi(x_i))}{\sum_{y \in C_i} exp(W_y^T \phi(x_i))} \tag{4.8}$$

where ϕ is an affine transformation followed by a nonlinear activation and W is the corresponding weight (bias is omitted). Now the problem becomes how to find the small set S_i. Given training example (x_i, t_i), the likelihood function of extracting subset S_i from L according to distribution $Q(y|x)$ is

$$P(S_i = S|x_i) = \prod_{y \in S} Q(y|x_i) \prod_{y \in (L-S)} (1 - Q(y|x_i)) \tag{4.9}$$

i.e., each class $y \in L$ is included in S_i independently with probability $Q(y|x_i)$. In practice, log-uniform distribution, also known as Zipfian distribution, is adopted. Zipf's law states that given some corpus of natural language utterances, the frequency of any word is inversely proportional to its rank in the frequency table. Note that this operation is for training only. It is generally an underestimate of the full softmax loss. A common use case is to use this method for training, and calculate the full softmax loss for evaluation or inference.

4.2.2 Hierarchical Encoder for Text Summarization

Each sentence in a paragraph can be represented by a vector using the method described in Section 4.2.1. These vectors $x_i, i \in \{1...T_x\}$ are fed into a hierarchical encoder [102], and then the summarized text is generated word by word in an RNN decoder, as shown in Figure 4.3. This method first divides all T_x vectors into several chunks $(x_1, x_2, ..., x_n)$, $(x_{1+s}, x_{2+s}, ..., x_{n+s})$, ..., $(x_{T-n+1}, x_{T-n+2}, ..., x_T)$, where s is the stride and it denotes the number of temporal units adjacent chunks are apart. For each chunk, a feature vector is extracted using a LSTM layer and fed into the second layer. Each feature vector gives a proper abstract of its corresponding chunk. The LSTM units are also used in the second layer to build the hierarchical encoder. The first LSTM layer serves as a filter and it is used to explore local temporal structure within subsequences. The second LSTM learns the temporal dependencies among subsequences. As a result, the feature vector generated from the second layer, which is called “paragraph2vec”, summarizes all input vectors extracted from the entire paragraph. Finally, a RNN decoder converts “paragraph2vec” into word sequence $(y_1, y_2, ...)$, forming a summarized sentence.

A soft attention mechanism [103] is integrated in the hierarchical encoder. Specifically, dynamic weights are used to generate a new sequence $(v_1, v_2, ..., v_m)$:

$$v_t = \sum_{i=1}^{T_x} \alpha_i^{(t)} x_i \tag{4.10}$$

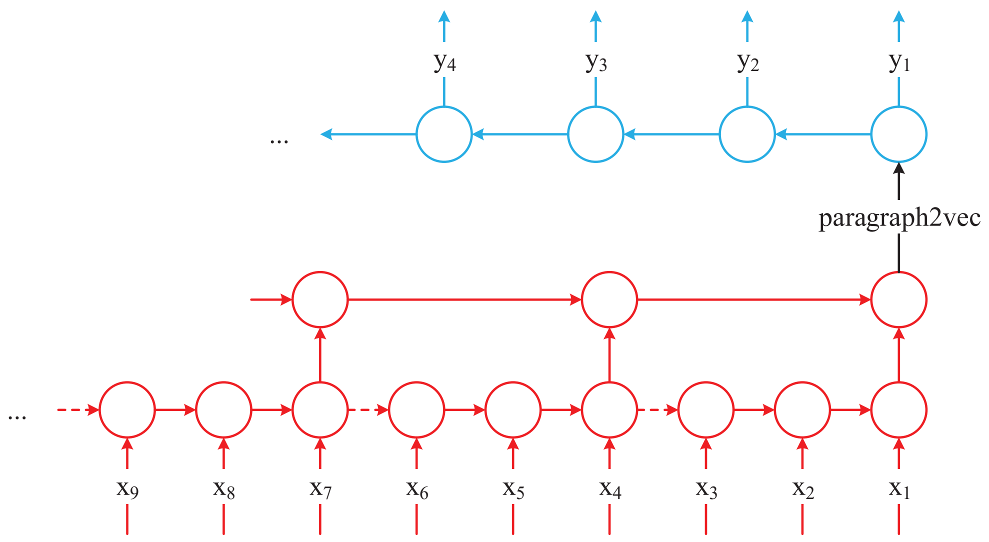


Figure 4.3: The paragraph summarizer. The red and blue cells represent encoder and decoder respectively. The encoder inputs x_i are the vector representation generated using sent2vec of the sentences in the paragraph. The decoder outputs y_i are the words in summarized text. The intermediate vector in black (paragraph2vec) is vector encoded paragraph. Dashed arrows indicate temporal boundaries.

where $\sum_{i=1}^{T_x} \alpha_i^{(t)} = 1$ and $\alpha_i^{(t)}$ is calculated at each time step $t = 1, 2, ..., m$:

$$\alpha_i = \exp(e_i^{(t)}) / \sum_{j=1}^{T_x} \exp(e_j^{(t)}) \tag{4.11}$$

and the relevance score $e_i^{(t)}$ is

$$e_i^{(t)} = W^T \tanh(W_a x_i + U_a h_{t-1} + b_a) \tag{4.12}$$

where W, W_a, U_a, b_a are trained parameters and h_{t-1} is the hidden state of the LSTM at the $(t-1)$th time step. The attention mechanism allows the LSTM to pay attention to different temporal locations of the input sequence. When the input sequence and the output sequence are not strictly aligned, attention can be especially helpful.

4.3 Experimental Results

4.3.1 Datasets

Visual Caption Datasets There are numerous datasets with multiple captions for images or videos. For example, MSR-VTT dataset [104] is comprised of 10,000 videos with 20 sentences each describing the videos. The 20 sentences are paraphrases since all the sentences are describing the same visual input. Pairs of these sentences are formed to create input-target samples. Likewise, MSVD [105], MSCOCO [32], and Flickr-30k [106] are used. Table 4.1 lists the statistics of datasets used. 5% of captions from all datasets was held out as a test set. In total, there are over 10M training samples that have been created.

Table 4.1: Sentence pairs statistics in captioning datasets.

	MSVD	MSRVTT	MSCOCO	Flickr
#sent	80K	200K	123K	158K
#sent/samp.	~42	20	5	5
# sent pairs	3.2 M	3.8 M	2.4 M	600 K

The SICK dataset The Sentences Involving Compositional Knowledge (SICK) dataset [107] is used as a test set for sentence paraphrasing task. It consists about 10,000 English sentence pairs, which are annotated for relatedness by means of crowd sourcing techniques. The sentence relatedness score (on a 5-point rating scale) for each sentence pair is provided and meant to quantify the degree of semantic relatedness between sentences. For human evaluation in section 4.2.1, the pool consists of the sentence pairs with the relatedness score above 3, resulting in 3,872 pairs within the $[3, 4)$ range, and 3,672 pairs within the $[4, 5]$ range.

TACoS Multi-Level Corpus The training pairs are extracted for the paragraph summarization task from TACoS Multi-Level Corpus [108]. This dataset provides coherent multi-sentence descriptions of complex videos featuring cooking activities with three levels of detail: "detailed", "short" and "single sentence" description. There are 143 training and 42 test video sequences with ∼20 annotations for each of the description levels in each sequence.

4.3.2 Training Details

The developed frameworks are implemented using TensorFlow [92] and trained with dual GeForce GTX 1080 graphics cards. Below are training details of two experiments.

Sentence Paraphrasing In the training phase, the <START> token is added at the beginning of the sentence and the <END> token is attached at the end of the sentence so that the model can generate sentences of varying lengths. In inference mode, the sentence generation is started with <START> and the word combination with highest probability will be selected. Thehe model is trained as described in Section 4.2.1 on the Visual Caption Datasets. The word embedding is initialized using GloVe [25]. The number of units per layer in both encoder and decoder are empirically set to 300 and 1024. Two sets of vocabularies are generated with size $20k$ and $50k$. Stochastic gradient descent is employed for optimization, where the initial learning rate and decay factor are set to 0.0005 and 0.99, respectively.

Paragraph Summarization: This task summarizes "detailed" description to "single sentence" in TACoS Multi-Level Corpus. Detailed descriptions with less than 20 sentences are selected. There are total of 3,176 samples, 2,467 are used for training and 709 are used for testing. The hierarchical architecture described in Section 4.2.2 with stride s of 5 is employed in this model. 20 feature vectors (short paragraphs are zero padded) are fed into the model, with each vector's sentence representation extracted from the developed paraphrasing model. To make the model more robust, soft attention is used between each layer. During training, the learning rate is set to 0.0001 and Adam optimizer is used. All the LSTM cells are set to 1024 units, except the one in sentence generation layer which is set to 256 units.

4.3.3 Sentence Paraphrasing

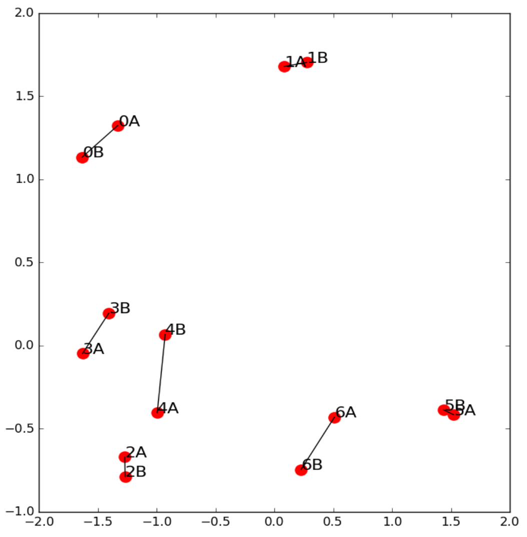
Given a reference sentence, the objective is to produce a semantically related sentence. The paraphrasing model was trained on Visual Caption Datasets and evaluated on the SICK dataset, without any fine-tuning. The results are shown in Table 4.2. The evaluation metrics for this experiment are Pearson's r, Spearman's ρ and Mean Squared Error (MSE). The setup is the same as it is in [109] for calculation of these metrics.

Table 4.2: Test set results on the SICK semantic relatedness task, where 300, 1024 denote the number of hidden units and 20k, 50k denote the size of the vocabulary. r and ρ are Pearson's and Spearman's metric respectively.

	r	ρ	MSE
sent2vec(300,20k)	0.7238	0.5707	0.4862
sent2vec(300,50k)	0.7472	0.5892	0.4520
sent2vec(1024,50k)	0.6673	0.5285	0.5679

In order to visualize the performance of the developed method, PCA is applied to the vector representation. Figure 4.4 visualizes some of the paraphrase sentence pairs in the SICK dataset. Representations are sensitive to the semantic information of the sentences since pairwise sentences are close to each other. For example, point 2A and 4A are close because "watching" and "looking" are semantically related.

The semantic relatedness and grammar correctness are verified by human generated scores. Each score is the average of 32 different human annotators. Scores take values between 1 and 5. A score of 1 indicates that the sentence pair is not at all related or totally incorrect syntax, while a score of 5 indicates they are highly related or grammatically correct. The sentences in human evaluation come from the Visual Caption and SICK test sets. The human evaluated scores for most sentence pairs are inversely proportional to the Euclidean distance between the vector representation of the corresponding sentences.



0A: the young boys are playing outdoors and the man is smiling nearby
0B: a group of kids is playing in a yard and an old man is standing in the background
1A: a brown dog is attacking another animal in front of the man in pants
1B: a brown dog is helping another animal in front of the man in pants
2A: two people are kickboxing and spectators are watching
2B: two people are fighting and spectators are watching
3A: kids in red shirts are playing in the leaves
3B: three kids are jumping in the leaves
4A: a little girl is looking at a woman in costume
4B: the little girl is looking at a man in costume
5A: a woman is removing the peel of a potato
5B: a woman is peeling a potato
6A: five children are standing in front of a wooden hut
6B: five kids are standing close together and one kid has a gun

Figure 4.4: Some paraphrase sentence pairs are represented by the sent2vec and then projected into 2D space using PCA. Each point represents a sentence in SICK dataset and the corresponding sentence is shown on the right.

