

UNIVERSITY OF TECHNOLOGY SYDNEY  
Faculty of Engineering and Information Technology

**Video Representation Learning with  
Deep Neural Networks**

by

**Linchao Zhu**

A THESIS SUBMITTED  
IN PARTIAL FULFILLMENT OF THE  
REQUIREMENTS FOR THE DEGREE

**Doctor of Philosophy**

Sydney, Australia

2019



## Certificate of Authorship/Originality

I certify that the work in this thesis has not been previously submitted for a degree nor has it been submitted as a part of the requirements for other degree except as fully acknowledged within the text.

I also certify that this thesis has been written by me. Any help that I have received in my research and in the preparation of the thesis itself has been fully acknowledged. In addition, I certify that all information sources and literature used are quoted in the thesis.

This research is supported by the Australian Government Research Training Program.

Linchao Zhu

Production Note:  
Signature removed  
prior to publication.

# ABSTRACT

## Video Representation Learning with Deep Neural Networks

by

Linchao Zhu

Despite the recent success of neural networks in image feature learning, a major problem in the video domain is the lack of sufficient labeled data for learning to model temporal information. One method to learn a video representation from untrimmed videos is to perform unsupervised temporal modeling. Given a clip sampled from a video, its past and future neighboring clips are used as temporal context, and reconstruct the two temporal transitions, i.e., present→past transition and present→future transition, which reflect the temporal information in different views. In this thesis, the two transitions are exploited simultaneously by incorporating a bi-direction reconstruction which consists of a backward reconstruction and a forward reconstruction. To adapt an existing model to recognize a new category which was unseen during training, it may be necessary to manually collect hundreds of new training samples. Such a procedure is rather tedious and labor intensive, especially when there are many new categories. In this thesis, a classification model is proposed to learn from a few examples in a life-long manner. To evaluate the effectiveness of the learned representation, extensive experiments are conducted on multimedia event detection, image classification, video captioning, and video question answering.

Dissertation directed by Professor Yi Yang

Centre for Artificial Intelligence, School of Software

## Acknowledgements

First and foremost, I would like to thank my supervisor Professor Yi Yang. I am extremely grateful for his patience and support. He guided me on any research directions I was excited about. He also provided tremendous help on building up my research career, and offered great kindness to my personal life. He also taught me how to work with colleagues, which is valuable to my future career. Thanks to Prof Alexander G. Hauptmann, my advisor when I visited Carnegie Mellon University, from whom I learned how to do research for real-world applications. Thanks to Heng, Du, and Laura, my supervisors when I interned at Facebook Research, from whom I learned critical thinking and how to perform research in a systematic way.

I would also like to thank my colleagues at University of Technology Sydney. I would like to thank Xiaojun Chang, Xuanyi Dong, Hehe Fan, Qianyu Feng, Qingji Guan, Yang He, Wenhe Liu, Yanbin Liu, Ping Liu, Yutian Lin, Peike Li, Fan Ma, Jiaxu Miao, Pingbo Pan, Yu Wu, Xiaohan Wang, Zhongwen Xu, Yan Yan, Zongxin Yang, Fengda Zhu, Hu Zhang, Zhong Zhun, Zhedong Zheng, Liang Zheng, Xiaolin Zhang, and many others. I was really fortunate to work with them and participate in intellectual conversations with them.

I would also like to thank Data to Decision CRC for supporting my research.

Lastly I would like to thank my mother Supin Chen and Qiming Zhu for their support and love throughout the years.

Linchao Zhu  
Sydney, Australia, 2019.

# List of Publications

## Journal Papers

- J-1. **Zhu, L.**, Xu, Z., Yang, Y. and Hauptmann, A.G., 2017. Uncovering the temporal context for video question answering. *International Journal of Computer Vision*, 124(3), pp.409-421.
- J-2. Gan, C., Yang, Y., **Zhu, L.**, Zhao, D. and Zhuang, Y., 2016. Recognizing an action using its name: A knowledge-based approach. *International Journal of Computer Vision*, 120(1), pp.61-77.

## Conference Papers

- C-1. **Zhu, L.**, Xu, Z. and Yang, Y., 2017, July. Bidirectional Multirate Reconstruction for Temporal Modeling in Videos. In *Computer Vision and Pattern Recognition (CVPR)*, 2017 IEEE Conference on (pp. 1339-1348). IEEE. **Spotlight.**
- C-2. **Zhu, L.** and Yang, Y., 2018, September. Compound Memory Networks for Few-Shot Video Classification. In *European Conference on Computer Vision* (pp. 782-797). Springer, Cham.
- C-3. **Zhu, L.\***, Xu, Z.\*, and Yang, Y., 2017, July. Few-Shot Object Recognition from Machine-Labeled Web Images. In *Computer Vision and Pattern Recognition (CVPR)*, 2017 IEEE Conference on (pp. 5358-5366). IEEE. **Spotlight.**  
(\* indicates equal contribution)
- C-4. Fan, H., **Zhu, L.** and Yang, Y., 2019. Cubic LSTMs for Video Prediction. In *The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI)*.
- C-5. Fan, H., Xu, Z., **Zhu, L.**, Yan, C., Ge, J. and Yang, Y., 2018. Watching a Small Portion could be as Good as Watching All: Towards Efficient Video

- Classification. In *International Joint Conference on Artificial Intelligence (IJ-CAI)* (Vol. 2, No. 5, p. 6).
- C-6. Wu, Y., **Zhu, L.**, Jiang, L. and Yang, Y., 2018. Decoupled Novel Object Captioner. In 2018 *ACM Multimedia Conference on Multimedia Conference* (pp. 1029-1037). ACM.
- C-7. Dong, X., **Zhu, L.**, Zhang, D., Yang, Y. and Wu, F., 2018, October. Fast Parameter Adaptation for Few-shot Image Captioning and Visual Question Answering. In 2018 *ACM Multimedia Conference on Multimedia Conference* (pp. 54-62). ACM.

# Contents

Certificate	iii
Abstract	iv
Acknowledgments	v
List of Publications	vi
List of Figures	xii
<b>1 Introduction</b>	<b>1</b>
1.1 Video Feature Learning . . . . .	1
1.2 Video and Language . . . . .	2
1.3 Contributions . . . . .	3
<b>2 Literature Review</b>	<b>5</b>
2.1 Video Classification . . . . .	5
2.1.1 Convolutional Networks for Video Classification . . . . .	6
2.1.2 Recurrent Networks for Video Classification . . . . .	7
2.2 Bridging Vision and Language . . . . .	8
2.2.1 Video Captioning . . . . .	8
2.2.2 Video Question Answering . . . . .	8
2.3 Few-shot Video Classification . . . . .	10
2.3.1 Memory-Augmented Neural Networks . . . . .	11
<b>3 Bidirectional Multirate Reconstruction for Temporal Mod-</b>	



<b>eling in Videos</b>	<b>13</b>
3.1 Introduction . . . . .	13
3.2 Multirate Visual Recurrent Models . . . . .	15
3.2.1 Multirate Gated Recurrent Unit . . . . .	15
3.2.2 Unsupervised Video Sequence Reconstruction . . . . .	18
3.2.3 Complex Event Detection . . . . .	21
3.2.4 Video Captioning . . . . .	23
3.3 Results . . . . .	23
3.3.1 Complex Event Detection . . . . .	23
3.3.2 Video Captioning . . . . .	28
3.4 Conclusion . . . . .	32
<b>4 Uncovering the Temporal Context for Video Question Answering</b>	<b>34</b>
4.1 Introduction . . . . .	34
4.2 Dataset Collection and Task Definitions . . . . .	37
4.2.1 Dataset and QA Pair Generation . . . . .	38
4.2.2 Task Definitions and Analysis . . . . .	40
4.3 The Proposed Approach . . . . .	42
4.3.1 Learning to Represent Video Sequences . . . . .	44
4.3.2 Dual-Channel Learning to Rank . . . . .	48
4.4 Results . . . . .	51
4.4.1 Evaluation of Describing the Present . . . . .	51
4.4.2 Evaluation of Inferring the Past and Predicting the Future . . . . .	56
4.4.3 Limitations and Future Work . . . . .	57

4.5 Conclusion . . . . .	58
<b>5 Few-Shot Object Recognition from Machine-Labeled Web Images</b>	<b>60</b>
5.1 Introduction . . . . .	60
5.2 Proposed Approach . . . . .	64
5.2.1 Preliminaries . . . . .	64
5.2.2 Model Overview . . . . .	65
5.2.3 Model Components . . . . .	68
5.2.4 Training . . . . .	72
5.2.5 Inference . . . . .	72
5.3 Experiments . . . . .	72
5.3.1 Preprocessing . . . . .	73
5.3.2 Model Specifications . . . . .	73
5.3.3 Datasets . . . . .	74
5.3.4 Few-shot Learning with Human-labeled annotations . . . . .	74
5.3.5 Few-shot Learning with Machine-labeled Annotations . . . . .	77
5.3.6 Hyperparamter Study . . . . .	79
5.4 Conclusion . . . . .	80
<b>6 Compound Memory Networks for Few-shot Video Classification</b>	<b>81</b>
6.1 Introduction . . . . .	81
6.2 Few-shot Video Classification Setup . . . . .	84
6.3 Compound Memory Network . . . . .	85
6.3.1 Multi-saliency Embedding Function . . . . .	86

6.3.2	Compound Memory Structure . . . . .	87
6.3.3	Training . . . . .	91
6.4	Experiments . . . . .	92
6.4.1	Datasets . . . . .	92
6.4.2	Implementation Details . . . . .	92
6.4.3	Evaluation . . . . .	94
6.4.4	Ablation Study . . . . .	98
6.5	Conclusion . . . . .	99
<b>7</b>	<b>Future Works</b>	<b>100</b>
	<b>Bibliography</b>	<b>101</b>

## List of Figures

- 3.1 Frame sampling rate should vary in accordance with different motion speed. In this example, only the last three frames have fast motion. The dashed arrow corresponds to a fixed sampling rate, while the solid arrow corresponds to multiple rates. . . . . 14
- 3.2 We illustrate the two modes in the mGRU. In the slow to fast mode, the state matrices  $\mathbf{V}_*$  are block upper-triangular matrices and in the fast to slow mode, they are block lower-triangular matrices. . . . . 17
- 3.3 Unrolled mGRU. In the example, the state is divided into three groups and the slow to fast mode is shown. At each step  $t$ , groups satisfying  $(t \text{ MOD } T_i) = 0$  are activated (cells with black border). For example, at step 2, group 1 and group 2 are activated. The activated groups take the frame input and previous states to calculate the next states. For those that are inactivated, we simply pass the previous states to the next step. Group 1 is the fastest and group 3 is the slowest with larger  $T_i$ . The slow to fast mode is the mode by which the slower groups pass the states to the faster groups. 20
- 3.4 The model architecture of unsupervised video representation learning. In this model, two decoders are used to predict surrounding contexts by reconstructing previous frames and next frame sequences. The “<G0>” input, which is a zero vector, is used at step 0 in the decoder. During training, one of the two decoders is used with a probability of 0.5 for reconstruction. . . . . 22