4.3.4 Text Summarization

In addition to paraphrasing, sent2vec is useful for text summarization. Using the TACoS Multi-Level Corpus [108], sentences from detailed descriptions of each video sequence are first converted into vectors using the developed model. These vectors are fed into the summarizer described in Section 4.2.2. The performance of the summarized text is evaluated based on the metric scores and compared to skip-thoughts [30] and skip-gram

Table 4.3: Evaluation of short to single sentence summarization on TACoS Multi-Level Corpus using vectors from sent2vec, skip-thoughts, and skip-gram respectively.

	sent2vec	skip-gram	skip-thoughts
BLEU-1	0.479	0.514	0.520
BLEU-2	0.342	0.378	0.392
BLEU-3	0.213	0.245	0.276
BLEU-4	0.144	0.173	0.206
METEOR	0.237	0.250	0.253
ROUGE-L	0.48	0.509	0.522
CIDEr	1.129	1.430	1.562

[110]. Note that skip-gram is used as the frequency-based average of word2vec for each word in the sentence. As shown in Table 4.3, the scores generated by this model are very close and comparable to the benchmark skip-thoughts. This result is reasonable since the dataset used in training sent2vec are all from captions. The styles and topics of the sentences in this dataset are limited. However, the approach of forming sentence paraphrasing pairs and representing sentences using vectors are valid.

Figure 4.5 shows t-SNE [111] vector representation using (a) the developed sent2vec, (b) skip-thoughts and (c) skip-gram of randomly selected ∼15 test sequences. In these plots, each point represent a single sentence. Points describing the same video sequence should be clustered. Points with the same color are nicely grouped in sent2vec visualization.

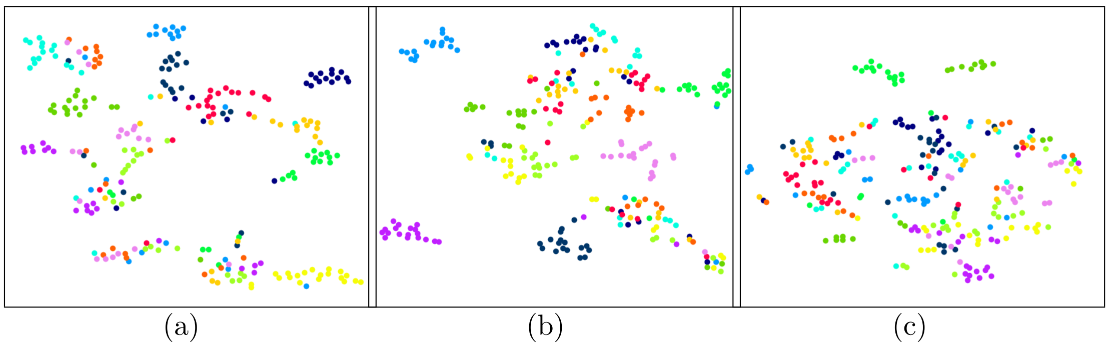


Figure 4.5: t-SNE visualizations of single sentence descriptions of a subset of all sequences on TACoS Multi-Level Corpus. (a) The developed sent2vec; (b) Skip-thoughts and (c) Skip-gram. Points are colored based on their sequence IDs. There are ∼20 different annotations for each sequence.

4.4 Discussion

This chapter showed the use of a deep LSTM based model in a sequence learning problem to encode sentences with common semantic information to similar vector representations. The presented latent representation of sentences has been shown useful for sentence paraphrasing and document summarization. We believe that reversing the encoder sentences helped the model learn long dependencies over long sentences. One of the advantages of the developed simple and straightforward representation is the applicability into a variety of tasks. Further research in this area can lead into higher quality vector representations that can be used for more challenging sequence learning tasks.

Chapter 5

Multi-Modal Vector Representation

This chapter is going to further explore the relationship between multimedia and vectors. The ultimate goal of this chapter is to find the common vector representation for different types of sources, as shown in Figure 5.1. In other words, given a(n) video sequence/image/audio/sentence/paragraph, we would like to extract an embedded vector containing the semantics of the source, and decode it to any type of the multimedia. The embedded vectors from different types of sources lie in a common space so that there is no need to align the vectors in the generative models. As an example, video text summarization converts a video sequences into a vector, and then the vector is used to generate text. In the other direction, the vector representation should be able to generate an image with related content. The common vectors form a space referred to as common vector space (CVS).

Considering the complexity of dealing with multiple types of sources, the study is focusing on images and text. Specifically, the common vector space deals with four source-target conversion: text-text (text2text), text-image (text2im), image-text (im2text), and image-image (im2im), as shown in Figure 5.2. Among these tasks, im2text is an image captioning model as illustrated in Chapter 3, and text2text is a sentence paraphrasing model described in Chapter 4. The generative model text2im and im2im are new problems we are going to address along with combination of the other two objectives.

Recent success in image captioning [33, 34, 35, 36] has shown that deep networks are capable of providing apt textual descriptions of visual data. In parallel, advances in conditioned image generation [37, 38, 39, 40] provide diverse images from a text based prior. An ambitious goal for machine learning in the vision and language domain is to be able to represent different modalities of data that have the same meaning with

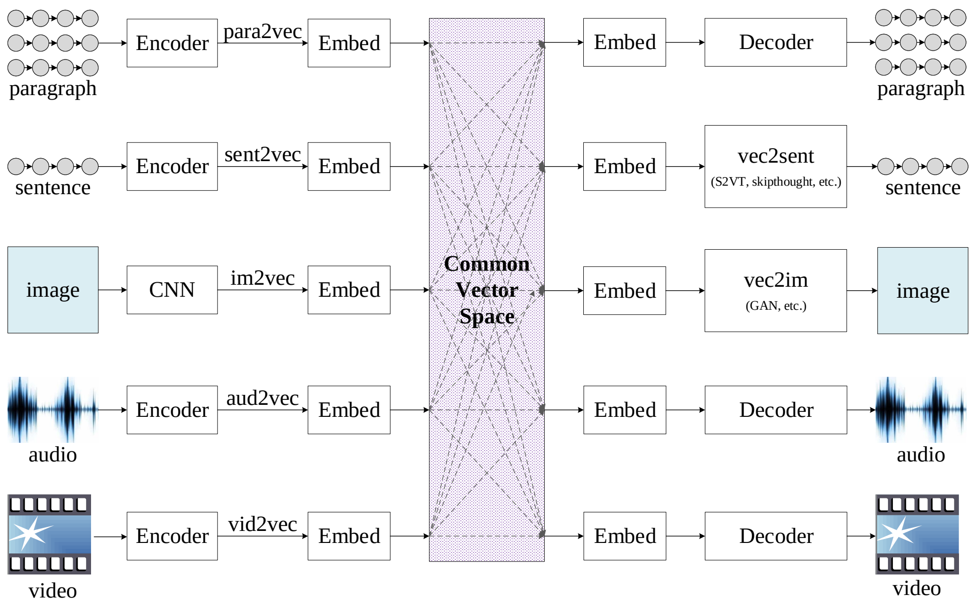


Figure 5.1: Different types of multimedia are semantically connected to each other by a common vector space (CVS).

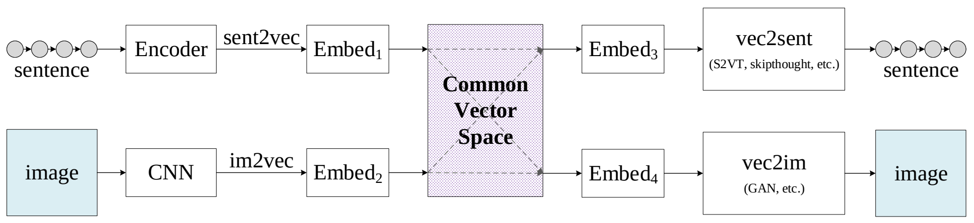


Figure 5.2: High level view of image-to-text conversion using CVS.

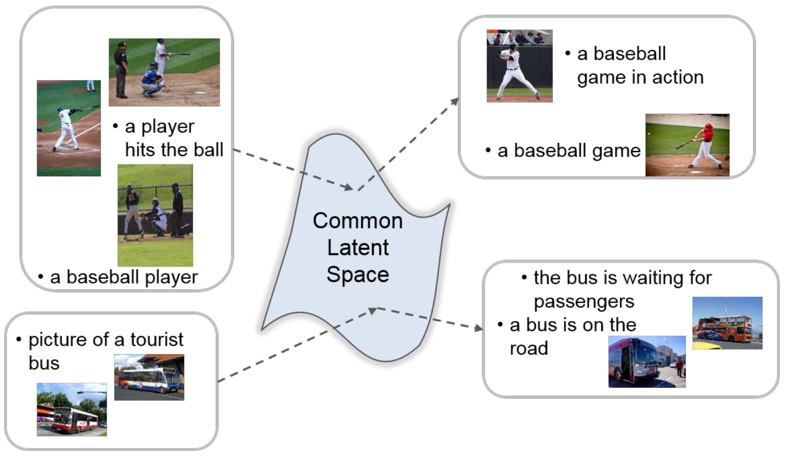


Figure 5.3: Overview of the bidirectional image-text generation model. To form the respective encoder and decoder networks, a CNN-GAN combination is used for visual and a recurrent machine translation model for text modalities.

a common latent representation. For example, words like “baseball” and “batter”, a sentence describing a baseball game, or image representations of a baseball game all refer to similar concepts. Generally, concepts that are semantically similar would lie close together in the descriptor’s space while dissimilar concepts would lie far apart. A sufficiently powerful model should be able to store similar concepts in a similar representation or produce any of these realizations from the same latent space. Successfully mapping visual and textual modalities in and out of this latent space would significantly impact the broad task of information retrieval.

This chapter introduces a cross-domain model capable of converting between text and image. The networks used in these domains are combined by merging the latent representations obtained during transition as shown in Figure 5.3. By modifying the cost function and introducing multiple sentence conditioning, the developed model, which we call Multi-Modal Vector Representation (MMVR) improves state of the art [37] by 23.7% (from 6.71 to 8.30 inception score).

The contributions of this work are as follows: 1) The formulation of a latent representation based model that merges inputs across multiple modalities; 2) The development of an n-gram based cost function that generalizes better to a text prior; 3) The improvements in image quality while using multiple semantically similar sentences for conditioning image generation on generalized text; 4) To advance qualitative measurement of text-to-visual models, an object detector based metric is introduced, and the human evaluations is conducted which compare the used metric to the standard inception score [41]. Results show that adding paraphrased sentences improves images quality across all three metrics. Along with quantitative evaluation, the qualitative evaluation is also performed through text and image arithmetic in latent space. The results demonstrate mathematical properties exhibited by latent representations for certain objects.

The rest of this paper is organized as follows: Section 5.1 reviews related work associated with models using latent representation and introduces the relevant prerequisites for MMVR. Section 5.2 details the MMVR and the introduced methodology. Section 5.3 describes the experiments along with the results. Concluding remarks are presented in Section 5.4.

5.1 Related Work

The notion of a latent space where similar points are close to each other is a key principle of metric learning. The representations obtained from this formulation generalize well when the test data has unseen labels. Models based on metric learning have been used extensively in the domain of face verification [112], image retrieval [113], person-re-identification [114] and zero-shot learning [115].

Multi-Modal Learning Using Vector Representation – Ngiam *et al.* [116] used an autoencoder model to learn cross-modal representations and showed results on audio and video datasets. Srivastava *et al.* [117] used deep Boltzmann machines to generate tags from images or images from tags. Sohn *et al.* [118] introduced a novel information theoretic objective that was shown to improve deep multi-modal learning for language and vision. Joint learning based on image category was shown in [119].

They used joint training for zero-shot image recognition and image retrieval. Sohn *et al.* [120] introduced multi-class N-tuple loss and showed superior results on image clustering, image retrieval and face re-identification. Eisenschtat *et al.* [121] introduced a 2-layer bidirectional network with batch-normalization and dropout techniques to map vectors coming from two data sources by optimizing correlation loss. Wang *et al.* [122] learned joint embeddings of images and text by enforcing margin constraints on training objectives. Recently, Wu *et al.* [123] leveraged this concept to associate data from different modalities. This work shares similarities with [123]. However, we focus on generating visual/textual data.

Conditional Image Generation – Generative Adversarial Networks (GANs) [124] are a sub-class of generative models based on an adversarial game. Training a GAN involves two models: a generator that maps a random distribution to the data distribution; and a discriminator that estimates the probability of a sample being fake or real. A GAN can produce sharp images but the generated images are not always photo-realistic. To improve upon photo-realistic quality, class category [40, 125, 126], caption [37, 38] or a paragraph [127] has been used to condition image generation. Reed *et al.* [38] encoded text into a vector to condition images, however direct encoding reduces the diversity of generated images. Introducing an additional prior on the latent code, Plug and Play Generative Networks (PPGN) [37] drew a wide range of image types and introduced an image conditioning framework. This work is complementary to such captioning and generative models as we define a common latent space that allows transitioning within and from modalities.