4.1	Questions and answers about the past, the present and the future. Our system includes three subtasks, which infer the <i>past</i> , describe the <i>present</i> , and predict the <i>future</i> , while <i>only the current frames are observable</i> . Best viewed in color . . . . .	35
4.2	t-SNE visualization of word embeddings for each category learned from word2vec model. . . . .	37
4.3	Examples of QA pairs for different categories and levels of difficulty. The words colored in green are correct answers, and the difficult candidates are marked in red. . . . .	41
4.4	Distribution of question types for each dataset . . . . .	44
4.5	Distribution of question lengths for each dataset . . . . .	45
4.6	Distribution of answer lengths for each dataset . . . . .	46
4.7	The encoder-decoder model (top): encoder state of last time step is passed to three decoders for reconstruction. Learn to answer questions (bottom): encoder state of last time step is passed to the ranking module which selects an answer based on the visual information . . . . .	47
4.8	Illustration of dual-channel learning to rank . . . . .	49
4.9	The effectiveness of dual-channel learning to rank. We conduct experiments on the <i>Present-Easy</i> task to showcase. $\lambda = 0$ corresponds to using the sentence channel only and $\lambda = 1$ corresponds to using the word channel only . . . . .	55
4.10	Example results obtained from our model. Each candidate has a score corresponding to a clip. Correct answers are marked in green while failed cases are in red . . . . .	56

5.1	Given a large vocabulary of labels and their corresponding images, we conduct few-shot learning on a novel category which is not in the vocabulary and only has a handful of positive examples. The image examples in the vocabulary are stored in the external memory of our model, and the image example from the novel category queries the external memory. Our model returns helpful information according to visual similarity and LSTM controllers. The retrieved information, i.e., visual features and their corresponding labels, are combined to classify this query image example. . . . .	62
5.2	An illustration of our proposed model. Best viewed in color. . . . .	65
5.3	Sample images from the OpenImages dataset. Annotations on the images are shown in the bottom. The annotations listed are “label id”, “label name”, “confidence” tuples. . . . .	75
5.4	We show the query results returns from the external memory. The scores are the softmax probabilities. Only top-3 results are shown. . .	78
6.1	The setting of the few-shot video classification. There are two non-overlapping datasets in this figure, i.e., meta-training and meta-testing. The meta-training set is for meta-learning and the meta-testing set is for evaluating the generalization performance on novel categories. The network is trained in an episodic way and each episode has a support set and a query example. . . . .	82
6.2	Illustration of the input embedding model. The embedding function generates the multi-saliency descriptor $\mathbf{Q}$ , which is flattened and normalized to a query vector. . . . .	87

6.3	Our CMN structure. A video is first mapped to a matrix representation via the multi-saliency embedding function. This hidden representation is then vectorized and normalized as a query vector, which performs a nearest neighbour search over the abstract memory. The most similar memory slot is retrieved and the label stored in the value memory will be used as the prediction. The constituent key memory contains the matrix representations of the inputs, while the abstract memory is constructed on top of the stacked constituent keys. . . . .	88
6.4	Illustration of the update rule for CMN. . . . .	90
6.5	Per class accuracy on the 5-way 1-shot setting. We show the accuracies of 24 classes on the meta-testing set. . . . .	95
6.6	We illustrate the inference procedure. There are 5 classes and the memory has 16 slots. Two different update rules will be used depending on the query results. . . . .	97

# Chapter 1

## Introduction

### 1.1 Video Feature Learning

Temporal information plays a key role in video representation modeling. In earlier years, hand-crafted features, e.g., Dense Trajectories (DT) and improved Dense Trajectories (iDT) [118, 119], use local descriptors along trajectories to model video motion structures. Despite achieving promising performance, DT and iDT are very expensive to extract, due to the heavy computational cost of optical flows and it takes about a week to extract iDT features for 8,000 hours of web videos using 1,000 CPU cores [128]. Deep visual features have achieved significantly better performance in image classification and detection tasks than hand-crafted features at an efficient processing speed [54, 35, 29]. However, learning a video representation on top of deep Convolutional Neural Networks (ConvNets) remains a challenging problem. Two-stream ConvNet [91] is groundbreaking in learning video motion structures over short video clips. Although it achieves comparable performance to iDT for temporally trimmed videos, two-stream ConvNet still needs to extract optical flows. The heavy cost severely limits the utility of methods based on optical flows, especially in the case of large scale video data.

Extending 2D ConvNet to 3D, C3D ConvNet has been demonstrated to be effective for spatio-temporal modeling and it avoids extracting optical flows. However, it can only model temporal information in short videos, usually of 16 frames [104]. Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) [37, 72] and a modified Hierarchical Recurrent Neural Encoder (HRNE) [74], have been



used to model temporal information in videos.

Notwithstanding the appealing ability of end-to-end approaches for learning a discriminative feature, such approaches require a large amount of labeled data to achieve good performance with plausible generalization capabilities. Compared to images, a large number of videos are very expensive to label by humans. For example, the largest public human-labeled video dataset (ActivityNet) [23] only has 20,000 labeled videos while the ImageNet dataset has over one million labeled instances [88]. Temporal ConvNet trained on the UCF-101 dataset [94] with about 10,000 temporally trimmed videos did not generalize well on temporally a untrimmed dataset [130]. Targeting short video clips, Srivastava et al. [96] proposed training a composite autoencoder in an unsupervised manner to learn video temporal structures, essentially by predicting future frames and reconstructing present frames.

## 1.2 Video and Language

Apart from learning video representation using neural networks, there is increasing interest in achieving deeper understanding of visual content by jointly modeling images and natural language. We propose a video question answering by uncovering video temporal context. As Convolutional Neural Networks (ConvNets) have raised the bar in image classification and detection tasks [30, 39, 99], RNNs, particularly LSTM [37], also play a key role in visual description tasks, such as image captioning [20, 116, 127]. Image Question Answering (Image QA), which is one step beyond image captioning and requires an extra layer of interaction between humans and computers, has started to attract research attention [4, 28, 65]. In the area of video analysis, a small number of systems have been proposed for video captioning [86, 111, 131, 74]. These methods have demonstrated promising performance in describing a video by a single short sentence. Similar to image captioning, video captioning may not be as intelligent as desired, especially when only a particular

section or object in the video [4] is concerned with. In addition, it lacks interaction between the computer and the user [28]. A MovieQA dataset [102] was released focusing on story comprehension based on both movie clips and texts. The dataset contains some questions regarding “Why” and “How” about something. It is a challenging task and can only be resolved by exploiting visual and textual information. However, the task of MovieQA is basically to answer questions about a movie clip without any video context information.

ConvNets and RNNs have become built-in modules in various fields. However, large amounts of labeled training data are required to train a deep neural network. To adapt an existing model to recognize a new category which was unseen during training, it may be necessary to manually collect hundreds of new training samples. Such a procedure is rather tedious and labor intensive, especially when there are many new categories. There is an increasing need to learn a classification model from a few examples in a life-long manner, which is also known as the few-shot learning task [89, 115]. In a few-shot recognition setting, the network needs to effectively learn classifiers for novel concepts from only a few examples. Unlike traditional models trained on many data samples, the model in a few-shot setting is trained to generalize across different episodes. In contrast to training new classifiers by fine-tuning, a learning to learn approach under the meta learning paradigm [89] can be used.

### 1.3 Contributions

This thesis is organised as follows. In Chapter 2, I present the related works on video representation learning. I cover recent study on video classification, video captioning, video question answering, few-shot classification. I also discuss the recent advances on memory-augmented neural networks and recurrent neural networks. In Chapter 3, a multi-rate encoder is proposed to leverage different temporal speeds.

The model is evaluated on video event detection and video captioning. In Chapter 4, I propose a video question answering task to answer questions about past and future. An unsupervised encoder-decoder is introduced to model temporal context and explore relationships between neighboring clips. In Chapter 5, when train with only limited number of images, I explore to use noisy labeled data for mining related knowledge from web data. In Chapter 6, I study the few-shot setting for video classification. A novel video representation based on key-value memory is proposed for few-shot classification. In Chapter 7, I briefly summarize the thesis and show the future directions for improvements.

In this thesis, I make the following contributions. First, I show bi-directional unsupervised pre-training is beneficial to video classification, video captioning, and video question answering. It indicates the importance of temporal context learning for videos. Second, to better model video temporal structure, I propose a multi-rate GRU for multi temporal scale video modeling. I show the effectiveness of modeling different speeds in both accuracy and efficiency. Third, effectively abstract knowledge from web data or episode input sequence is the key to the success of generalization from few examples. I propose an abstract memory for few-example image classification. Similarly, a video abstract memory compresses the multi-dimensional video data to a single vector for fast addressing. More exploration for better architectures of sequence data modeling will be made in the future.

## Chapter 2

### Literature Review

#### 2.1 Video Classification

Video classification methods have evolved from using hand-crafted features, e.g., improved dense trajectories [119], to deep models, e.g., two-stream Convolutional Neural Networks (ConvNets) [91, 121], 3D ConvNets [105], two-stream 3D ConvNets [11]. Recurrent Neural Networks have also been utilized to model video sequences [136, 140]. Many efforts have been made to train a video classification model using large amounts of video data, however, it would be expensive to collect large datasets and retrain the classifier for all novel categories. The few-shot video classification task is realistic in a real-world scenario, where the model will encounter novel categories that are never seen during training. The networks should be trained to adapt to new tasks.

Research efforts to improve visual representations for videos have been ongoing. Local features such as HOF [58] and MBH [18] extracted along spatio-temporal tracklets have been used as motion descriptors in the Dense Trajectories feature [118] and its variants [119]. However, it is notoriously inefficient to extract hand-crafted features like improved Dense Trajectories (iDT) [119, 128], mostly due to the dense sampling nature of local descriptors and the time-consuming extraction of optical flows. On the other hand, the classification performance of state-of-the-art hand-crafted features has been surpassed by many methods based on neural networks in web video classification and action recognition tasks [128, 120].

### 2.1.1 Convolutional Networks for Video Classification

Many video feature learning methods based on ConvNets have been proposed. In deep video representation learning, many research works focus on designing clip-level or frame-level representations for videos. One line of research builds upon two-stream ConvNets [91], which takes stacked optical flow images as inputs to a ConvNet. The optical flow ConvNet and RGB ConvNet prediction scores are then averaged to obtain the final prediction. The two-stream network achieved better performance than iDT on several action recognition datasets, which is later heavily used in the literature [121, 136, 11, 25, 126]. The idea has also successfully been used in other tasks, e.g., video captioning [111]. However, optical flow extraction is required in the two-stream network, which is an expensive process. Another line of research lies in leveraging 3D ConvNets for video representation learning. 3D ConvNets for human action recognition has been studied in [42, 104, 46]. A common practice is to stack continuous video frames and feed them to a ConvNet, where the 2D convolution is replaced by 3D convolution. However, compared to 2D ConvNets, the number of parameters of 3D ConvNets is relatively large when extending spatial filters to spatio-temporal ones, which could lead to over-fitting on small datasets. With the raise of large action recognition datasets like Kinetics [11], different variations of 3D ConvNets focus on reducing the computational cost and memory usage while improving the action recognition accuracy. For example, I3D [11] adapts the Inception-V1 architecture [100], inflating the 2D filter to 3D filters. Qiu et al. [80] replaced spatio-temporal 3D convolution with spatial and temporal convolutions using a residual connection style. [104] proposed 3D ConvNets which capture temporal dynamics in video clips without the very time-consuming optical flow extraction procedure. Tran et al. [106] decomposed spatial and temporal convolution with a (2+1)D block, which doubles the number of non-linearities. Xie et al. [126] used a similar approach which replaces some 3D convolutions with 2D convolutions. They

also used separable 3D convolution with fewer parameters, which is more computational efficient and achieved better accuracy. Zhou et al. [139] introduced a Mixed Convolutional Tube that adds the 3D convolution features map with 2D convolution features.

One way to use ConvNets for video classification is to perform temporal pooling over convolutional activations. Ng et al. [72] proposed learning a global video representation by using max pooling over the last convolutional layer across video frames. Wang et al. [120] aggregated ConvNet features along the tracklets obtained from iDT. Xu et al. [128] applied VLAD encoding [41] over ConvNet activations and found that the encoding methods are superior to mean pooling. The other common solution is to feed multiple frames as input to ConvNets. Karpathy et al. [46] proposed a convolutional temporal fusion network, but it is only marginally better than the single frame baseline.

### 2.1.2 Recurrent Networks for Video Classification

Ng et al. [72] and Donahue et al. [21] investigated the modeling of temporal structures in videos with Long Short-Term Memory (LSTM) [37]. However, even with five-layer LSTMs, trained on millions of videos, they do not show promising performance compared to ConvNets [72]. Patraucean et al. [77] used a spatio-temporal autoencoder to model video sequences through optical flow prediction and reconstruction of the next frame. Ballas et al. [8] used a Convolutional Gated Recurrent Unit (ConvGRU) which leverage information from different spatial levels of the activations. A general sequence to sequence framework *encoder-decoder* was introduced by [98] which utilizes a multilayered RNN to encode a sequence of inputs into one hidden state, following which another RNN takes the encoded state as inputs and decodes it into a sequence of outputs. [96] extended this general model to learn features from consecutive frames and proposed a composite model for unsupervised

LSTM autoencoder. I utilize the RNNs on video representation learning, improving the representation by being aware of the multirate nature of video content. Moreover, the temporal consistency between frames in the neighborhood is incorporated into the networks in an unsupervised way, providing richer training information and creating opportunities to learn from abundant untrimmed videos.

## 2.2 Bridging Vision and Language

### 2.2.1 Video Captioning

There is increased interest in the field of multimodal learning for bridging computer vision and natural language understanding [20, 45, 133, 111, 116, 131, 73]. Captioning is a particular popular task, and LSTM is heavily used as a recurrent neural network language model to automatically generate a sequence of words conditioned on the visual features, inspired by the general recurrent encoder-decoder framework [98]. Conditioned on the visual context, RNNs produce one word per step to generate captions for videos. Venugopalan et al. [111] used a stacked sequence to sequence (seq2seq) [98] model, in which an LSTM is used as a video sequence encoder and the other LSTM serves as a caption decoder. Yao et al. [131] incorporated the temporal attention mechanism in the description decoding stage. Pan et al. [74] proposed using a hierarchical LSTM to model videos sequences, while Yu et al. [134] used a hierarchical GRU network to model the structure of captions. [67] proposed a Speaker-Listener method to generate unambiguous descriptions. I demonstrate that the strong video representation learned in our model improves the video captioning task, confirming the generalization ability of our features.

### 2.2.2 Video Question Answering

The captioning task only generates a generic description for the entire image or video clip, and it is difficult to evaluate the quality of the sentences generated;

that is, it is difficult to judge whether one description is better than another. In addition, designing a proper metric for visual captioning which can reflect human judgment [22, 110] is still an open research problem.

I instead focus on a more fine-grained description of video content, and our method is simple to evaluate in multiple-choice form, i.e., by selection of the correct answer. A number of QA datasets and systems have been developed on images [4, 28, 64]. [28] used a complex dataset with free-style multilingual question-answer pairs; however it is difficult to evaluate the answers, and human judgement is usually required. [62] introduced an interesting multiple-choice fill-in-the-blank question answering task on abstract scenes, and [135] applied the task to natural images using various question templates. [124] incorporated an external knowledge base, i.e., DBpedia [5], to facilitate the image QA task by querying the relevant information from the knowledge base. In contrast, I leverage the external knowledge base by directly using the pre-trained models based on large datasets, e.g., BookCorpus [143], rather than training the language model from scratch. Unlike still images, video analysis can utilize the temporal information across frames, along with object and scene information. The richer structural information in videos potentially enables better understanding of the visual content while at the same time imposing challenges.

One of the video-based question answering works is [107], which built a query answering system based on a joint parsing graph from both text and videos. However, Tu et al. restricted their model to surveillance videos of predefined structure, which cannot deal with open-ended questions. MovieQA [102] uses movies as a single source, which are produced by professionals in controlled environment. Differently, I aim to deal with videos which could be produced by anyone in the wild. Therefore, I collect videos from various sources, e.g., cooking scenarios, unconstrained web videos from YouTube, based on which a model is trained to capture the dynamic temporal structure of unconstrained videos. Action forecasting, from the aspect of



temporal structure learning, was initially studied by [117]. To predict the potential actions, Vondrick et al. proposed to use a regression loss built upon a ConvNet and forecast limited categories of actions and objects in a very short period, e.g., one second. In contrast, I utilize a more flexible encoder-decoder framework, modeling a wider range of temporal information, and I mainly focus on multiple-choice question answering tasks in the temporal domain, which goes well beyond standard visual recognition.

### 2.3 Few-shot Video Classification

. Early works from Miller et al. [71], Fei-Fei et al. [24] and Lake et al. [56] utilized generative models for one-shot learning. Koch [51] attempted to train a Siamese network in a supervised way. Santoro et al. [89] was the first work to successfully bridge memory-augmented neural networks and one-shot learning. They took training examples in an episode as sequential inputs and trained the network to predict the label given previous examples. Vinyals et al. [115] used metric learning for few-shot recognition and utilized the attention kernel to measure the distance. Given a query, the network is trained to “point” to the nearest instance in the support set and the corresponding label is retrieved as the prediction. Ravi and Larochelle [81] trained a meta-learner based on Long Short-Term Memory (LSTM) [37] to generate updates for the classifier rather than using gradients. The meta-learner also learns a task-common weight initialization which captures shared knowledge across tasks. Finn et al. [26] used stochastic gradient descent as a meta-learner to update the parameters of the learner, which only learns the weight initialization. Snell et al. [93] applied a similar model to Vinyals [115], but they used Euclidean distance with their embedding function. Hariharan and Girshick [34] proposed the generation of images at testing time to improve few-shot recognition performance. Xu et al. [129] presented a key-value memory network to facilitate few-shot learning by extracting

knowledge from external knowledge bases, e.g., noisy web images. However, their setting is not the meta-learning paradigm. These works focus on image few-shot recognition, whereas I aim to learn a few-shot video model, which requires modeling complex video data.

### 2.3.1 Memory-Augmented Neural Networks

Memory-Augmented neural networks have gained increasing interest with the success of attention mechanism [7], Neural Turing Machine [33], and Memory Networks [123]. In RNNs, the states transferred between the steps can be interpreted as internal memory representations for the inputs. The state vector of the last step is usually used as the final representation for the whole input sequence. The fixed-size vector representation cannot encode long sequences in an effective way. Instead, the attention mechanism retains a sequence vectors as contexts for content-based addressing. The states in RNNs can change quickly over a few steps, while an external memory can retain information over the long term. Neural Turing Machine [33] is a computer-like network augmented with an external memory that can be addressed via content and location. The reading and writing operations are fully differentiable and weight updates through backpropagation are applied to every memory slot. Memory networks [123] and the improved end-to-end memory networks [97] have a large memory component for fact search and retrieval through content-based addressing. Key-value memory networks [70] decompose the memory into key and value parts, introducing a structural memory component to store question-answer pairs in a flexible way. Soft addressing is used in all these works, which is computationally expensive with growth of the memory size. Kaiser et al. [44] proposed a key-value memory module which performs hard updating to the memory, and a ranking loss is used to train the model to make accurate predictions. However, the memory stores only a fixed-size vector for an input, which is not suitable when the

input is a long sequence, e.g., video data. I thus propose our compound memory network, in which each slot stores a series of vectors that are stacked as a matrix representation.

## Chapter 3

# Bidirectional Multirate Reconstruction for Temporal Modeling in Videos

### 3.1 Introduction

In this chapter, I deal with learning a discriminative video representation from videos. I discuss the Multirate Visual Recurrent Model (MVRM) for modeling motion speed variances. Such a multirate learning process makes the learned model more capable of dealing with motion speed variances. I apply the proposed method to two challenging video tasks, i.e., complex event detection and video captioning, where it achieves the state-of-the-art performance. Notably, the method generates the best single feature for event detection with a relative improvement of 10.4% on the MEDTest-13 dataset and achieves the best performance in video captioning across all evaluation metrics on the YouTube2Text dataset.

A major limitation of [72] and [74] is that the input frames are encoded with a fixed sampling rate when training the RNNs. On the other hand, the motion speed of videos varies even in the same video. As shown in the Figure 3.1, there is almost no apparent motion in the first four frames, but fast motion is observed in the last three frames. The encoding rate should be correspondingly low for the first four frames, but high for the last three, as indicated by the solid arrow. The fixed rate strategy, however, is redundant for the first four frames, while important information for the last three frames is lost. The gap between the fixed encoding rate

---

This chapter is based on joint work with Zhongwen Xu, and Yi Yang (Zhu et al. 2017 [140]), presented primarily as it appears in the CVPR 2017 proceedings.

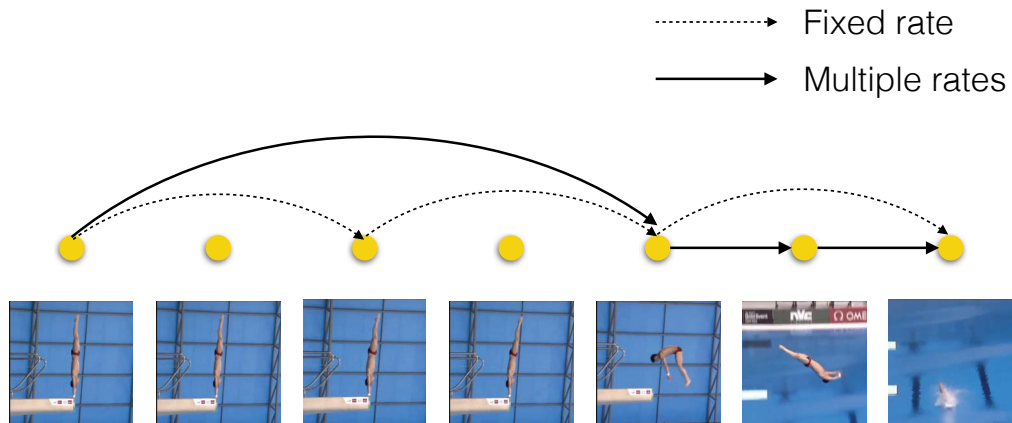


Figure 3.1 : Frame sampling rate should vary in accordance with different motion speed. In this example, only the last three frames have fast motion. The dashed arrow corresponds to a fixed sampling rate, while the solid arrow corresponds to multiple rates.

and motion speed variance in real world videos may degrade performance, especially when the variance is extensive.