Sequence to Sequence Models – Sequence to sequence [95] models encode the inputs one at a time, then decodes one word at a time, using a recurrent neural network architecture. These models have been used in applications such as sentence vector representations [30, 128], visual question answering [129, 130] as well as video captioning [96, 131] that encodes the entire video, then decodes one word at a time. Paraphrasing sentences [132, 133] is another application of sequence to sequence models.

This work leverages the paraphrasing application to generate synthetic captions from a single caption to improve the quality of the generated images.

Image Captioning – Recent advances in recurrent neural networks have enabled generation of a natural language description of still images [33, 35, 34, 134]. The extension of this to video can be done by pooling over frames [21] or utilizing a fixed number of frames [22]. The developed model uses an image captioner to add a caption based prior on image generation.

5.2 Methodology

5.2.1 Multi-Modal Vector Representation

Inspired by [37], we introduce Multi-Modal Vector Representation (MMVR) to create a unified representation for visual and text modality in latent space. Given an image or sentence, MMVR performs iterative sampling to generate data in either modality while conditioning on an input. Figure 5.4 provides an overview of the MMVR architecture. The model can be divided into two interdependent modules: an image generator based on [135] and an image captioner based on [33].

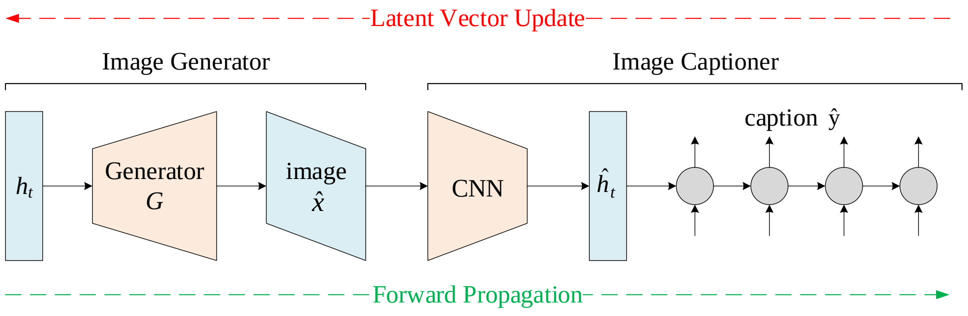


Figure 5.4: Overview of the MMVR model. It consists of two pre-trained modules – an image generator (G) that inputs a latent representation h_t and generates an image $\hat{x}$; and an image captioner that inputs an image $\hat{x}$ and generates a caption $\hat{y}$. To update the latent vector h_t, cross-entropy between the generated caption $\hat{y}$ and a ground truth caption y is used while the weights for the generator and CNN are fixed.

The forward pass is initiated by passing a random latent vector h_t into the image generator which generates an image $\hat{x}$. The image captioner uses the generated image to create a caption. Word-level cross entropy is used to determine the error between the generated caption, $\hat{y}$ and a ground truth caption y. This error is used to iteratively update h_t (and thus $\hat{x}$), while keeping all other components fixed. With each iteration, $\hat{y}$ approaches y, and the generated image $\hat{x}$ serves as a proxy for the target caption. The gradient associated with the cross-entropy error is specified in (5.1).

$$grad(C) = \frac{\partial \mathcal{L}(C_{pred}, C_{gt})}{\partial h_t} \tag{5.1}$$

where $grad(C)$ is the gradient of cross-entropy with respect to latent vector h_t, C_{pred} is the predicted caption and C_{gt} is a ground truth caption. $\mathcal{L}$ is the word level cross-entropy between the two captions. The $grad(C)$ component of the update rule ensures that the generated images have relevant context. However, to improve the realistic nature of the images, a reconstruction error is included in the update rule. This is computed as the difference between h_t and $\hat{h_t}$. This component is referred to as a Denoising Autoencoder (DAE) in [37]. Finally, to add diversity in generated images, a noise term $\mathcal{N}$ is also included. The resulting update rule is a weighted sum of four terms and is described in (5.2).

$$h_{t+1} = h_t + \gamma_1 grad(C) + \gamma_2 R(h_t, \hat{h_t}) + \mathcal{N}(0, \gamma_3) \tag{5.2}$$

where $R(h_t, \hat{h_t})$ is the reconstruction error which is computed as difference between h_t and $\hat{h_t}$, $\mathcal{N}$ is Gaussian noise with standard deviation γ_3 and h_{t+1} is the latent vector after the update. γ_1 and γ_2 are weights associated with the gradient of cross entropy and the DAE, respectively.

The update rule is based upon previous works on latent space interpolation [125, 37, 40]. The developed model updates the latent vector h iteratively, which is the input to the image generator, based on (5.2). It also encourages h and $\hat{h}$ to be similar, thereby creating a common latent representation capable of generating both images and sentences.

5.2.2 n-gram Metric Conditioning

An intrinsic limitation with the above introduced model is that the cross-entropy loss requires exact word level correspondences between generated and ground truth captions. For example, consider a case when the generator is conditioned on "a red car", whereas the captioner outputs "the car is red". Both captions are semantically very similar but lack one-to-one correspondence between words. This may result in unwanted updates of the latent vector h_t due to high word level cross-entropy. This problem is addressed by introducing a n-gram metric in the latent vector update. The metric is responsive to cases when generated and reference captions are different, but semantically similar.

(5.3) describes the updated γ_1 term when the n-gram metric is used in conjunction with cross-entropy. We compute word level differences and scale γ_1 with the n-gram metric between the generated and reference captions:

$$\gamma_1 \frac{1 - \mathcal{F}(C_{pred}, C_{gt})}{n} grad(C) \tag{5.3}$$

where $\mathcal{F}$ is the n-gram metric, such as BLEU [87] and CIDEr [89]. In the experiments, the BLEU scores is used as n-gram metric. As before, the latent vector h_t is obtained through an iterative process. When the captions are semantically similar, the magnitude of the update is significantly reduced by n-gram metric scaling, preventing unwanted updates to the latent vector.

5.2.3 Conditioning on Multiple Captions

Another way to overcome one-to-one word correspondences between a predicted and reference sentence is to use semantically similar sentences.

Multiple captions would increase syntactic variability for the generator to condition on, hence improving the overall image quality. The forward pass is performed in a same way as MMVR. As shown in Figure 5.5, the predicted caption is compared against multiple captions from a sentence paraphraser [136] to obtain the individual gradients. The aggregated gradients are used to update the latent vector h_t. The caption gradient component of the h_t update rule is replaced by the summation of gradients from multiple captions as

$$grad_{avg} = \frac{1}{N_C} \sum_{i=1}^{N_C} grad(C_i) \tag{5.4}$$

where N_C is the total number of reference captions and $grad_{avg}$ is the average gradient over the N_C captions.

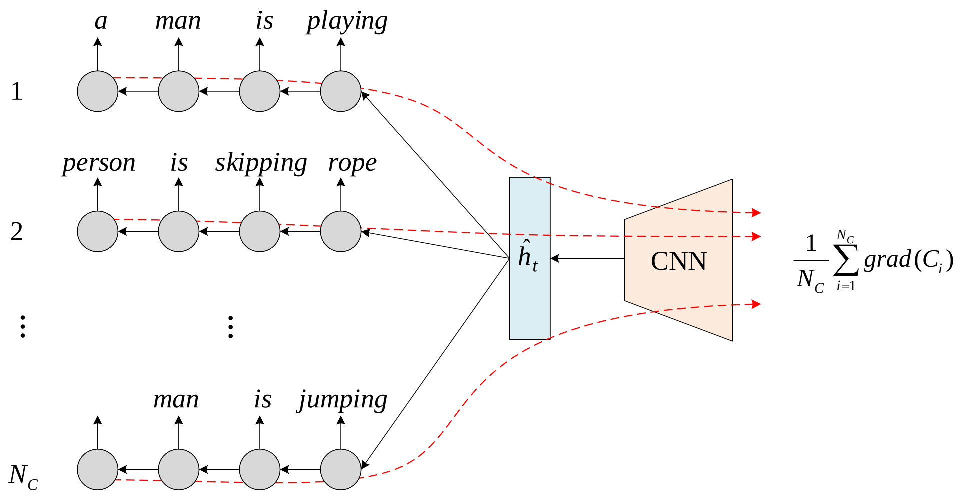


Figure 5.5: Conditioning the image generation through multiple captions by aggregating the gradients from individual caption cross-entropy. Solid black lines show the direction of forward pass during sentence generation and dashed red lines show direction of error back-propagation during latent vector update.

5.2.4 MMVR Architecture

The image generator use in the model is based upon DeePSiM [135] which comprises of three networks:

- an AlexNet [137] CNN encoder. It yields a 4096-dimensional vector.
- an inverted-AlexNet [138] based generator that up-samples the 4096-dimensional vector to an image of size 256×256.
- a discriminator that takes a 256×256 dimensional image and classifies it as real or fake.

Given an input image, the generator is trained to invert the features extracted from a pre-trained AlexNet and reconstruct the input image. A limitation of this generator is that it can only generate single object categories. A typical caption would be a description involving multiple object categories. In order to address this issue and improve conditioning on captions, fine-tuning the generator on MS-COCO [32] is needed. Thus, the fine-tuned generator is capable of rendering multiple objects in a image, a characteristic missing in the model trained on ImageNet [139]. Additional training details are provided in the supplementary material.

The image captioner uses a Long-term Recurrent Convolutional Network (LRCN) [33] which was trained on 82,783 images and 414,113 captions from the MS-COCO dataset [32]. The image captioner is used to steer the search for the 4096-dimensional vector required by the generator to render a representative image for the caption.

The developed framework uses a sentence paraphrasing model as described in Section 4.2.1 to obtain multiple semantically similar sentences from a single sentence. Three layers of attention based LSTM [103] are stacked in both encoder and decoder networks. For paraphrasing, the model is trained on captions from a combination of MS-COCO [32], MSVD [105], MSR-VTT [104] and Flickr-30k [106] image and video captioning datasets. Further training and dataset details are provided in the supplementary material.

The bi-modal nature of MMVR allows the model to take as input an image or a sentence which is then used to condition data generation in either modality. This section describes the transitions between visual and text modalities using the MMVR:

- **Visual-to-Text** – As described in Chapter 3, the visual-to-text is a simple transition in the model as a pre-trained image-captioner is an independent sub-module of the developed model.
- **Text-to-Visual** – As illustrated in Figure 5.4, the latent vector is initialized as a random vector. The forward propagation through the network provides a sentence as caption. The given text is considered as the ground truth of the image captioner. The word-level cross entropy between the generated caption and ground truth is calculated and back-propagated through the network to update the latent vector iteratively. Consequently, the image is also updated by the image generator along with the latent vector. Finally the generated image serves as the proxy for the target text.
- **Text-to-Text** – Paraphrasing sentences is achieved through the sequence-to-sequence model present in MMVR. The sequence-to-sequence model as shown in Figure 4.2 was pre-trained on a large corpus of similar sentences for the purpose of paraphrasing.
- **Visual-to-Visual** – An image can be translated into a visually different but semantically similar image. Starting with an image, we generate a caption. Using sentence paraphraser, a paraphrased caption is generated from the input caption. Then the process described in text-to-visual mode is performed to generate an alternate image representation. Paraphrased captions are employed to increase the image diversity, in addition to noise.

5.3 Results and Discussion

5.3.1 Inference

For the text-to-visual transition, the image captioner guides the generator during image generation. To this end, the model starts with a random 4096-dimensional vector h_t

to render a 256×256 image and iteratively update h_t. A resulting caption is obtained using a captioner that is compared with the ground truth caption and the difference between them is used to modify h. The process is terminated after 200 iterations and the resulting image is treated as a representative image for the caption. γ_1 and γ_2 hyper-parameters from (5.2) are set to 1 and 10^{-3}, respectively.

5.3.2 Evaluation Metrics

The image generation tasks are evaluated through qualitative comparisons as well as by quantitative metrics and human evaluations. In addition to using the inception score [41] metric, a new metric based on object detection is developed that captures the quality of multiple objects present in a generated image. A pre-trained YOLO object detector model [140] is used for this purpose. The model is trained on 80 object categories commonly present in the MS-COCO dataset. Figure 5.6 shows some examples with synthesized images. Each synthesized image is passed through the object detector model that yields bounding boxes and their corresponding confidences. Formally,

$$detection\ score = \sum_d \frac{A_d}{A_T} p_d \tag{5.5}$$

which reports the weighted sum of all detections (d) greater than a 0.1 confidence threshold (p_d), where the weight is the ratio of the detected bounding box area (A_d) and the full image area (A_T). Having an area weight is critical since some object detector models may predict a large number of small bounding boxes. The YOLO architecture intrinsically takes care of that and hence area weighting was not applied. Finally, the reported score is the average over the entire test set comprising of 1000 generated images.