Inspired by a recent study on neuroscience which shows that a common brain network underlies the capacity both to remember the past and imagine the future [90], we consider reconstructing two temporal transitions, i.e., present $\rightarrow$ past transition and present $\rightarrow$ future transition. Importantly, video motion speed changes constantly in untrimmed videos and Srivastava et al. directly used an LSTM with a single fixed sampling rate, making it vulnerable to motion speed variance.

We propose an unsupervised method to learn from untrimmed videos for temporal information modeling without the heavy cost of computing optical flows. It makes the following two major contributions. First, our Multirate Visual Recurrent Model adopts multiple encoding rates, and together with the reading gate and the updating gate in the Gated Recurrent Unit, it enables communication between different encoding rates and collaboratively learns a multirate representation which is

robust to motion speed variance in videos. Second, we leverage the mutual benefit of two learning processes by reconstructing the temporal context in two directions. The two learning directions regularize each other, thereby reducing the overfitting problem. The two contributions yield a new video representation, which achieves the best performance in two different tasks. Note that the method proposed in [128] has been demonstrated to be the best single feature for event detection, and our method outperforms this method with a relative improvement of 10.4% and 4.5% on two challenging datasets, i.e., MEDTest-13 and MEDTest-14 respectively. In the video captioning task, our single feature outperforms other state-of-the-art methods across all evaluation metrics, most of which use multiple features. It is worthwhile mentioning that in very rare cases, one method can outperform all others for video captioning over all evaluation metrics. These results demonstrate the effectiveness of the proposed method.

## 3.2 Multirate Visual Recurrent Models

In this section, we introduce our approach for video sequence modeling. We first review the structure of Gated Recurrent Unit (GRU) and extend the GRU to a multirate version. The model architecture for unsupervised representation learning is then introduced, which is followed by task specific models for event detection and video captioning. In the model description, we omit all bias terms in order to increase readability.

### 3.2.1 Multirate Gated Recurrent Unit

**Gated Recurrent Unit.** At each step  $t$ , a GRU cell takes a frame representation  $\mathbf{x}_t$  and previous state  $\mathbf{h}_{t-1}$  as inputs and generates a hidden state  $\mathbf{h}_t$  and an output

$\mathbf{o}_t$  which are calculated by,

$$\begin{aligned}
\mathbf{r}_t &= \sigma(\mathbf{U}_r \mathbf{x}_t + \mathbf{V}_r \mathbf{h}_{t-1}), \\
\mathbf{z}_t &= \sigma(\mathbf{U}_z \mathbf{x}_t + \mathbf{V}_z \mathbf{h}_{t-1}), \\
\bar{\mathbf{h}}_t &= \tanh(\mathbf{U}_{\bar{h}} \mathbf{x}_t + \mathbf{V}_{\bar{h}} (\mathbf{r}_t \odot \mathbf{h}_{t-1})), \\
\mathbf{h}_t &= (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \bar{\mathbf{h}}_t, \\
\mathbf{o}_t &= \mathbf{W}_o \mathbf{h}_t,
\end{aligned} \tag{3.1}$$

where  $\mathbf{x}_t$  is the input,  $\mathbf{r}_t$  is the reset gate,  $\mathbf{z}_t$  is the update gate,  $\mathbf{h}_t$  is the proposed state,  $\bar{\mathbf{h}}_t$  is the internal state,  $\sigma$  is the sigmoid activation function,  $\mathbf{U}_*$  and  $\mathbf{V}_*$  are weight matrices, and  $\odot$  is element-wise multiplication. The output  $\mathbf{o}_t$  is calculated by a linear transformation from the state  $\mathbf{h}_t$ . We denote the whole process as:

$$\mathbf{h}_t, \mathbf{o}_t = \text{GRU}(\mathbf{x}_t, \mathbf{h}_{t-1}), \tag{3.2}$$

and when it has iterated  $S$  steps, we can obtain the state of the last step  $\mathbf{h}_S$ .

**Multirate Gated Recurrent Unit (mGRU).** Inspired by clockwork RNN [52], we extend the GRU cell to a multirate version. The clockwork RNN uses delayed connections for inputs and inter-connections between steps to capture longer dependencies. Unlike traditional RNNs where all units in the states follow the protocol in Eq. 3.1, states and weights in the clockwork RNN are divided into groups to model information at different rates. We divide state  $\mathbf{h}_t$  into  $k$  groups, and each group  $g_i$  has a clock period  $T_i$ , where  $i \in \{1, \dots, k\}$ .  $T_i$  can be arbitrary numbers, and we empirically use  $k = 3$  and set  $T_1, T_2, T_3 = 1, 3, 6$ . Faster groups (with smaller  $T_i$ ) take inputs more frequently than slower groups, and the slower module skips more inputs. Formally, at each step  $t$ , matrices of the group satisfying  $(t \bmod T_i) = 0$  are

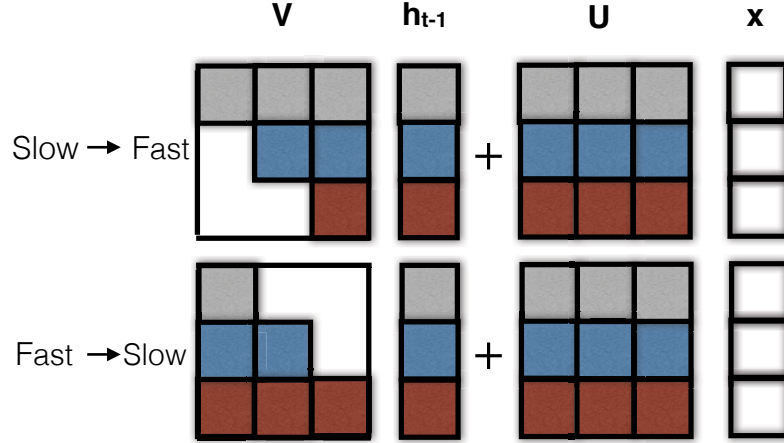


Figure 3.2 : We illustrate the two modes in the mGRU. In the slow to fast mode, the state matrices  $\mathbf{V}_*$  are block upper-triangular matrices and in the fast to slow mode, they are block lower-triangular matrices.

activated and are used to calculate the next state, which is

$$\begin{aligned}
 \mathbf{r}_t^i &= \sigma(\mathbf{U}_r^i \mathbf{x}_t + \sum_{j=1}^k \mathbf{V}_r^{i,j} \mathbf{h}_{t-1}^j), \\
 \mathbf{z}_t^i &= \sigma(\mathbf{U}_z^i \mathbf{x}_t + \sum_{j=1}^k \mathbf{V}_z^{i,j} \mathbf{h}_{t-1}^j), \\
 \bar{\mathbf{h}}_t^i &= \tanh(\mathbf{U}_h^i \mathbf{x}_t + \sum_{j=1}^k \mathbf{V}_h^{i,j} (\mathbf{r}_t^i \odot \mathbf{h}_{t-1}^j)), \\
 \mathbf{h}_t^i &= (1 - \mathbf{z}_t^i) \odot \mathbf{h}_{t-1}^i + \mathbf{z}_t^i \odot \bar{\mathbf{h}}_t^i,
 \end{aligned} \tag{3.3}$$

where the state weight matrices  $\mathbf{V}_*$  are divided into  $k$  block-rows and each block-row is partitioned into  $k$  block-columns.  $\mathbf{V}_*^{i,j}$  denotes the sub-matrix in block-row  $i$  and block-column  $j$ . The input weight matrices  $\mathbf{U}_*$  are divided  $k$  block-rows and  $\mathbf{U}_*^i$  denotes the weights in block-row  $i$  and

$$\sum_{j=1}^k \mathbf{V}_*^{i,j} \mathbf{h}_{t-1}^j = \begin{cases} \sum_{j=1}^i \mathbf{V}_*^{i,j} \mathbf{h}_{t-1}^j, & \text{Fast} \rightarrow \text{slow mode} \\ \sum_{j=i}^k \mathbf{V}_*^{i,j} \mathbf{h}_{t-1}^j, & \text{Slow} \rightarrow \text{fast mode} \end{cases} \tag{3.4}$$

Two modes can be used for state transition. In the slow to fast mode, states of faster groups consider previous slower states, thus the faster states incorporate information not only at the current speed but also information that is slower and



more coarse. The intuition for the fast to slow mode is that when the slow mode is activated, it can take advantage of the information already encoded in the faster states. The two modes are illustrated in Figure 3.2. Empirically, we use the fast to slow mode in our model as it performed better in the preliminary experiments.

If  $(t \bmod T_i) \neq 0$ , the previous state is directly passed over to the next state,

$$\mathbf{h}_t^i = \mathbf{h}_{t-1}^i. \quad (3.5)$$

Figure 3.3 illustrates the state iteration process. Note that not all previous modules are considered to calculate the next state at each step, thus fewer parameters will be used and the training will be more efficient.

### 3.2.2 Unsupervised Video Sequence Reconstruction

Video sequences are highly correlated to their neighboring context clips. We use the idea of context reconstruction for video sequence modeling. The similar methods have been successfully applied for language modeling and other language tasks [68, 49]. In the unsupervised training process, we follow the classic sequence-to-sequence (seq2seq) model [98] where an encoder encodes a sequence of inputs and passes the last state to the decoder for target sequence generation. In our scenario, the mGRU encoder takes frame-level features extracted from the pre-trained convolutional models as inputs and generates the state at each step which will be attended by the decoders. The state of the last step of the encoder is passed to the decoder, i.e.,  $\mathbf{h}_0^{\text{dec}} = \mathbf{h}_S^{\text{enc}}$ . Two decoders are used to predict the context sequences of the inputs, i.e., reconstructing the frame-level representations of the previous sequence and next sequence.

**Decoder.** We use the seq2seq model with attention mechanism to model video temporal structures via context reconstruction. We denote that  $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n)$  is the previous sequence of input sequence  $\mathbf{X}$ , and  $\mathbf{Z} = (\mathbf{z}_1, \dots, \mathbf{z}_n)$  is the next

sequence. The decoder is a GRU conditioned on the encoder outputs  $\mathbf{o}_{1,\dots,S}^{\text{enc}}$  and the last step state  $\mathbf{h}_S^{\text{enc}}$  of the encoder. We use the attention mechanism at each step to help the decoder to decide which frames in the input sequence might be related to the next frame reconstruction. At step  $t$ , the decoder  $\phi$  generates the prediction  $\mathbf{o}_t^{\text{dec}}$  by calculating,

$$\begin{aligned}
\mathbf{y}_t^{\text{attn}} &= \text{Linear}(\mathbf{y}_t, \mathbf{a}_{t-1}), \\
\mathbf{h}_t^{\text{dec}}, \mathbf{o}_t^{\text{attn}} &= \text{GRU}(\mathbf{y}_t^{\text{attn}}, \mathbf{h}_{t-1}^{\text{dec}}), \\
e_t^i &= \mathbf{v}^T \tanh(\mathbf{W}_{he} \mathbf{h}_t^{\text{dec}} + \mathbf{W}_{oe} \mathbf{o}_i^{\text{enc}}), \\
a_t^i &= \exp(e_t^i) / \sum_{j=1}^S \exp(e_t^j), \\
\mathbf{a}_t &= \sum_{i=1}^S a_t^i \mathbf{o}_i^{\text{enc}}, \\
\mathbf{o}_t^{\text{dec}} &= \text{Linear}(\mathbf{o}_t^{\text{attn}}, \mathbf{a}_t),
\end{aligned} \tag{3.6}$$

where  $\text{Linear}(\mathbf{a}, \mathbf{b}) = \mathbf{W}_a \mathbf{a} + \mathbf{W}_b \mathbf{b}$ ,  $a_t^i$  is the normalized attention weight for encoder output  $\mathbf{o}_i^{\text{enc}}$  and  $\mathbf{a}_t$  is the weighted average of the encoder outputs. We use two decoders that do not share parameters: one for the past sequence reconstruction and the other for the future sequence reconstruction (Figure 3.4). The decoders are trained to minimize the reconstruction loss of two sequences, which is

$$\begin{aligned}
&\sum_t \ell(\phi(\mathbf{y}_{<t}, \mathbf{o}_{1,\dots,S}^{\text{enc}}, \mathbf{h}_S^{\text{enc}}; \theta), \mathbf{y}_t) + \\
&\sum_{t'} \ell(\phi(\mathbf{z}_{<t'}, \mathbf{o}_{1,\dots,S}^{\text{enc}}, \mathbf{h}_S^{\text{enc}}; \theta'), \mathbf{z}_{t'}).
\end{aligned} \tag{3.7}$$

We choose the Huber loss for regression due to its robustness following Girshick [29],

$$\ell(y, \bar{y}) = \begin{cases} \frac{1}{2}(y - \bar{y})^2 & \text{for } |y - \bar{y}| \leq \delta, \\ \delta |y - \bar{y}| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases} \tag{3.8}$$

We set  $\delta = 0.5$  in all experiments.

For the past reconstruction, we reverse the input order as well as the target order to minimize information lag [98]. The two decoders are trained with the encoder via

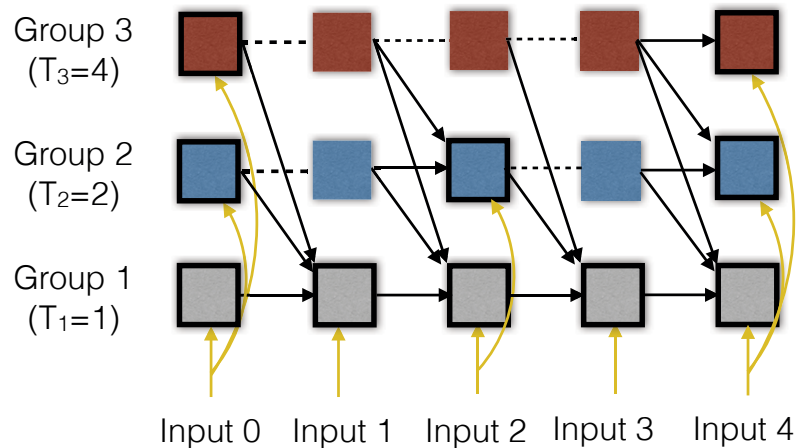


Figure 3.3 : Unrolled mGRU. In the example, the state is divided into three groups and the slow to fast mode is shown. At each step  $t$ , groups satisfying  $(t \bmod T_i) = 0$  are activated (cells with black border). For example, at step 2, group 1 and group 2 are activated. The activated groups take the frame input and previous states to calculate the next states. For those that are inactivated, we simply pass the previous states to the next step. Group 1 is the fastest and group 3 is the slowest with larger  $T_i$ . The slow to fast mode is the mode by which the slower groups pass the states to the faster groups.

backpropagation, and we regularize the network by randomly dropping one decoder for each batch. As we have two decoders in our model, each decoder will have the probability of being chosen for training of 0.5 (Figure. 3.4).

During unsupervised training, we uniformly sample video frames and extract frame-level features from convolutional models. We set the sequence length to  $K$ , i.e., the encoder takes  $K$  frames as inputs, while the decoders reconstruct previous  $K$  frames and next  $K$  frames. We randomly sample a temporal window of consecutive  $3K$  frames (3 segments) during training. If the video length is less than  $3K$ , we pad zeros for each segment.

### 3.2.3 Complex Event Detection

We validate the unsupervised learned features on the task of complex event detection. We choose the TRECVID Multimedia Event Detection (MED) task as it is more dynamic and complex compared to the action recognition task, in which the target action duration is short and usually lasts only seconds. As the features from the unsupervised training are not discriminative, i.e., label information has not been applied during training, we further train the encoder for video classification. We use the mGRU encoder to encode the video frames and take the last hidden state in the encoder for classification. We do not apply losses at each step, e.g., the LSTM model in [72], as the video data in our task is untrimmed, which is more noisy and redundant. We use the network structure of FC(1024)-ReLU-Dropout(0.5)-FC(1024)-ReLU-Dropout(0.5)-FC( $class\_num+1$ )-Softmax. Since there are background videos which do not belong to any target events, we add another class for these videos.

During supervised training, we first initialize the weights of the encoder with the weights pre-trained via unsupervised context reconstruction. For each batch, instead of uniformly sampling videos within the training set, we keep the ratio of the number of positive and background videos to 1 : 2. We bias the mini-batch sampling because of the imbalance between the positive and negative examples.

During inference, the encoder generates multirate states at each step, and there are several ways to pool the states to obtain a global video representation. One simple approach is to average the outputs, and the obtained global video representation is then classified with a Linear SVM. The other way is to encode the outputs with an encoding method. Xu et al. [128] found that Vector of Locally Aggregated Descriptors (VLAD) [41] encoding outperforms average pooling and Fisher Vectors [78] over ConvNets activations by a large margin on the MED task. We thus apply the VLAD encoding method to encode the RNN representations.

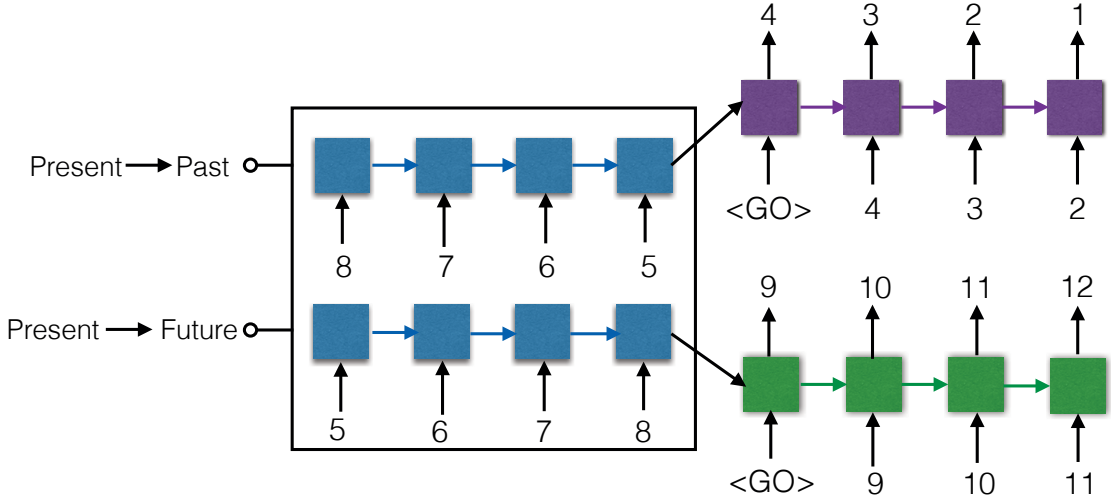


Figure 3.4 : The model architecture of unsupervised video representation learning. In this model, two decoders are used to predict surrounding contexts by reconstructing previous frames and next frame sequences. The “<GO>” input, which is a zero vector, is used at step 0 in the decoder. During training, one of the two decoders is used with a probability of 0.5 for reconstruction.

Given inputs  $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$  and centers  $\mathbf{C} = \{\mathbf{c}_1, \dots, \mathbf{c}_K\}$  which are calculated by the k-means algorithm on sampled inputs, for each  $k \in \{1, \dots, K\}$ , we have,

$$\mathbf{u}_k = \sum_{i:\text{Nearest}(\mathbf{x}_i)=\mathbf{c}_k} \mathbf{x}_i - \mathbf{c}_k, \quad (3.9)$$

where  $\mathbf{x}_i$  is assigned to the center  $\mathbf{c}_k$  if it is the nearest center. Concatenating  $\mathbf{u}_k$  over all  $K$  centers, we obtain the feature vector of size  $DK$  where  $D$  is the dimension of  $\mathbf{x}_i$ . Normalization methods are used to improve the encoding performance. Power normalization, often signed square rooting (SSR), is usually used to convert each element  $x_i$  into  $\text{sign}(x_i)\sqrt{|x_i|}$ . The intra-normalization method normalizes representations for each center, followed by the  $\ell_2$  normalization for the whole feature vector [78]. The final normalized representation is classified with a Linear SVM.

Note that the states in mGRU are divided into groups, we thus encode the state

of the three different scales independently. We combine the three scores by average fusion.

### 3.2.4 Video Captioning

We also demonstrate the generalization ability of our proposed video representation on the video captioning task. In video captioning, an encoder is used to encode video representations and a decoder is used to generate video descriptions. We follow the basic captioning decoding process. Given a video sequence  $\mathbf{X}$  and a description sequence  $Y = \{y_1, \dots, y_N\}$ , where each word is represented by a one-hot vector and a one-of- $K$  ( $K$  is the vocabulary size) embedding is used in the decoder input to represent a discrete word with a continuous vector, the overall objective is to maximize the log-likelihood of the generated sequence,

$$\max_{\boldsymbol{\theta}} \sum_{t=1}^N \log \Pr(y_t | y_{<t}, \mathbf{X}; \boldsymbol{\theta}). \quad (3.10)$$

Softmax activation is used on the decoder output to obtain the probability of word  $y_t$ . The attention mechanism (Eq. 3.6) is used in both the input and output of the decoder.

## 3.3 Results

We show the results of our experiments on complex event detection and video captioning tasks. We implement our model using the TensorFlow framework [3].

### 3.3.1 Complex Event Detection

#### *Dataset*

We collect approximately 220,000 videos without label information from TRECVID MED data, which excludes videos in MEDTest-13 and MEDTest-14, for unsupervised training. The average length of the collected videos is 130 seconds with a total duration of more than 8,000 hours.

We use the challenging MED datasets with labels, namely, TRECVID MEDTest-13 100Ex [1] and TRECVID MEDTest-14 100Ex [2] for video classification\*. There are 20 events in each dataset, 10 of which overlap. It consists of approximately 100 positive exemplars for each event in the training set, and 5,000 negative exemplars. In the testing set, there are about 23,000 videos and the total duration in each collection is approximately 1,240 hours. The average video length is 120 seconds. These videos are temporally untrimmed YouTube videos of various resolutions and quality. We use the mean Average Precision (mAP) as the performance metric following the NIST standard [1, 2].

### *Model Specification*

For both unsupervised training and classification, we uniformly sample video frames at the rate of 1 FPS and extract features for each frame from GoogLeNet with the Batch Normalization [39] pre-trained on ImageNet. Following standard image preprocessing procedures, the shorter edges of frames are rescaled to 256 and we crop the image to  $224 \times 224$ . We use activations after the last pooling layer and obtain representations with length 1,024. There are 20 classes in the MEDTest-13 and MEDTest-14 datasets, thus with the background class, we have 21 classes in total. In the training stage, we set sequence length  $K$  to 30 and pad zeros if the video has fewer than 30 frames. During inference, we take the whole video as input and use 150 steps.