Human Evaluations – Human evaluations are conducted to validate image generation. 50 image-caption pairs and asked 80 humans (not including any of the authors) are collected to judge the performance. Each participant was shown eight random images from all methods in random order totaling to 40 samples per person. Each evaluator

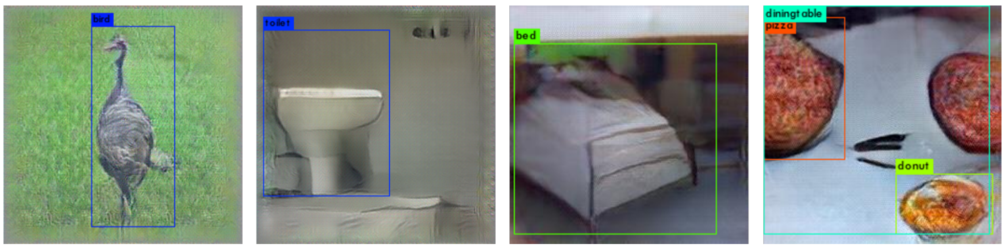


Figure 5.6: Examples of the YOLO object detection on generated images. The bounding boxes and corresponding labels are detections with confidence greater than 0.5 threshold.

was asked to rate on a 1 (bad) − 5 (good) Likert-type scale. On average, each method received more than 600 ratings. The questions asked to the human judges were: (1) Can you identify any one object in the image? and (2) How well does the sentence align with the image? The human evaluation results are shown in Table 5.1 and 5.2.

5.3.3 Text-to-Visual

An important property of common latent space is cross-modal transformations. Cross-modal experiments aid in proving that the representations of individual modalities are well aligned in the common space. Figure 5.7 shows examples of text-to-visual generation. It can be observed that MMVR synthesizes reasonable images from captions. As noted in [37], one of the major challenges while conditioning on text include the cross-entropy computation from a sentence with many words. The captions could be 10-15 words long including stop-words which have limited significance on the image content. Moreover, gradients for all words are aggregated and back-propagated, hence significant words may loose importance. This may result in poor image quality. The inclusion of n-gram scaling to the update function and conditioning on multiple ground truth sentences help address such limitations. One can observe that the captioner generating good captions even for unrealistic images. These could be "fooling" images [39] which are unrecognizable to humans but deep neural networks recognize them with high confidence.

Table 5.1 compares the text-to-visual techniques against a baseline (direct FC-6).

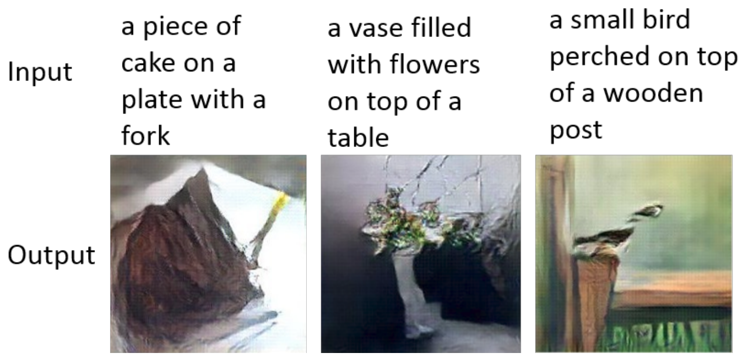

Input
a piece of cake on a plate with a fork
a vase filled with flowers on top of a table
a small bird perched on top of a wooden post
Output

Figure 5.7: Examples of text-to-visual transformation.

The inception scores indicate the improvement in generated images when BLEU-1 (B-1) and the multiple caption conditioning ($N_C = 5$) are used. The detection scores for multiple captions are significantly better than other variants. However, BLEU-1 is slightly lower than the baseline result. Despite having worse inception scores, baseline methods got higher human evaluation scores. Possibly the reason for this trend is lack of detail in objects generated by multiple captions. The baseline model generates images with single objects, making them visually appealing.

Table 5.1: Evaluation of the generated image quality using the inception, detection and human scores on the test set.

Method	Inception	Detection	Human
Baseline (FC-6)	5.77 ± 0.96	0.762	**2.95**
PPGN [37]	6.71 ± 0.45	0.717	2.34
MMVR (B-1)	7.22 ± 0.81	0.713	2.31
MMVR ($N_c = 5$)	$\mathbf{8.30 \pm 0.78}$	**1.004**	2.71

Conditional Image Generation on Multiple Sentences – To understand the effect of conditioning image generation on multiple sentences, experiments are performed by varying the number of sentences. Synthetic sentences were generated using a sentence paraphraser [136]. Figure 5.8 shows the input caption and the generated images with 1, 3 and 5 captions. Image quality enhances with increase in number of sentences. The *food* example also show gains in understanding the concept of quantity (*four*) through text. Similar trends are observed through the inception and detection score metrics as reported in Table 5.2. The detection score helps prove that multiple sentences assist in generating multiple objects in the image that are recognized by the object detector.

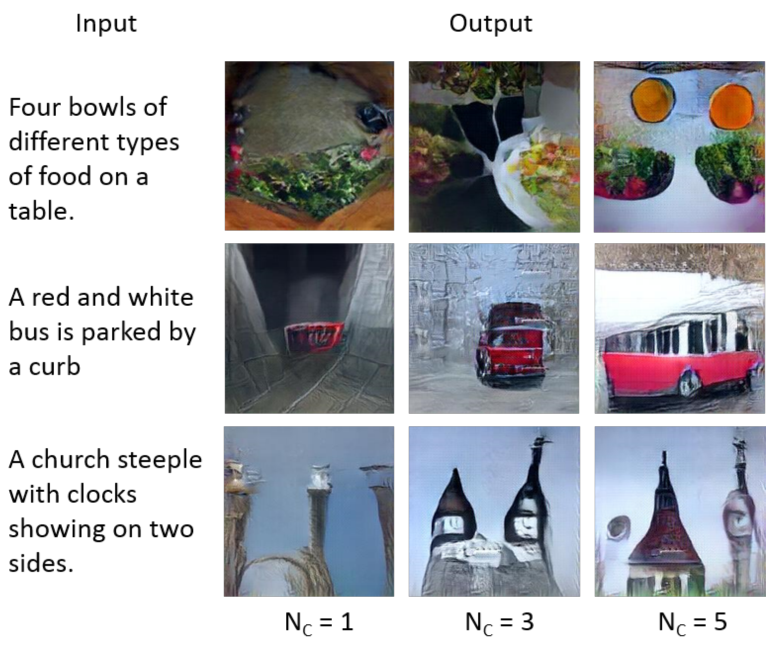

Input
Output
Four bowls of different types of food on a table.
A red and white bus is parked by a curb
A church steeple with clocks showing on two sides.
$N_C = 1$
$N_C = 3$
$N_C = 5$

Figure 5.8: Examples of the text-to-image generation as conditioned on varying number of input captions. One can observe more detailed images being synthesized with increase in captions.

Table 5.2: Evaluation of the generated image quality by conditioning on varying number of paraphrased sentences (N_C).

N_C	Inception	Detection	Human
1	7.22 ± 0.81	0.713	2.30
3	8.04 ± 0.57	0.915	**2.73**
5	$\mathbf{8.30 \pm 0.78}$	**1.005**	2.71

Was the n-gram scaling useful?

Figure 5.9 shows examples with and without the n-gram scaling of the gradient term in (5.3). It is very difficult to judge the two techniques visually. This experiment uses only a single caption to condition the image generator to have a fair comparison in this case. The BLEU-1 score was used as the word level error multiplier and it scales the gradients accordingly. The inception scores in Table 5.1 show slight improvement for BLEU-1 against the PPGN.

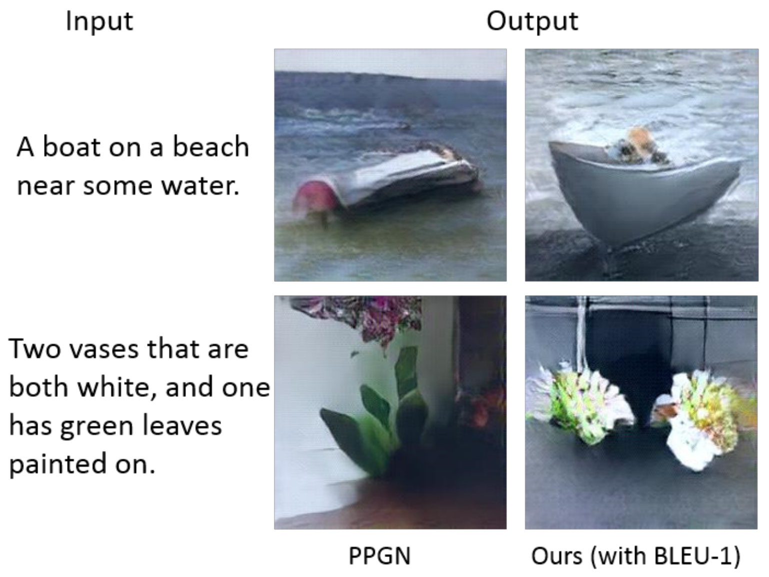

Input
Output
A boat on a beach near some water.
Two vases that are both white, and one has green leaves painted on.
PPGN
Ours (with BLEU-1)

Figure 5.9: Examples comparing the text-to-image for PPGN and the BLEU-1 scaled cross-entropy. Even though slight improvements could be observed with the n-gram scaling, judging the image quality visually is very challenging.

Which degree n-gram is better for scaling?

Different BLEU scaling in (5.3) are compared by varying the n-gram metric. Results are reported in Table 5.3. One reason the BLEU-1 performs better than the higher n-gram techniques might be the simple removal of one-to-one word correspondences between the predicted and ground truth captions is sufficient. Higher BLEU metrics require n-gram matching which puts hard constraints on the generated caption. This may dampen the significance on important words in the overall update.

Do stop words have significance?

A caption might have more stop words ("a", "an", "the", "to", etc.) than actual informative words that describe image content. Experiments are performed by masking the gradient for the stop words. This did not improve the image quality. This is

Table 5.3: Comparison of image quality with different BLEU metrics for scaling the latent vector update function.

Scaling n-gram Metric	Inception Score
BLEU-1	$\mathbf{7.22 \pm 0.81}$
BLEU-2	7.12 ± 0.66
BLEU-3	7.05 ± 0.73
BLEU-4	6.83 ± 0.74

attributed to the lack of sentence structure after masking stop words. The captioner was trained to generate complete English language sentences. It always generates a complete text description, even though the ground truth caption may be a collection of only relevant words. Hence, all other experiments take the running average of the number of words in the caption so all words contribute equally.

Does fine-tuning the image generator help?

The generator was unable to address common words that occur in a caption (“man”, “woman”, “person”, *numbers*, etc.) since ImageNet does not contain such categories. Moreover, some dominant categories in MS-COCO dataset like *giraffe*, *stop sign* and *person* are not present in ImageNet dataset. By fine-tuning, the generator is able to semantically capture such categories. Additionally, one can observe that multiple objects could also be generated since the original ImageNet model mostly comprised of single object images. An example caption and generated images are shown in Figure 5.10. It could also be interpreted that the generator model correlates better with the captioner since the caption cross-entropy is computed on MS-COCO trained captioner.

5.3.4 Text Generation

Similar to image generation, both input modalities can independently yield text as output. Since LRCN [33] is used, the evaluation of the visual-to-text mode is performed on the test partition of the MS-COCO dataset. Examples are shown on the left side of Figure 5.11.