**Training details.** We use the following settings in all experiments unless otherwise stated. The model is optimized with ADAM [48], and we fix the learning rate at  $1 \times 10^{-4}$  and clip the global gradients at norm 10. We use a single RNN layer for both the encoder and decoder, and the cell size is set to 1,024. We set the attention size to 50 and regularize the network by using Dropout [95] in the input and output

---

\*Development data is not updated for TRECVID MED 15 and TRECVID MED 16 competition.

Methods	MEDTest-13	MEDTest-14
GoogLeNet	32.0	25.1
mGRU	<b>39.6</b>	<b>32.2</b>

Table 3.1 : Comparison between GoogLeNet features and our mGRU model. Average pooling is used for both models. The result shows our feature representation significantly outperforms the GoogLeNet feature.

layer [79]. We also add Dropout when the decoder copy state from the encoder and all dropout probability is set to 0.5. Weights are initialized with Glorot uniform initialization [31] and weight decay of  $1 \times 10^{-4}$  is applied for regularization. In the supervised training, we initialize the weights of the encoder using the learned weights during unsupervised learning, and the same sequence length is used as in the unsupervised training stage.

### *Results*

**Average pooling.** For the GoogLeNet baseline, we average frame-level features and use a Linear SVM for classification. For our model, we first train an unsupervised encoder-decoder model with mGRU and fine-tune the encoder with label information. To make a fair comparison with the GoogLeNet baseline, we extract outputs of the mGRU encoder at each step and average them to obtain a global representation for classification. Note that both feature representations have same dimensions and we empirically set  $C = 1$  for both of the linear classifiers. The result is shown in Table 3.1 and shows that our model with temporal structure learning is able to encode valuable temporal information for classification.

**VLAD Encoding.** We now show that VLAD encoding is useful for aggregating RNN representations. We compare our method with GoogLeNet features using



Methods	MEDTest-13	MEDTest-14
GoogLeNet	42.0	33.6
mGRU	<b>44.5</b>	<b>37.3</b>

Table 3.2 : Comparison between GoogLeNet and mGRU models when VLAD encoding is used to aggregate frame-level features.

Methods	MEDTest-13	MEDTest-14
mGRU w/o attention	32.7	27.5
mGRU w/o context	37.1	30.1
mGRU w/o multirate	36.5	29.3
mGRU (full)	37.4	30.6

Table 3.3 : Comparison between mGRU and other variants in the unsupervised training stage. Detailed discussion can be found in text.

VLAD encoding. Following [128], we set the number of k-means centers to 256 and the dimension of PCA is 256. Three scales are learned at each step for our mGRU model. We divide the state into three segments and each sub-state is individually aggregated by VLAD. Note that each encoded representation has the same feature vector length as the GoogLeNet model, and we use late fusion to combine the scores of the three scales. The results in Table 3.2 show that our mGRU model outperforms GoogLeNet features when encoded by VLAD. It also shows that VLAD encoding outperforms average pooling for RNN representations. Our model also achieves state-of-the-art performance on the MEDTest-13 and MEDTest-14 100Ex datasets.

### *Ablation Study*

We compare several variants in the unsupervised training, and show the performance of different components. The results are shown in Table 3.3. We obtain features from the unsupervised model by extracting states from the encoder at each step, which are then averaged to obtain a global video representation. The results show that the representation learning from unsupervised training without discriminative information also achieves good results.

**Attention.** We compare our model with a model without the attention mechanism, where temporal attention is not used and the decoder is forced to perform reconstruction based only on the last encoder state, i.e., “mGRU w/o attention” in Table 3.3. The results show that the attention mechanism is important for learning good video representations and also helps the learning process of the encoder.

**Context.** In a model without context reconstruction, i.e., only one decoder is used (autoencoder), neither past nor future context information is considered, i.e., “mGRU w/o context” in Table 3.3. The results show that with context prediction, the encoder has to consider temporal information around the video clip, which models the temporal structures in a better way.

**Multirate.** We also show the benefit of using mGRU by comparing it with the basic GRU, i.e., “mGRU w/o multirate” in Table 3.3. Note that the mGRU model has fewer parameters but better performance. It shows that an mGRU that encodes multirate video information is capable of learning better representations from long, noisy sequences.

**Pre-training.** We now show the advantages of the unsupervised pre-training process by comparing an encoder with random initialization with the same encoder whose weights are initialized by the unsupervised model. The result is shown in Table 3.4 and demonstrates that the unsupervised training process is beneficial to

Methods	MEDTest-13	MEDTest-14
mGRU (random)	38.3	29.5
mGRU (pre-trained)	39.6	32.2

Table 3.4 : Comparison between models which have the same structure but different initialization. This shows that good initialization enables better features to be learned.

video classification. It incorporates context information in the encoder, which is an important cue for the video classification task.

### *Comparison with the State-of-the-art*

We compare our model with other models and the results are shown in Table 3.5. Our single model achieves the state-of-the-art performance on both the MEDTest-13 and MEDTest-14 100Ex settings compared with the performances of other single models. We report the C3D result by using the pre-trained model [104] and we set the length of the input short clip to 16. Features are averaged across clips which are classified with a Linear SVM. Our model with VLAD encoding outperforms previous state-of-the-art results with 4.2% on MEDTest-13 100Ex and 1.6% on MEDTest-14 100Ex.

### **3.3.2 Video Captioning**

We now validate our model on the video captioning task. Our single model outperforms previous state-of-the-art single models across all metrics.

Models	MEDTest-13	MEDTest-14
IDT + FV [128]	34.0	27.6
IDT + skip + FV [57]	36.3	29.0
VGG + RBF [138]	-	35.0
C3D [104] *	36.9	31.4
VGG16 + VLAD [128]	-	33.2
NI-SVM <sub>2</sub> [12]	39.2	34.4
VGG16+LCD+VLAD [128]	40.3	35.7
LSTM autoencoder [96] *	38.2	31.0
GoogLeNet + VLAD *	42.0	33.6
Our method	<b>44.5</b>	<b>37.3</b>

Table 3.5 : Comparison with other methods. We achieve state-of-the-art performance on both MEDTest-13 and MEDTest-14 100Ex datasets. \* denotes that the model is implemented by ourselves.

### *Dataset*

We use the YouTube2Text video corpus [13] to evaluate our model on the video captioning task. The dataset has 1,970 video clips with an average duration of 9 seconds. The original dataset contains multi-lingual descriptions covering various domains, e.g., sports, music, animals. Following [113], we use English descriptions only and split the dataset into training, validation and testing sets containing 1,200, 100, 670 video clips respectively. In this setting, there are 80,839 descriptions in total with about 41 sentences per video clip. The vocabulary size we use is 12,596 including <GO>, <PAD>, <EOS>, <UNK>.

We evaluate the performance of our method on the test set using the evaluation

Methods	B@1	B@2	B@3	B@4	M	C
GRU	79.46	67.52	57.98	47.14	32.31	72.46
mGRU	79.42	67.79	58.32	48.12	32.79	73.21
mGRU+ pre-train	<b>80.76</b>	<b>69.49</b>	<b>60.03</b>	<b>49.45</b>	<b>33.39</b>	<b>75.45</b>

Table 3.6 : Comparison between different models on YouTube2Text dataset. GoogLeNet features are used as frame-level representations. **B**, **M**, **C** are short for BLEU, METEOR, CIDEr.

script provided by [14] and the results are returned by the evaluation server. We report BLEU [76], METEOR [19] and CIDEr [110] scores for comparison with other models. We stick with a single rule during model selection, namely we choose the model with the highest METEOR score on the validation set.

### *Model Specification*

The video length in the YouTube2Text dataset is short, thus we uniformly sample frames at a higher frame rate of 15 FPS. The sequence length is set to 50 and we use the default hyper-parameters in the last experiment. We use two different convolutional features for the video captioning task, i.e., GoogLeNet features and ResNet-200 features [36]. We use beam search during decoding by default and set the beam size to 5 following [134] in all experiments. Attention size is set to 100 empirically.

### *Results*

We first use GoogLeNet features and the result is shown in Table 3.6. We compare our mGRU with GRU which shows that mGRU outperforms GRU on all metrics except BLEU@1. However, the difference is only 0.04%. We initialize the

Methods	B@1	B@2	B@3	B@4	M	C
GRU	80.88	70.15	61.08	51.06	33.48	79.16
mGRU	82.03	71.41	62.38	52.49	33.91	78.41
mGRU+ pre-train	<b>82.49</b>	<b>72.16</b>	<b>63.30</b>	<b>53.82</b>	<b>34.45</b>	<b>81.20</b>

Table 3.7 : Comparison between different models on YouTube2Text dataset. ResNet-200 features are used as frame-level representations. **B**, **M**, **C** are short for BLEU, METEOR, CIDEr.

mGRU encoder via unsupervised context learning and the result shows that with good initialization, performance is improved by more than 1.0% on the BLEU and CIDEr scores and 0.6% on the METEOR score compared with random initialization. We also utilize the recent ResNet-200 network as a convolutional model. We use the pre-trained model and follow the same image preprocessing method. The result of using ResNet-200 is shown in Table 3.7 and demonstrates that our MVRM method not only works better than GRU on different tasks, but also works better on different convolutional models. Additionally, we can improve all the metrics with ResNet-200 network.

### *Comparison with the State-of-the-art*

We compare our methods with other models on the YouTube2Text dataset. Results are shown in Table 3.8. “S2VT” [111] is the first model to use a general encoder-decoder model for video captioning. “Temporal Attention” [131] uses the temporal attention mechanism on the video frames to obtain better results. “Bi-GRU-RCN” [131] uses a ConvGRU to encode activations from different convolutional layers. “LSTM-E” [75] uses embedding layers to jointly project visual and

Methods	BLEU@4	METEOR	CIDEr
S2VT [111]	-	29.20	-
Temporal attention [131]	41.92	29.60	51.67
GoogLeNet+ Bi-GRU-RCN <sub>1</sub> [8]	48.42	31.70	65.38
GoogLeNet+ Bi-GRU-RCN <sub>2</sub> [8]	43.26	31.60	68.01
VGG+LSTM-E [75]	40.20	29.50	-
C3D+LSTM-E [75]	41.70	29.90	-
GoogLeNet+HRNE+ Attention [74]	43.80	33.10	-
VGG+p-RNN [134]*	44.30	31.10	62.10
C3D+p-RNN [134]*	47.40	30.30	53.60
GoogLeNet+MVRM	<b>49.45</b>	<b>33.39</b>	<b>75.45</b>

Table 3.8 : Comparison with other models without fusion. \* denotes that the model is trained with different settings ([134] used the train+val data for training).

text features. Our MVRM method has similar performance to [74], but with the pre-training stage, we outperform [74] in all metrics. Some methods fuse additional motion features like C3D [104] features, e.g., Pan et al. [74] obtained 33.9% on METEOR after combining multiple features. With ResNet-200, we can obtain **34.45%** on METEOR.