MMVR in language translation tasks is useful. Given a reference sentence, the objec-

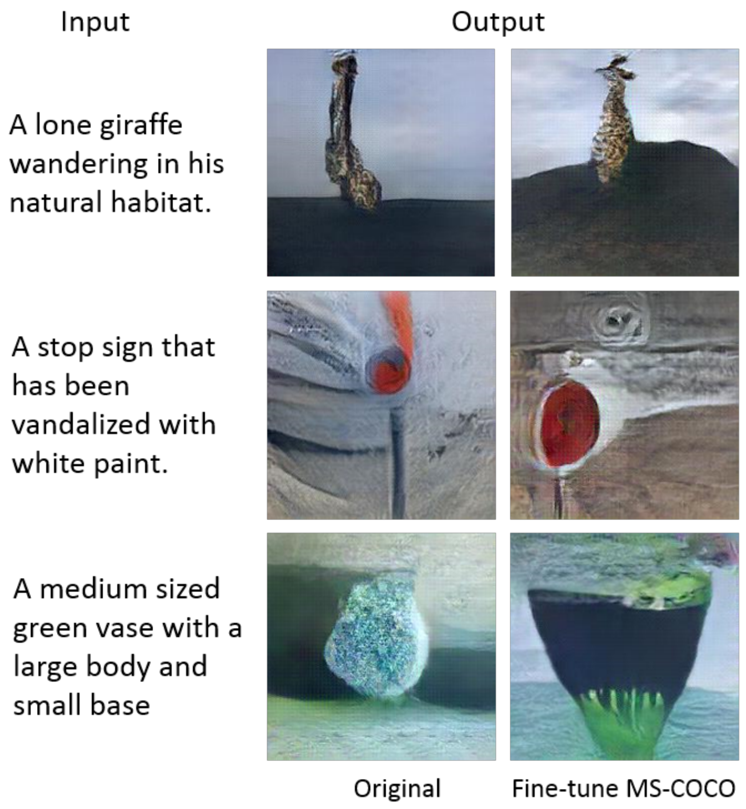

Input
Output
A lone giraffe wandering in his natural habitat.
A stop sign that has been vandalized with white paint.
A medium sized green vase with a large body and small base
Original
Fine-tune MS-COCO

Figure 5.10: Examples that show text-to-image improvements after fine-tuning the generator on MS-COCO dataset. Object categories such as *giraffe* and *stop sign* that are not part of ImageNet dataset show some enhancement in details. One can also observed slight improvements in understanding of size, shape and quantity aspects.

tive is to produce a semantically related sentence. The right side in Figure 5.11 shows examples of paraphrasing. Furthermore, to test the robustness of the sentence paraphraser, experiments were ran by varying noise levels in the latent space. To evaluate the quality of generated captions, the model uses BLEU [87], METEOR [31], CIDEr [89] and ROUGE [88] natural language metrics. Since every sample from MS-COCO dataset consists of five captions, one of the captions is used as the input to the paraphraser and the remaining four captions for evaluation. The input caption is fed in the encoder to obtain a vector representation. This representation is corrupted using random uniform noise before being input to the decoder. The results are reported in Table 5.4, where

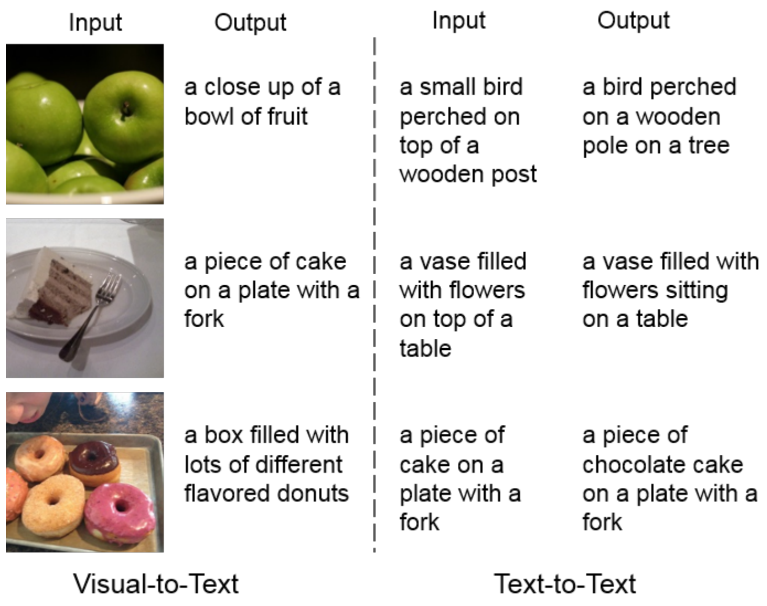

Input
Output
Input
Output
a close up of a bowl of fruit
a small bird perched on top of a wooden post
a bird perched on a wooden pole on a tree
a piece of cake on a plate with a fork
a vase filled with flowers on top of a table
a vase filled with flowers sitting on a table
a box filled with lots of different flavored donuts
a piece of cake on a plate with a fork
a piece of chocolate cake on a plate with a fork
Visual-to-Text
Text-to-Text

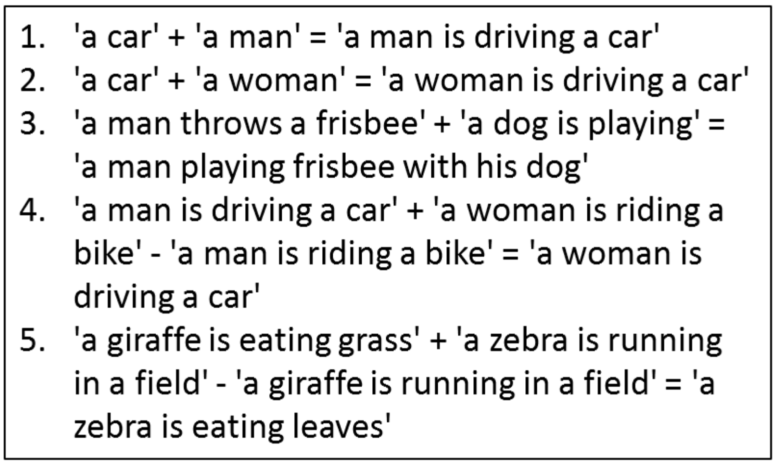
Figure 5.11: Examples of the visual-to-text (left) and text-to-text (right) modes of the MMVR. The inputs can be the visual or text modalities.

the scale is the noise multiplier. A scale of 0.0 is equivalent to feeding the latent vector without any noise and could be considered as the upper-limit of the paraphraser. Observation shows that the model is robust to noise up to 1 standard deviation but the performance degrades significantly beyond that. This also indicates that the sentences do not form very dense clusters in the latent space.

Table 5.4: Evaluation of Text-to-Text paraphrasing model with variation of noise in the latent vector space. The noise scale is the multiplier for the standard deviation of the feature space to generate random uniform noise. Noise with scale 0.0 could be considered as the upper-limit of the paraphraser.

Scale	0.0	0.1	0.5	1.0	2.0	3.0
BLEU-1	**0.71**	0.71	0.7	0.68	0.56	0.3
BLEU-2	**0.53**	0.53	0.52	0.5	0.38	0.15
BLEU-3	**0.38**	0.38	0.38	0.35	0.24	0.07
BLEU-4	**0.27**	0.27	0.27	0.24	0.16	0.04
METEOR	**0.24**	0.24	0.24	0.23	0.18	0.09
ROUGE	**0.52**	0.52	0.52	0.5	0.42	0.24
CIDEr	**1.03**	1.03	1.02	0.92	0.59	0.15

Vector Arithmetic in Latent Space – Lastly, the text-to-text model is evaluated by performing arithmetic operations in the latent space. Vector arithmetic for language has been shown with words [25] but is still in a nascent stage for complete sentences. The input sentences are fed in the encoder to obtain vector representations. A composite vector is obtained after performing simple mathematical operations on the vector and is fed to the decoder to generate a sentence description. Examples are shown in Figure 5.12. The first three samples demonstrate simple additive properties. Samples 4 and 5 validate more complicated operations and show relationships between objects and actions in the latent space.



1. 'a car' + 'a man' = 'a man is driving a car'
2. 'a car' + 'a woman' = 'a woman is driving a car'
3. 'a man throws a frisbee' + 'a dog is playing' = 'a man playing frisbee with his dog'
4. 'a man is driving a car' + 'a woman is riding a bike' - 'a man is riding a bike' = 'a woman is driving a car'
5. 'a giraffe is eating grass' + 'a zebra is running in a field' - 'a giraffe is running in a field' = 'a zebra is eating leaves'

Figure 5.12: Examples of arithmetic operations in the latent space for the text-to-text model.

5.4 Conclusion

This work advances the area of caption conditioned image generation by allowing the vector space to be shared between vision and language representations. MMVR shows flexibility in performing cross-modal transformations and improves state-of-the-art by more than 20%. This chapter addressed some limitations such as one-to-one word correspondence by using a n-gram metric and conditioning on multiple semantically similar sentences. Also, this chapter introduced a new objective metric for evaluating generated images which allows for multiple objects per generated image. We believe this to be the first effort to directly tie a common vector connection space in a bidirectional visual to text framework by adopting image and text generative techniques.

Chapter 6

Conclusions and Future Work

This chapter will first draw and summarize conclusions from this current research, and then discuss some future work that should be carried out related to this study.

6.1 Conclusions

The addressed problems in this research are: 1) Developing a decision level video analysis scheme that performs the spatio-temporal segmentation of video sequences and designing a video text summarization architecture to automatically extract the text description from a video sequence; 2) Exploring a better RNN architecture to ensure the stable gradient distribution during training, which can be used in computer vision tasks, such as image captioning; 3) Researching a new way to encode sentences into vectors so that the vision/language based deep networks can make use of this embedding method, thus providing remarkable performance in computer vision tasks; 4) Exploring a latent common vector representation of different types of sources (modalities), such as visual inputs and languages, achieving the goal of conversion between different modalities with no need to align the vectors in the generative models.

Chapter 2 developed a novel and accurate coarse-to-fine approach to segment the salient object in video sequences and generate interesting and pleasing visual outputs.

This approach involves a parallel scheme which consists of utilizing the KLT, SSC and 3D SLIC algorithms to identify the approximate location of the most salient object and as a result, an unsupervised graph-based method can be used in subsequent step. The experimental results on SegTrack [71] dataset and Kodak Alaris Consumer Video Dataset reveal that this method outperforms the state-of-the-art supervised and unsupervised approaches in terms of accuracy and robustness.

Chapter 3 introduced a novel recurrent neural network model that is based on batch normalization and recurrent highway networks. The analyses provide insight into the ability of the batch normalized recurrent highway model to dynamically control the gradient flow across time steps. Additionally, this model takes advantages of faster convergence compared to the original RHN, and keeps the feature of increasing depth in the recurrent transitions while retaining the ease of training. Experimental results on image captioning task reveals that the developed model achieves high METEOR and BLEU scores compared to previous models. Moreover, the theoretical advantages were supported by the results and analysis.

The work in Chapter 4 inspired us to use a deep LSTM based model in a sequence learning problem to encode sentences with common semantic information to similar vector representations. The presented latent representation of sentences has been shown useful for sentence paraphrasing and document summarization. We believe that reversing the encoder sentences helped the model learn long dependencies over long sentences. Experimental results help gain insight how vector representations are suitable for advanced language embedding. One of the advantages of this simple and straightforward representation is the applicability into a variety of tasks. Further research in this area can lead into higher quality vector representations that can be used for more challenging sequence learning tasks.

Image captioning models have been shown to be capable of generating plausible text given input images or videos. Further, recent work in image generation has shown significant improvements in image quality when text is used as a prior. Chapter 5 ties these concepts together by creating an architecture that can enable bidirectional generation of images and text. We call this network Multi-Modal Vector Representation (MMVR). Along with MMVR, two improvements are introduced to the text conditioned image generation. Firstly, a n-gram metric based cost function is introduced that generalizes the caption with respect to the image. Secondly, multiple semantically similar sentences are shown to help in generating better images. Qualitative and quantitative evaluations demonstrate that MMVR improves upon existing text conditioned image generation results by over 20%, while integrating visual and text modalities.

To summarize, the main contributions of this thesis are listed below.

- This research developed a novel coarse-to-fine framework and prototype system for automatically segmenting a video sequence and extracting a salient moving object from it. The study showed that the developed coarse-to-fine framework is capable to generate interesting and pleasing visual outputs.

- This research developed Batch-Normalized Recurrent Highway Networks (BNRHN)– a novel recurrent framework based on recurrent highway networks for sequence modeling using batch normalization. It has been shown that in computer vision tasks, such as image captioning, BNRHN reasonably controls the gradient flow and effectively strengthens the ability of processing the data as sequences by increasing the depth of the network

- This research explored the semantic sentence embedding method for paraphrasing and text summarization. This work made use of sentences from widely available image and video captioning datasets to form sentence paraphrase pairs, whereby these pairs are used to train the encoder-decoder model. The application was also demonstrated of the sentence embeddings for paragraph summarization and sentence paraphrasing, whereby evaluations are performed using metrics, vector visualizations and qualitative human evaluation. Moreover, we expend of the vectorized sentence approach to a hierarchical architecture, enabling the encoding of more complex structures such as paragraphs for applications such as text summarization.

- This research developed a cross-domain Multi-Modal Vector Representation (MMVR) model, capable of converting between text and image. A latent representation based model was formulated that merges inputs across multiple modalities introduced an n-gram based cost function that generalizes better to a text prior. The improvements in image quality were shown while using multiple semantically similar sentences for conditioning image generation on generalized text. To advance qualitative measurement of text-to-visual models, an object detector based metric was introduced, and human evaluations were conducted which compare the used metric to the standard inception score [41].

- This research introduced the concept of common vector space (CVS), which ensures the embedded vectors from different types of sources lie in a common space so that there is no need to align the vectors in the generative models. Also, the vectors embedded from semantically similar modalities are close in the common space, whereas dissimilar patches are further away.

6.2 Future Work

Recently, deep learning has enabled dramatic advancement in image, video and text understanding. For example, image classification [137, 141, 10, 142], object detection [143, 144], image captioning [33, 134], localized image description [145], image and sentence retrieval [146, 147] and visual question answering [148] tasks have witnessed tremendous progress in the last few years. However, general vision or language models are hard to emerge within a paradigm that focuses on the particularities of a single metric, dataset, and task.

Chapter 5 introduced a model that achieve the goal of conversion between images and texts using MMVR. This model successfully relates different modalities by a latent vector space. Note that this is not the only way to discover the latent vector representation of different type of sources. In fact, a straight-forward way is worthy to be considered: utilizing the unified Common Vector Space (CVS) for vision and language, that spans across five broad tasks: classification, captioning, object detection, retrieval and visual question answering. By learning a common vector space, the similar inputs from different modalities cluster together. It is expected that the combination of these resources will facilitate research in multitask learning, transfer learning, general embeddings and encoders, architecture search, zero-shot learning, general purpose question answering, meta-learning, and other related areas of vision and language. The novelty of CVS includes: 1) The formulation of an efficient vector space based model using neural embeddings that act as a bridge between vision and language modalities and is easily expandable to new modalities; 2) The multi-modal loss function that includes metric loss, category loss and adversarial loss terms. The adversarial framework includes within-modality and across modality discriminators.

6.2.1 Common Vector Space

The new CVS model deals with multiple modalities. For simplicity, taking image and text as an example, Figure 6.1 shows the high level structure of the CVS model. The images and texts are encoded by their encoders (CNN for image encoding and sent2vec for text encoding) respectively, resulting in the initial vector representations of the input modalities, h_i and h_t. The size of each vector in both paths are unified by the fully connected layer F_i and F_t so that they can be sent into the subsequent embedding block F_c. Note that embedding F_c can be single or consist of multiple fully connected layers, and the embedding F_c layer(s) from different modalities are shared weights during training. With proper loss function (see Section 6.2.2 for details) adopted, this architecture leads the resulting vector in a common space. The input of the CVS model can be the pair of the modalities, for example, image-image, sentence-sentence, or image-sentence pairs (based on the task). During training, the input also includes the label for the pair and a ground truth output. The CVS model outputs image or sentence based on the task, for example, bounding box of image, retrieved image or sentence, class category of input, generated caption, etc.

6.2.2 Losses for Metric Learning

Similarity between objects plays an important role in both human cognitive processes and artificial systems for recognition and categorization. Many approaches in machine learning relies on the distance/similarity metric between two samples, for example, Euclidean distance. The problem is that each problem has its own semantic notion of similarity, which is often badly captured by standard metrics. In other words, these distances are problem dependent. How to appropriately measure such similarities for a given task is crucial to the performance of many machine learning, pattern recognition and data mining methods. This section is devoted to the metric learning, a set of losses to learn similarity and distance functions from data that has attracted a lot of interest in machine learning and related fields in the past years. This section provides a thorough review of the metric learning losses that can be utilized in the CVS model potentially.

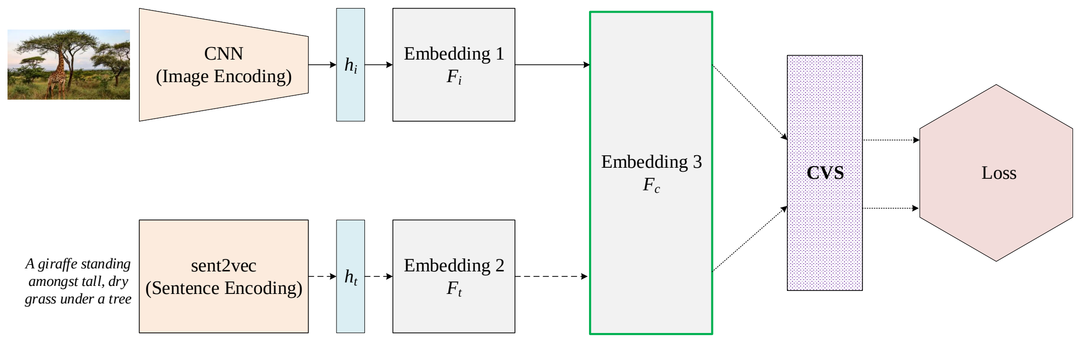


Figure 6.1: The CVS model training architecture. The solid arrows and dashed arrows represent image path and text path respectively, the dotted line indicates the connection to CVS.

Triplet Loss

Usually in supervised learning there is a fixed number of classes and the network is trained using the softmax cross entropy loss. However in some cases it needs to be able to have a variable number of classes. In face recognition [112] for instance, the model compares two unknown faces and indicates whether they are from the same person or not. Triplet loss in this case is a way to learn good embeddings for each face. In the embedding space, faces from the same person should be close together and form well separated clusters.

The goal of the triplet loss is to make sure that two examples with the same label have their embeddings close together in the embedding space, and with different labels have their embeddings far away. However, it is not expected to push the train embeddings of each label to collapse into very small clusters. The only requirement is that given two positive examples of the same class and one negative example, the negative should be farther away than the positive by the margin. Similarly to the margin used in Support Vector Machines (SVMs), the clusters of each class are separated by the margin. As shown in Figure 6.2, to formalize this requirement, the loss is defined over triplets of

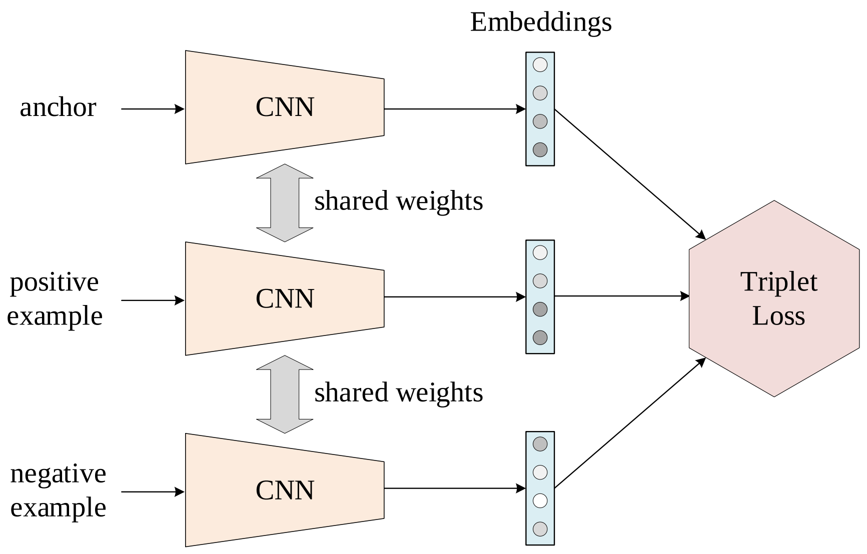


Figure 6.2: Triplet loss on two positive examples and one negative example.

embeddings: an anchor a, a positive example of the same class as the anchor p, and a negative example of a different class n. In terms of CVS model, the anchor and positive example indicate the modalities with similar semantics, and negative example refers to the modality which is semantically different from the anchor.

For some distance on the embedding space d, the loss of a triplet (a, p, n) is

$$L = \max(d(a, p) - d(a, n) + \Delta, 0) \tag{6.1}$$

where Δ indicates the margin. This loss is minimized, that pushes $d(a, p)$ to 0 and $d(a, n)$ to be greater than $d(a, p) + \Delta$. As soon as n becomes an "easy negative", the loss becomes zero.

Based on the definition of the loss, there are three categories of triplets:

- Easy triplets: triplets which have a loss of 0, because $d(a, p) + \Delta < d(a, n)$.
- Hard triplets: triplets where the negative is closer to the anchor than the positive, *i.e.*, $d(a, n) < d(a, p)$
- Semi-hard triplets: triplets where the negative is not closer to the anchor than the positive, but which still have positive loss, *i.e.*, $d(a, p) < d(a, n) < d(a, p) + \Delta$.

Each of these definitions depend on where the negative is, relatively to the anchor and positive. Therefore, these three categories can be extended to the negatives: hard negatives, semi-hard negatives or easy negatives. The three corresponding regions of the embedding space for the negatives are shown in Figure 6.3.

Choosing the kind of triplets to train on will greatly impact the metrics. In the CVS model, a random semi-hard negative can be picked for every pair of anchor and positive samples, and train on these triplets.

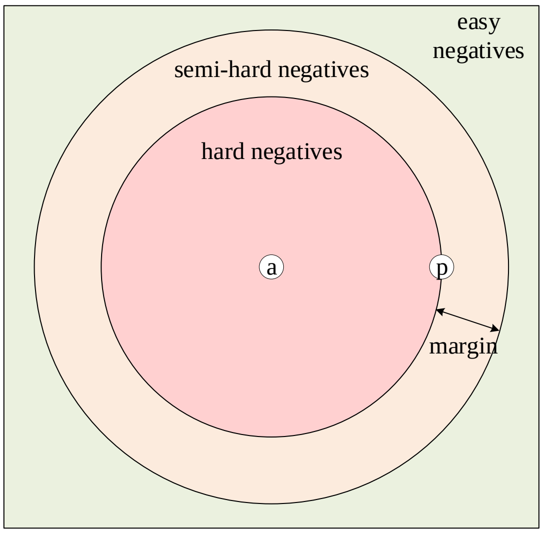


Figure 6.3: The three types of negatives, given an anchor and a positive.

N-pair Loss

The fundamental philosophy behind triplet loss is the following: for an input (query) example, it is desire to shorten the distances between its embedding vector and those of positive examples while enlarging the distances between that of negative examples. However, during one update, the triplet loss only compares an example with one negative example while ignoring negative examples from the rest of the classes. As a consequence, the embedding vector for an example is only guaranteed to be far from the selected negative sample but not necessarily the others. Thus it ends up only differentiating an example from a limited selection of negative samples yet still maintain a small distance from many other classes. In practice, the hope is that, after looping over sufficiently many randomly sampled triplets, the final distance metric can be balanced correctly; but individual updates can still be unstable and the convergence would be slow. Specifically, towards the end of training, most randomly selected negative examples can no longer yield non-zero triplet loss error. An evident way to improve the vanilla triplet loss is to select a negative example that violates the triplet constraint. However, hard negative data mining can be expensive with a large number of output classes for deep metric learning. According to [120], the alternative can be a loss function that recruits multiple negatives for each update. In this case, an input example is being compared against negative examples from multiple classes and it needs to be distinguishable from all of them at the same time. Ideally, the loss function incorporates examples across every class all at once. But it is usually not attainable for large scale deep metric learning due to the memory bottleneck from the neural network based embedding. Motivated by this thought process, [120] propose the computationally feasible N-pair loss, which approximates the ideal loss by pushing N examples simultaneously.

Consider an $(N+1)$-tuplet of training examples $\{x, x^+, x_1, ..., x_{N-1}\}$, the embeddings of anchor x, the positive example x^+, and negative examples $x_i, i = 1, ..., N-1$ are represented as a, p, and n_i. The $(N+1)$-tuplet loss is defined as

$$L = \log\big[1 + \sum_{i=1}^{N-1} \exp(a^T n_i - a^T p)\big] \tag{6.2}$$

where the natural exponential hugely increase the probability of the biggest score and decrease the probability of the lower scores when compared with standard normalization, and also it makes the loss always be non-negative during training. Suppose the $(N+1)$-tuplet loss is directly applied to the deep metric learning framework. When the batch size of SGD is M, there are $M \times (N+1)$ examples to be passed through the embedding vectors at one update. Since the number of examples to evaluate for each batch grows in quadratic to M and N, it again becomes impractical to scale the training for a very deep convolutional network. The solution is to make use of an effective batch construction to avoid excessive computational burden. Let $\{(x_1, x_1^+), ..., (x_N, x_N^+)\}$ be N pairs of examples from N different classes (denoted by y_i, where $i \in \{1, 2, ..., N\}$). N tuplets, denoted as $\{S_i\}_{i=1}^N$, are built from N the pairs, where $S_i = \{x_i, x_1^+, x_2^+, ..., x_N^+\}$. In this expression, x_i is the query for S_i, x_i^+ is the positive example and x_j^+, $j \neq i$ are the negative examples. The corresponding $(N+1)$-tuplet loss, which referred to as the multi-class N-pair loss, is formulated as

$$L = \frac{1}{N} \sum_{i=1}^{N} \log \left[1 + \sum_{j \neq i} \exp(a_i^T n_j - a_i^T p_i)\right] \tag{6.3}$$

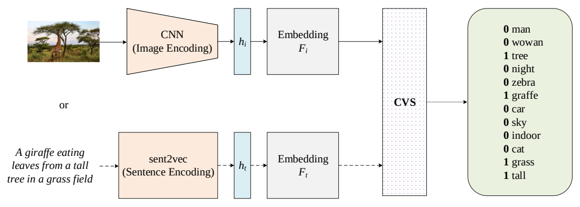
where a_i, p_i and $n_j, i \neq j$ are the network-defined embeddings of the anchor (x_i), positive (x_i^+) and negative (x_j^+) example respectively. The batch construction is designed to achieve the utmost potential of such $(N + 1)$-tuplet loss, when using deep CNNs as embedding kernel on large scale datasets both in terms of training data and number of output classes. Therefore, this framework consists of two indispensable components: the $(N+1)$-tuplet loss, as the building block loss function, and the N-pair construction, as the key to enable highly scalable training.