### 3.4 Conclusion

We propose a Multirate Visual Recurrent Model to learn multirate representations for videos. We model the video temporal structure via context reconstruction,

and show that unsupervised training is important for learning good representations for both video classification and video captioning. The proposed method achieves state-of-the-art performance on two tasks. In the future, we will investigate the generality of the video representation in other challenging tasks, e.g., video temporal localization [23] and video question answering [141, 102]



## Chapter 4

# Uncovering the Temporal Context for Video Question Answering

### 4.1 Introduction

In this chapter, I introduce Video Question Answering in the temporal domain to infer the past, describe the present and predict the future. I present an encoder-decoder approach using Recurrent Neural Networks to learn the temporal structures of videos and introduce a dual-channel ranking loss to answer multiple-choice questions. I explore approaches for finer understanding of video content using the question form of “fill-in-the-blank”, and collect the Video Context QA dataset (VCQA) consisting of 109,895 video clips with a total duration of more than 1,000 hours from existing TACoS, MPII-MD and MEDTest 14 datasets. In addition, 390,744 corresponding questions are generated from annotations. Extensive experiments demonstrate that the approach significantly outperforms the compared baselines.

We focus on Video Question Answering (Video QA) in the temporal domain. Different from MovieQA [102], our Video QA task focuses on video temporal context understanding and thus consists of three subtasks, which are describing the present, inferring the past and the future.

As shown in Fig. 4.1, if we see a man slicing cucumber on a cutting board, we can infer that he *previously* took a knife from the drawer, and predict that he will put the cucumber slices on a plate *afterwards*. As with Image QA, Video QA requires a

---

This chapter is based on joint work with and Zhongwen Xu, Yi Yang, and Alex G. Hauptmann (Zhu et al. 2017 [141]), presented primarily as it appears in the IJCV 2017.

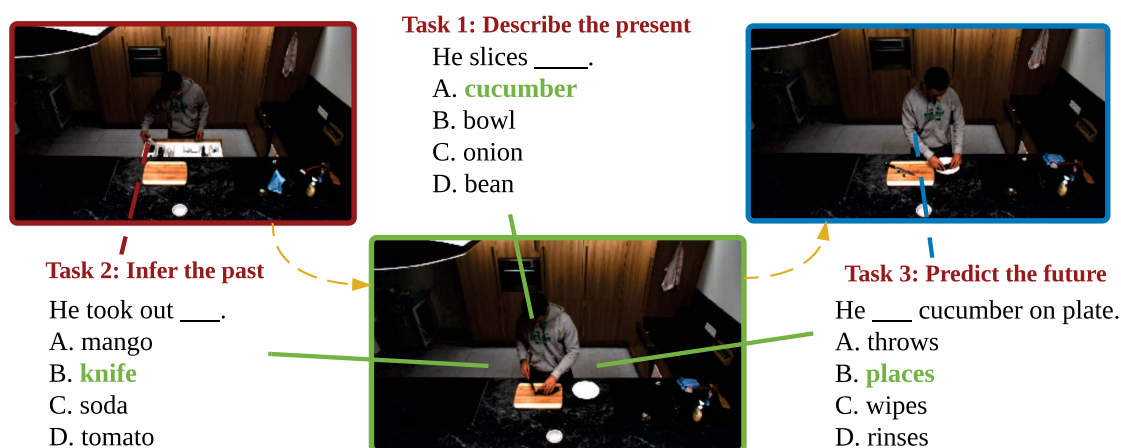


Figure 4.1 : Questions and answers about the past, the present and the future. Our system includes three subtasks, which infer the *past*, describe the *present*, and predict the *future*, while *only the current frames are observable*. Best viewed in color

finer understanding of videos and sentences than video captioning where questions about local objects and actions are required to be answered. Despite the success of video captioning in [111, 131, 74], a number of research challenges remain unsolved, which means these methods are not readily applicable to Video QA.

Firstly, a Video QA system should explore more knowledge beyond the visual information and coarse sentence annotations because it requires a finer understanding of video content and questions. For the sake of video captioning, existing systems train LSTM models based on the video content and associated coarse sentence annotations alone. Because the size of the description embedding matrix is very large and many words usually appear fewer than 10 times in all descriptions, the model overfits easily. A study [62] found that visual information and textual information are mutually beneficial. We developed a new way to approach Video QA, by appropriately integrating information, including sentences, words, and visual cues, within a joint learning framework to maximize the mutual benefit. Using this method, external knowledge bases (e.g. BookCorpus [143] and Google News [69]) can be readily

incorporated. Because the external knowledge bases reflect the underlying correlations between related entities, our approach is able to better parse questions and answers.

Secondly, a Video QA system should be capable of reasoning across video frames, including inferring the past, describing the present, and predicting the future, all of which are strongly correlated. Gated Recurrent Unit (GRU) [15] has demonstrated promising performance on sequence modeling tasks, partially because it has a simpler neural structure than LSTM. On top of GRU, we propose an *encoder-decoder* approach with a *dual-channel ranking loss* to learn three video representations, one for each Video QA subtask, i.e., past inference, present description, and future prediction. One appealing feature of our approach is that the encoder-decoder approach is able to model a wider range of temporal information, and the reduced number of weight parameters in GRU makes it more robust to overfitting in temporal modeling. Further, the approach eliminates the need to create a large number of labels to train the sequence model by embedding visual features in a semantic space.

Thirdly, a well-defined quantitative evaluation metric and datasets from different domains to track progress of this important research [65, 64, 4] are required. Manually providing groundtruth for a large number of videos is extremely human labor intensive. BLEU [76] has been widely used as an evaluation metric for image captioning, but a few research papers and competition reports have indicated that BLEU is not a reliable metric and cannot reflect human judgment [55, 110]. Following [62, 135], we evaluated our question answering approach in the form of “fill-in-the-blank” (FITB) multiple choice responses. We utilized over 100,000 real-world videos clips from existing TACoS, MPII-MD and MEDTest 14 datasets, and generated 400,000 designed questions with more than 1,000,000 candidate answers. This dataset will be released to the public and can be used as the benchmark for this research. The main advantage is that it is more convenient for quantitative

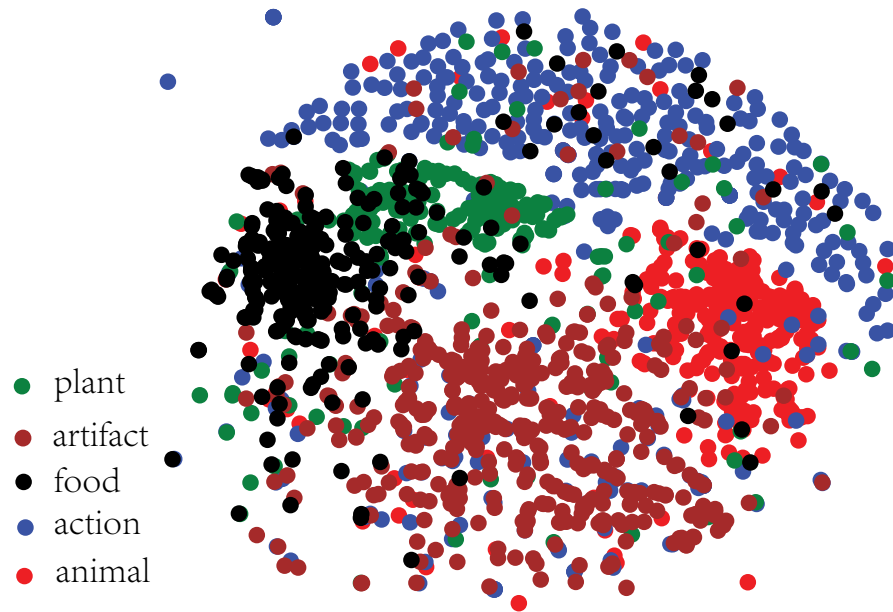


Figure 4.2 : t-SNE visualization of word embeddings for each category learned from word2vec model.

evaluation than free-style question answering.

We propose a new framework for Video QA by carefully addressing the three aforementioned challenges. The rest of sections are organized as follows. After introducing related works, we detail the large scale dataset we have collected for Video QA tasks. We then present our video temporal structure modeling approach and dual-channel learning-to-rank method for question answering. Extensive experiments are conducted to validate our approach.

## 4.2 Dataset Collection and Task Definitions

The goal of our work is to present a Video QA system in the temporal domain that can infer the past, describe the present and predict the future. We first describe our Video Context QA dataset (VCQA) collection and the method of automatically generating template questions in Section 4.2.1. Task definitions and dataset analysis will be discussed in Section 4.2.2.

### 4.2.1 Dataset and QA Pair Generation

We utilized more than 100,000 videos and 400,000 questions in total, while QA pairs were generated from existing datasets in different domains: a cooking scenario, DVD movies, and web videos:

1. **TACoS** [82]. The TACoS dataset consists of 127 long videos with a total of 18,227 annotations in the *cooking scenario*. It provides multiple sentence descriptions at a fine-grained level, i.e., for each short clip in each long video.
2. **MPII-MD** [85]. MPII-MD is collected from *DVD movies* where descriptions are generated from movie scripts semi-automatically. The dataset contains 68,375 clips and one annotation on average is provided for each clip.
3. **TRECVID MEDTest 14** [2]. TRECVID MEDTest 14 is a complex event wild video dataset collected from *web* hosting services such as YouTube. Videos in the dataset total some 1,300 hours in duration. The videos are untrimmed and an annotation is provided for each long video which can be regarded as a coarse high-level summarization compared to the TACoS and MPII-MD datasets.

**Question template generation.** [83] transform image descriptions to questions with only four question types, i.e., objects, number, color and location. It is difficult to automatically transform descriptions into free-form QA, which is still an open-ended topic [83]. In contrast, we use the FITB form where the questions are transformed in an easier way and the question types are more diverse. We use the Stanford NLP Parser [50] to obtain the syntactic structures of original video descriptions. We divide the questions into three categories, nouns (objects like food, animals, plants), verbs (actions) and phrases (short phrases, e.g., “play computer game”). Question templates are subsequently generated from noun phrases (NP)

and verb phrases (VP). Multiple words can be dropped to form the questions. During template generation, we eliminate prepositional phrases as they are mostly subjective. We use WordNet\* and NLTK† toolkits to identify word categories and choose a set of categories listed in Table 4.1. We visualize the distribution of words in each category using t-SNE [108] in Fig. 4.2. It shows that categories can be separated, where actions and objects have a clear margin.

**Candidate answer generation.** After question template generation, we obtained a question template and a correct answer, where distractors are still required to form the multiple-choice FITB dataset. We designed two different levels of difficulty for answering questions by altering the similarities between the correct answer and the distractors.

For each question in the easy task, we randomly chose three distractors in the same category from the same dataset. We thus have four candidates for the easy tasks. Words like “person” or “man” were filtered in advance, and words with a frequency of less than 10 were filtered following common practice.

Video clips in the same dataset can have totally different scenes, e.g., the web videos are very diverse in MEDTest 14 dataset. To generate more difficult questions, we selected hard negative distractors from phrases that are similar to the correct answer. We collected distractors not only from the video datasets, but also used annotations from Flickr8K [38], Flickr30K [132] and MS COCO [61] as description sources for the similarity comparison. We first parsed the annotations using the method described above and gathered about 8,000 phrases in total, resulting in an average length of 6.6 words per phrase. After the preprocessing, we encoded both the correct answers and the distractors with word2vec [69] and measured the similarity with cosine distance. For candidate phrases with length more than one, we average

---

\*<https://wordnet.princeton.edu>

†<http://www.nltk.org/>


the word2vec representation of each word [59, 62]. To avoid the ambiguity between the distractors and the correct answer (distractors should not be too close to the correct answer), we manually set an upper bound for the similarity between them. We chose the best threshold by sampling a few thousand candidates and checking the ambiguity by human labor. We discarded phrases that are too similar to the correct answer, and selected nine most similar distractors from the remaining phrases. We thus have ten candidates for the difficult task. We show examples of QA pairs of different categories and level of difficulty in Fig. 4.3.