6.2.3 Applications of CVS

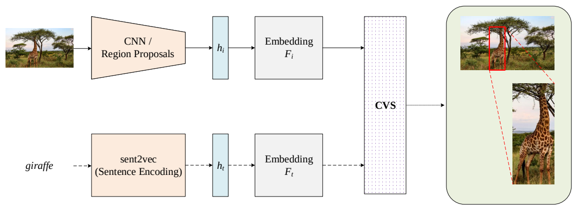
As an extension of this thesis, a lot of computer vision applications can be explored directly or indirectly supported by the concept of CVS, as shwon in Figure 6.4.

Classification

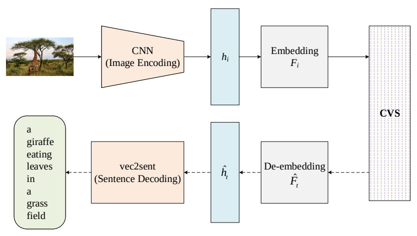
Classification models are trained to classify the input into a category. The models are also capable of zero-shot classification for categories that are not part of the training. This is an intrinsic advantage with CVS since it is trained in a multi-task manner across multiple datasets. For example, Figure 6.4(a) illustrates the classification inference using CVS model. Given an image or the language (paragraph, sentence, phrase, word, etc.) as input modality, the inference model encode the source into a vector in the trained CVS, which is linked to the list of categories. Note that this model is capable to classify the input into the category that is never seen in the training set, since the CVS is trained to learn semantics



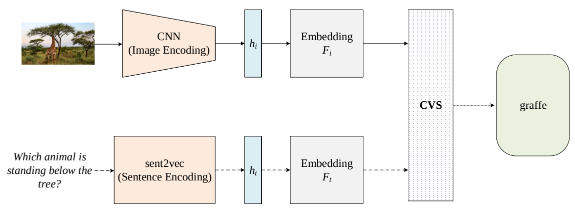
(a) Classification inference



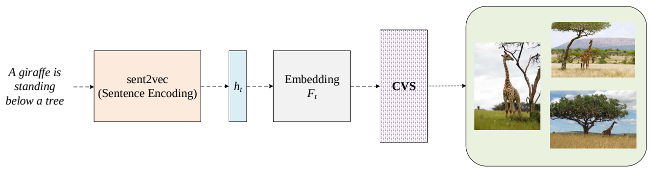
(b) Object localization inference



(c) Captioning inference



(d) VQA inference



(e) Cross-modal retrieval inference

Figure 6.4: Inference with the multi-task CVS model.

Object Localization

This involves localizing the object in the image with an object category or a natural language description, as shown in Figure 6.4(b). The input pair to the model is an image and the query text. The output is the bounding box of the region in the image that is most closely associated with the sentence. This model first extracts the information of objects and their spatial locations in an image. This is formulated through a bottom-up mechanism [149] to obtain a set of salient image regions each represented by a pooled convolutional feature vector. Practically, this bottom-up attention is achieved using faster R-CNN [143] to obtain the image region proposals. Once the image region proposals are obtained, the CVS model associates a query text (sentence/phrase/word) with the proposed locations, *i.e.*, the text is automatically aligned to different objects, that is represented by the extracted region proposals in the input image. An end-to-end trainable model for a similar task was presented in [145] for sentence generation and retrieval experiments.

Captioning

Image captioning task involves generating a natural language summary describing the image. Typically, the inputs to the captioning model is an image and the matching caption pair. The model makes use of a CNN which encodes the input image into a fixed dimensional vector, and uses this representation to decode it to the desired output sentence [134]. In other words, a pre-trained CNN is used for image feature extraction and an RNN to generate a sequence of words. Using CVS model, this goal can be achieved during inference by encoding the image into the vector in the trained CVS, and then decoding it into a sentence, as shown in Figure 6.4(c).

Visual Question Answering

The Visual Question Answering (VQA) model needs to answer text-based questions about images. The input pair consists of an image and corresponding question. Two parallel paths deal with image and question encoding, as shown in Figure 6.4(d). Since the model is trained to make the embedded vectors in the same space, the visual and text path end up with a probability distribution over a set of possible answers, which is in the form of the class label generated by a classifier, or an a sentence decoded by an RNN-based decoder.

Cross-modal Retrieval

Since the core of CVS is to learn a common subspace where items of different modalities can be directly compared to each other, a natural application is cross modal retrieval [147]. While retrieving from the same modality, the model translated the input to CVS and outputs the top results from the same modality from the test set. For example, providing an image, the single model retrieval finds out the similar images in the library according to its content. In the similar way, the CVS model is used to perform multi-modal retrieval as as shown in Figure 6.4(e). For cross-modal retrieval, the input modality is encoded and translated into CVS, and the output is generated from a different modality with a proper generator. As an example, the task can be sorting a set of images based on the relevance with the input sentence. Like the classification inference, this model is able to deal with zero-shot retrieval, that the test categories are not part of training.

Bibliography

[1] J. Lim and B. Han, "Generalized background subtraction using superpixels with label integrated motion estimation," in *European Conference on Computer Vision*, pp. 173–187, Springer, 2014.

[2] D. Giordano, F. Murabito, S. Palazzo, and C. Spampinato, "Superpixel-based video object segmentation using perceptual organization and location prior," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4814–4822, 2015.

[3] A. Khoreva, F. Galasso, M. Hein, and B. Schiele, "Classifier based graph construction for video segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 951–960, 2015.

[4] J. Papon, A. Abramov, M. Schoeler, and F. Worgotter, "Voxel cloud connectivity segmentation-supervoxels for point clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2027–2034, 2013.

[5] E. Trulls, S. Tsogkas, I. Kokkinos, A. Sanfeliu, and F. Moreno-Noguer, "Segmentation-aware deformable part models," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 168–175, 2014.

[6] P. Neubert and P. Protzel, "Compact watershed and preemptive slic: On improving trade-offs of superpixel segmentation algorithms," in *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pp. 996–1001, IEEE, 2014.

[7] F. Galasso, M. Keuper, T. Brox, and B. Schiele, "Spectral graph reduction for efficient image and streaming video segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 49–56, 2014.

[8] L. Fan and A. C. Loui, "A graph-based framework for video object segmentation and extraction in feature space," in *Multimedia (ISM), 2015 IEEE International Symposium on*, pp. 266–271, IEEE, 2015.

[9] C. Li, L. Lin, W. Zuo, S. Yan, and J. Tang, "Sold: sub-optimal low-rank decomposition for efficient video segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5519–5527, 2015.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[11] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[12] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *arXiv preprint arXiv:1505.00387*, 2015.

[13] J. G. Zilly, R. K. Srivastava, J. Koutník, and J. Schmidhuber, "Recurrent highway networks," *arXiv preprint arXiv:1607.03474*, 2016.

[14] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, pp. 448–456, 2015.

[15] C. Laurent, G. Pereyra, P. Brakel, Y. Zhang, and Y. Bengio, "Batch normalized recurrent neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pp. 2657–2661, IEEE, 2016.

[16] T. Cooijmans, N. Ballas, C. Laurent, Ç. Gülçehre, and A. Courville, "Recurrent batch normalization," *arXiv preprint arXiv:1603.09025*, 2016.

[17] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *European Conference on Computer Vision*, pp. 646–661, Springer, 2016.

[18] J. Moniz and C. Pal, "Convolutional residual memory networks," *arXiv preprint arXiv:1606.05262*, 2016.

[19] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Advances in Neural Information Processing Systems*, pp. 901–909, 2016.

[20] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[21] 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.

[22] L. Yao, A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, and A. Courville, "Describing videos by exploiting temporal structure," in *Proceedings of the IEEE international conference on computer vision*, pp. 4507–4515, 2015.

[23] S. Venugopalan *et al.*, "Sequence to sequence – video to text," in *ICCV*, 2015.

[24] T. Mikolov *et al.*, "Distributed representations of words and phrases and their compositionality," in *NIPS*, pp. 3111–3119, 2013.

[25] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *EMNLP*, pp. 1532–1543, 2014.

[26] K. Cho *et al.*, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[27] Q. V. Le and T. Mikolov, “Distributed representations of sentences and documents.,” in *ICML*, vol. 14, pp. 1188–1196, 2014.

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

[29] H. Zhao, Z. Lu, and P. Poupart, “Self-adaptive hierarchical sentence model,” *arXiv preprint arXiv:1504.05070*, 2015.

[30] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler, “Skip-thought vectors,” in *Advances in neural information processing systems*, pp. 3294–3302, 2015.

[31] S. Banerjee and A. Lavie, “Meteor: An automatic metric for mt evaluation with improved correlation with human judgments,” in *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, vol. 29, pp. 65–72, 2005.

[32] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in *European conference on computer vision*, pp. 740–755, Springer, 2014.

[33] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, “Long-term recurrent convolutional networks for visual recognition and description,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2625–2634, 2015.

[34] X. Chen and C. Lawrence Zitnick, “Mind’s eye: A recurrent visual representation for image caption generation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2422–2431, 2015.

[35] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3128–3137, 2015.

[36] 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 *International conference on machine learning*, pp. 2048–2057, 2015.

[37] A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski, "Plug & play generative networks: Conditional iterative generation of images in latent space.," in *CVPR*, vol. 2, p. 7, 2017.

[38] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," *arXiv preprint arXiv:1605.05396*, 2016.

[39] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 427–436, 2015.

[40] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune, "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks," in *Advances in Neural Information Processing Systems*, pp. 3387–3395, 2016.

[41] 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*, pp. 2234–2242, 2016.

[42] F. Galasso, R. Cipolla, and B. Schiele, "Video segmentation with superpixels.," in *ACCV (1)*, pp. 760–774, 2012.

[43] G. Palou and P. Salembier, "Hierarchical video representation with trajectory binary partition tree," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2099–2106, 2013.

[44] C. Xu, S. Whitt, and J. J. Corso, "Flattening supervoxel hierarchies by the uniform entropy slice," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2240–2247, 2013.

[45] M. Maire and S. X. Yu, "Progressive multigrid eigensolvers for multiscale spectral segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2184–2191, 2013.

[46] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille, "The role of context for object detection and semantic segmentation in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 891–898, 2014.

[47] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1915–1929, 2013.

[48] P. Pinheiro and R. Collobert, "Recurrent convolutional neural networks for scene labeling," in *International Conference on Machine Learning*, pp. 82–90, 2014.

[49] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440, 2015.

[50] M. Cimpoi, S. Maji, and A. Vedaldi, "Deep filter banks for texture recognition and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3828–3836, 2015.

[51] A. Sharma, O. Tuzel, and D. W. Jacobs, "Deep hierarchical parsing for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 530–538, 2015.

[52] D. Teney, M. Brown, D. Kit, and P. Hall, "Learning similarity metrics for dynamic scene segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2084–2093, 2015.

[53] S. D. Jain and K. Grauman, "Supervoxel-consistent foreground propagation in video.," in *ECCV (4)*, pp. 656–671, 2014.

[54] D. Zhang, O. Javed, and M. Shah, "Video object co-segmentation by regulated maximum weight cliques," in *ECCV (7)*, pp. 551–566, 2014.

[55] H. Fu, D. Xu, B. Zhang, S. Lin, and R. K. Ward, "Object-based multiple foreground video co-segmentation via multi-state selection graph," *IEEE Transactions on Image Processing*, vol. 24, no. 11, pp. 3415–3424, 2015.

[56] L. Wang, G. Hua, R. Sukthankar, J. Xue, Z. Niu, and N. Zheng, "Video object discovery and co-segmentation with extremely weak supervision," *IEEE transactions on pattern analysis and machine intelligence*, 2016.

[57] D. Zhang, O. Javed, and M. Shah, "Video object segmentation through spatially accurate and temporally dense extraction of primary object regions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 628–635, 2013.

[58] A. Papazoglou and V. Ferrari, "Fast object segmentation in unconstrained video," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1777–1784, 2013.

[59] Y. Zhang, X. Chen, J. Li, C. Wang, and C. Xia, "Semantic object segmentation via detection in weakly labeled video," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3641–3649, 2015.

[60] W. Zhang, S. Zeng, D. Wang, and X. Xue, "Weakly supervised semantic segmentation for social images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2718–2726, 2015.

[61] C. Zhang and A. Loui, "A coarse-to-fine framework for video object segmentation," in *IS&T International Symposium on Electronic Imaging*, Society for Imaging Science and Technology, 2017.

[62] A. Khoreva, F. Galasso, M. Hein, and B. Schiele, "Learning must-link constraints for video segmentation based on spectral clustering," in *GCPR*, pp. 701–712, 2014.

[63] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 5, pp. 603–619, 2002.

[64] B. D. Lucas, T. Kanade, *et al.*, "An iterative image registration technique with an application to stereo vision," 1981.

[65] Z. Kalal, K. Mikolajczyk, and J. Matas, "Forward-backward error: Automatic detection of tracking failures," in *Pattern recognition (ICPR), 2010 20th international conference on*, pp. 2756–2759, IEEE, 2010.

[66] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 11, pp. 2765–2781, 2013.

[67] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

[68] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Computer Vision, 1998. Sixth International Conference on*, pp. 839–846, IEEE, 1998.

[69] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International journal of computer vision*, vol. 59, no. 2, pp. 167–181, 2004.

[70] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut: Interactive foreground extraction using iterated graph cuts," in *ACM transactions on graphics (TOG)*, vol. 23, pp. 309–314, ACM, 2004.

[71] D. Tsai, M. Flagg, and J. M.Rehg, "Motion coherent tracking with multi-label mrf optimization," *BMVC*, 2010.

[72] P. Chockalingam, N. Pradeep, and S. Birchfield, "Adaptive fragments-based tracking of non-rigid objects using level sets," in *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 1530–1537, IEEE, 2009.

[73] Y. J. Lee, J. Kim, and K. Grauman, "Key-segments for video object segmentation," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 1995–2002, IEEE, 2011.

[74] T. Ma and L. J. Latecki, "Maximum weight cliques with mutex constraints for video object segmentation," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 670–677, IEEE, 2012.

[75] M. Ristin, J. Gall, M. Guillaumin, and L. Van Gool, "From categories to subcategories: large-scale image classification with partial class label refinement," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 231–239, 2015.

[76] J. Lu, G. Wang, W. Deng, P. Moulin, and J. Zhou, "Multi-manifold deep metric learning for image set classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1137–1145, 2015.

[77] T. Hyun Kim and K. Mu Lee, "Generalized video deblurring for dynamic scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5426–5434, 2015.

[78] S. Su and W. Heidrich, "Rolling shutter motion deblurring," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1529–1537, 2015.

[79] H. Zhang and J. Yang, "Intra-frame deblurring by leveraging inter-frame camera motion," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4036–4044, 2015.

[80] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, "How to construct deep recurrent neural networks," *arXiv preprint arXiv:1312.6026*, 2013.

[81] Y. Zhang, G. Chen, D. Yu, K. Yaco, S. Khudanpur, and J. Glass, "Highway long short-term memory rnns for distant speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pp. 5755–5759, IEEE, 2016.

[82] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush, "Character-aware neural language models," *arXiv preprint arXiv:1508.06615*, 2015.

[83] S. Gerschgorin, "Über die abgrenzung der eigenwerte einer matrix," *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques*, pp. 749–754, 1931.

[84] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 2342–2350, 2015.

[85] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE transactions on neural networks and learning systems*, 2016.

[86] X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollár, and C. L. Zitnick, "Microsoft coco captions: Data collection and evaluation server," *arXiv preprint arXiv:1504.00325*, 2015.

[87] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting on association for computational linguistics*, pp. 311–318, Association for Computational Linguistics, 2002.

[88] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text summarization branches out: Proceedings of the ACL-04 workshop*, vol. 8, Barcelona, Spain, 2004.

[89] R. Vedantam, C. Lawrence Zitnick, and D. Parikh, "Cider: Consensus-based image description evaluation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4566–4575, 2015.

[90] L. Baraldi, C. Grana, and R. Cucchiara, "Hierarchical boundary-aware neural encoder for video captioning," *arXiv preprint arXiv:1611.09312*, 2016.

[91] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.

[92] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[93] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, “Show and tell: Lessons learned from the 2015 mscoco image captioning challenge,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 4, pp. 652–663, 2017.

[94] J. Choi *et al.*, “Textually customized video summaries,” *arXiv preprint arXiv:1702.01528*, 2017.

[95] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Advances in neural information processing systems*, pp. 3104–3112, 2014.

[96] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko, “Sequence to sequence-video to text,” in *Proceedings of the IEEE international conference on computer vision*, pp. 4534–4542, 2015.

[97] H. Yu *et al.*, “Video paragraph captioning using hierarchical recurrent neural networks,” in *CVPR*, pp. 4584–4593, 2016.

[98] A. Shin *et al.*, “Beyond caption to narrative: Video captioning with multiple sentences,” in *ICIP*, pp. 3364–3368, IEEE, 2016.

[99] J. Chung *et al.*, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” *arXiv preprint arXiv:1412.3555*, 2014.

[100] C. Zhang *et al.*, “Batch normalized recurrent highway networks,” in *ICIP*, IEEE, 2017.

[101] S. Jean, K. Cho, R. Memisevic, and Y. Bengio, “On using very large target vocabulary for neural machine translation,” *arXiv preprint arXiv:1412.2007*, 2014.

[102] P. Pan *et al.*, “Hierarchical recurrent neural encoder for video representation with application to captioning,” in *CVPR*, pp. 1029–1038, 2016.

[103] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *arXiv preprint arXiv:1409.0473*, 2014.

[104] J. Xu *et al.*, "Msr-vtt: A large video description dataset for bridging video and language," in *CVPR*, 2016.

[105] D. L. Chen and W. B. Dolan, "Collecting highly parallel data for paraphrase evaluation," in *Proceedings of the 49th Association for Computational Linguistics: Human Language Technologies-Volume 1*, pp. 190–200, ACL, 2011.

[106] P. Young *et al.*, "From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions," *Transactions of the Association for Computational Linguistics*, 2014.

[107] M. Marelli *et al.*, "A sick cure for the evaluation of compositional distributional semantic models.," in *LREC*, pp. 216–223, 2014.

[108] A. Senina, M. Rohrbach, W. Qiu, A. Friedrich, S. Amin, M. Andriluka, M. Pinkal, and B. Schiele, "Coherent multi-sentence video description with variable level of detail," *arXiv preprint arXiv:1403.6173*, 2014.

[109] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," *arXiv preprint arXiv:1503.00075*, 2015.

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

[111] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

[112] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.

[113] A. Gordo, J. Almazan, J. Revaud, and D. Larlus, "End-to-end learning of deep visual representations for image retrieval," *International Journal of Computer Vision*, vol. 124, no. 2, pp. 237–254, 2017.

[114] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," *arXiv preprint arXiv:1703.07737*, 2017.

[115] R. Socher, M. Ganjoo, C. D. Manning, and A. Ng, "Zero-shot learning through cross-modal transfer," in *Advances in neural information processing systems*, pp. 935–943, 2013.

[116] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 689–696, 2011.

[117] N. Srivastava and R. R. Salakhutdinov, "Multimodal learning with deep boltzmann machines," in *Advances in neural information processing systems*, pp. 2222–2230, 2012.

[118] K. Sohn, W. Shang, and H. Lee, "Improved multimodal deep learning with variation of information," in *Advances in Neural Information Processing Systems*, pp. 2141–2149, 2014.

[119] S. Reed, Z. Akata, H. Lee, and B. Schiele, "Learning deep representations of fine-grained visual descriptions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 49–58, 2016.

[120] K. Sohn, "Improved deep metric learning with multi-class n-pair loss objective," in *Advances in Neural Information Processing Systems*, pp. 1857–1865, 2016.

[121] A. Eisenschtat and L. Wolf, "Linking image and text with 2-way nets," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[122] L. Wang, Y. Li, and S. Lazebnik, "Learning deep structure-preserving image-text embeddings," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5005–5013, 2016.

[123] L. Wu, A. Fisch, S. Chopra, K. Adams, A. Bordes, and J. Weston, "Starspace: Embed all the things!," *arXiv preprint arXiv:1709.03856*, 2017.

[124] 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*, pp. 2672–2680, 2014.

[125] 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.

[126] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.(CVPR)*, pp. 1–13, 2018.

[127] X. Liang, Z. Hu, H. Zhang, C. Gan, and E. P. Xing, "Recurrent topic-transition gan for visual paragraph generation," *arXiv preprint arXiv:1703.07022*, 2017.

[128] B. McCann, J. Bradbury, C. Xiong, and R. Socher, "Towards the imagenet-cnn of nlp: Pretraining sentence encoders with machine translation," in *Advances in Neural Information Processing Systems*, pp. 6285–6296, 2017.

[129] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Lawrence Zitnick, and D. Parikh, "Vqa: Visual question answering," in *Proceedings of the IEEE international conference on computer vision*, pp. 2425–2433, 2015.

[130] M. Malinowski, M. Rohrbach, and M. Fritz, "Ask your neurons: A deep learning approach to visual question answering," *International Journal of Computer Vision*, vol. 125, no. 1-3, pp. 110–135, 2017.

[131] R. Krishna, K. Hata, F. Ren, L. Fei-Fei, and J. C. Niebles, “Dense-captioning events in videos.,” in *ICCV*, pp. 706–715, 2017.

[132] A. Prakash, S. A. Hasan, K. Lee, V. Datla, A. Qadir, J. Liu, and O. Farri, “Neural paraphrase generation with stacked residual lstm networks,” *arXiv preprint arXiv:1610.03098*, 2016.

[133] A. Gupta, A. Agarwal, P. Singh, and P. Rai, “A deep generative framework for paraphrase generation,” *arXiv preprint arXiv:1709.05074*, 2017.

[134] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, “Show and tell: A neural image caption generator,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3156–3164, 2015.

[135] A. Dosovitskiy and T. Brox, “Generating images with perceptual similarity metrics based on deep networks,” in *Advances in Neural Information Processing Systems*, pp. 658–666, 2016.

[136] C. Zhang *et al.*, “Semantic sentence embeddings for paraphrasing and text summarization,” in *GlobalSIP*, 2017.

[137] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[138] A. Dosovitskiy and T. Brox, “Inverting convolutional networks with convolutional networks,” *CoRR abs/1506.02753*, 2015.

[139] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, “Imagenet large scale visual recognition challenge,” *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[140] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," *arXiv preprint*, 2017.

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

[142] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks.," in *CVPR*, vol. 1, p. 3, 2017.

[143] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, pp. 91–99, 2015.

[144] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[145] J. Johnson, A. Karpathy, and L. Fei-Fei, "Densecap: Fully convolutional localization networks for dense captioning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4565–4574, 2016.

[146] K. Wang, Q. Yin, W. Wang, S. Wu, and L. Wang, "A comprehensive survey on cross-modal retrieval," *arXiv preprint arXiv:1607.06215*, 2016.

[147] B. Wang, Y. Yang, X. Xu, A. Hanjalic, and H. T. Shen, "Adversarial cross-modal retrieval," in *Proceedings of the 2017 ACM on Multimedia Conference*, pp. 154–162, ACM, 2017.

[148] K. Kafle and C. Kanan, "Visual question answering: Datasets, algorithms, and future challenges," *Computer Vision and Image Understanding*, vol. 163, pp. 3–20, 2017.

[149] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang, "Bottom-up and top-down attention for image captioning and visual question answering," in *CVPR*, vol. 3, p. 6, 2018.

[150] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese, "Deep metric learning via lifted structured feature embedding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4004–4012, 2016.

[151] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML Deep Learning Workshop*, vol. 2, 2015.

[152] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka, "Metric learning for large scale image classification: Generalizing to new classes at near-zero cost," in *Computer Vision–ECCV 2012*, pp. 488–501, Springer, 2012.

[153] M. Bucher, S. Herbin, and F. Jurie, "Improving semantic embedding consistency by metric learning for zero-shot classiffication," in *European Conference on Computer Vision*, pp. 730–746, Springer, 2016.