Datasets	TACoS	MPII-MD	MEDTest 14	Combined
Verbs	268	869	671	2,925
Phrases	964	220	418	5,927
Animals	-	63	98	352
Food/Plant	62	129	174	598
Other objects	134	896	726	2,093

Table 4.1 : List of categories and number of collected words in three datasets. The right column shows the total number of words and phrases collected, including those from image domains such as MS COCO [61]


#### 4.2.2 Task Definitions and Analysis

In addition to describing the current clip, we introduce another two tasks which respectively infer the past and anticipate the future. In the task of describing the present, we use all three datasets for evaluation. For the other two tasks of past inference and future prediction, we perform experiments on the TACoS and MPII-MD datasets only because they are annotated in fine-grained clips. In these tasks, questions about the previous or next clip need to be answered for the given video




**Dog**

Category: Animal  
 Q: A/An \_\_\_\_ swims in a pool.  
 Easy distractors: - bee - horseback - bird  
 Hard distractors: - cat - **duck** - **dolphin** - clam - **goose** - bear - cow - penguin - elephant




**Pineapple**


Category: food/plant  
 Q: A man cutting a \_\_\_\_ in the food market.  
 Easy distractors: - snowball - popcorn - seed  
 Hard distractors: - grapefruit - **cucumber** - **broccoli** - **lemon** - orange - strawberry - wheat - watermelon



Category: Actions  
 Q: He \_\_ her.  
 - hugs  
 - **kisses**  
 - beats  
 - runs towards



Category: Actions  
 Q: Someone walks toward the fence to \_\_\_\_ Someone.  
 - **greet**  
 - hit  
 - knock  
 - laugh



Category: Phrases  
 Q: Two boys \_\_\_\_ in a bedroom.  
 - **play with toys**  
 - swing  
 - pick up empty recycle bin  
 - shave facial hair

Figure 4.3 : Examples of QA pairs for different categories and levels of difficulty. The words colored in green are correct answers, and the difficult candidates are marked in red.

clip. Note that for tasks of describing the past or the future, only the current clip is given and the model has to reason temporal structures based on the given clip. We restrict the past and future so that they are not too far away from the current clip and typically we choose the clip immediately before or after the given clip, where the time interval is less than 10 seconds.

We introduce two levels of questions for each task. For simplicity, we denote our tasks as *Past-Easy*, *Present-Easy*, *Future-Easy*, *Past-Hard*, *Present-Hard* and



*Future-Hard*. We randomly partitioned the dataset into three non-overlapping subsets, one for training the models, one for validation (hyper-parameter tuning), and one for testing the performance. As the performance may vary in accordance with different subset partitions, we randomly partitioned the dataset three times independently. We thus have three splits for each task. The models are trained individually on the training set of each split. The parameters are not shared between splits.

We now show the statistics of our dataset.

**Question types.** We visualize the distribution of question types in Fig. 4.4. It shows that there are more questions about objects than actions. The distribution of each question type also varies across different datasets, e.g., there are more questions about food and plant in the TACoS dataset.

**Popular questions and answer.** We show the top-5 popular questions and answers in Table 4.2. Each dataset has different popular questions and answers, from which we observe that the TACoS dataset asks more about cooking and it is less diverse than the other two. Questions such as “someone \_\_\_”, “people \_\_\_” can have richer answers than “the person rinse the \_\_\_”.

**Question and answer lengths.** The lengths of questions and answers are shown in Fig. 4.5 and Fig. 4.6, respectively. We can see that the length of most questions are in the range from 5 to 12, while most answers only have one word and in TACoS and MPII-MD, about 10% answers have two words.

### 4.3 The Proposed Approach

To answer questions about present, past and future, we first introduce an encoder-decoder framework to represent context. We then map the visual representation to a semantic embedding space and learn to rank the candidate answers.

	Rank	TACoS (%)
Questions	1	the person get out a _____. (1.11)
	2	the person _____ a knife. (0.52)
	3	the person rinse the _____. (0.46)
	4	he _____ cut board. (0.32)
Answers	1	take (0.05)
	2	cut (0.04)
	3	wash (0.03)
	4	rinse (0.02)
	Rank	MPII-MD (%)
Questions	1	someone _____. (1.08)
	2	he _____. (0.26)
	3	someone _____ someone. (0.17)
	4	someone _____ up. (0.10)
Answers	1	look (0.02)
	2	eye (0.01)
	3	hand (0.01)
	4	head (0.01)
	Rank	MEDTest 14 (%)
Questions	1	child _____ football. (0.14)
	2	guy _____ ping pong. (0.07)
	3	a man shave his _____. (0.06)
	4	one _____ lady dancing indoors. (0.04)
Answers	1	play (0.04)
	2	dog (0.03)
	3	dance (0.03)
	4	kid (0.02)

Table 4.2 : Top-4 most popular questions and answers in each dataset. The numbers in the parentheses are the percentages of questions and answers

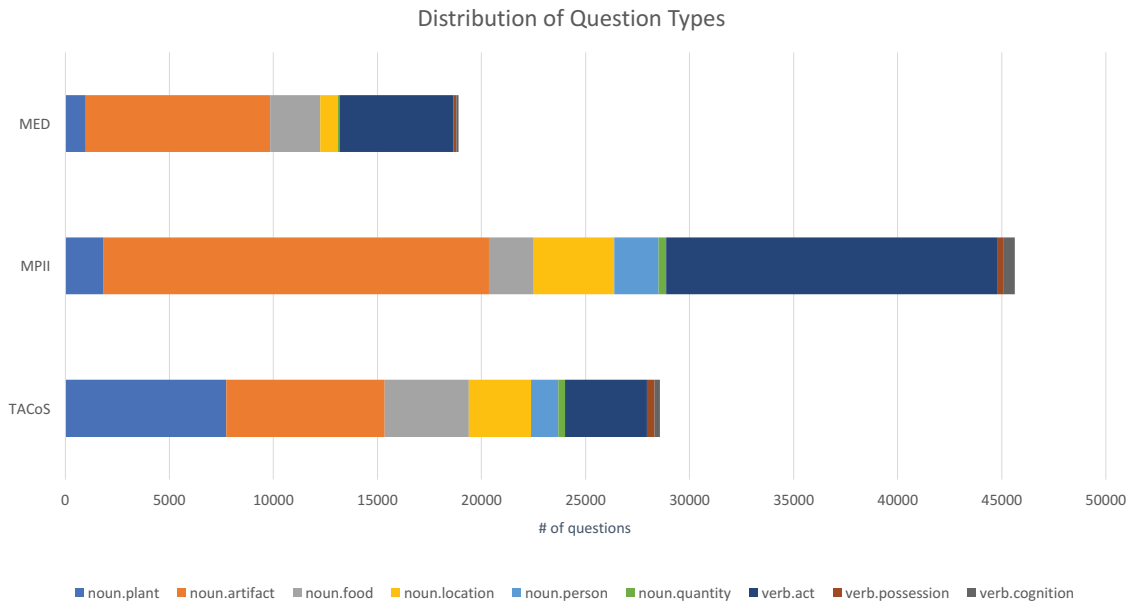


Figure 4.4 : Distribution of question types for each dataset

### 4.3.1 Learning to Represent Video Sequences

In this section, we describe our model for learning temporal context. We present an encoder-decoder framework using Gated Recurrent Unit (GRU) [15]. Compared with LSTM [37], GRU is conceptually simpler with only two gates (update gate and reset gate) and no memory cells, while the performance on the sequence modeling task is as good as LSTM [16]. Note that we trained our model with LSTM as well, but it performs worse than the model trained with GRU. With GRU, we can achieve mAP of 24.9% on the MEDTest 14 100Ex classification task, whereas we can only achieve 20.4% with LSTM.

**Gated Recurrent Unit.** Denote  $f_i^1, f_i^2, \dots, f_i^N$  as the frames in a video  $v_i$ , where  $N$  is the number of frames sampled from the video. At each step  $t$ , the encoder generates a hidden state  $\mathbf{h}_i^t$ , which can be regarded as the representation of sequence  $f_i^1, f_i^2, \dots, f_i^t$ . Thus the state of  $\mathbf{h}_i^N$  encodes the whole sequence of frames. States in GRU [15] are calculated as follows (the video subscript  $i$  is dropped for simplicity):

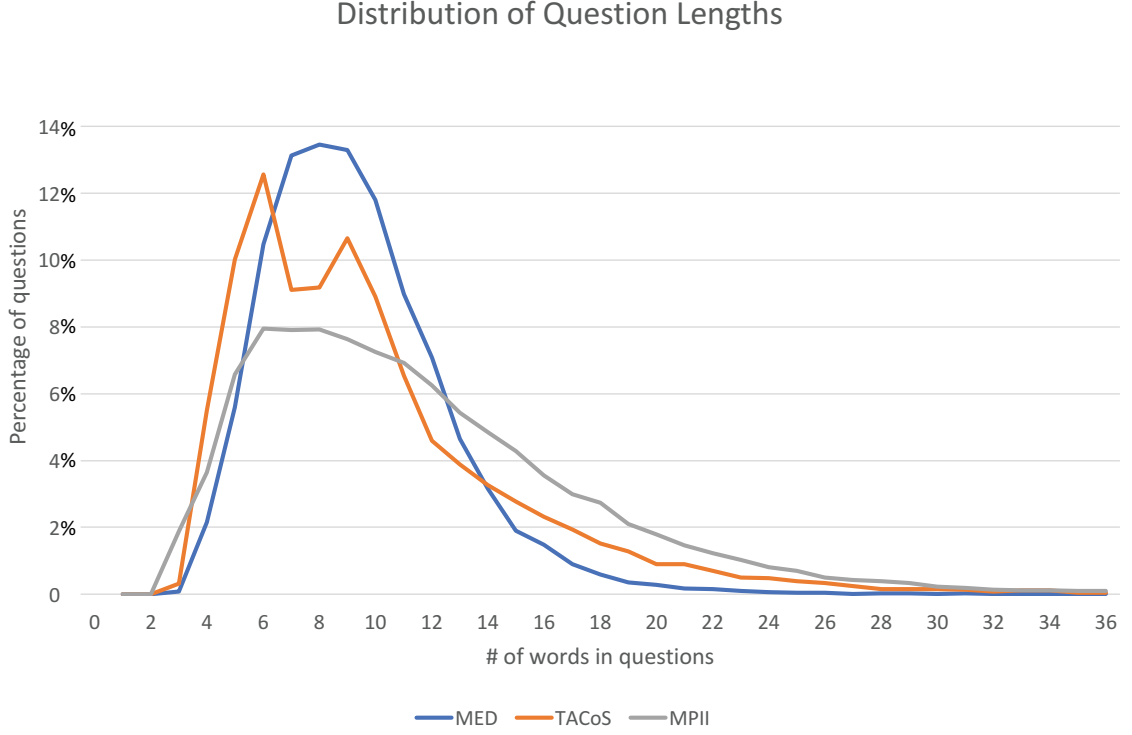


Figure 4.5 : Distribution of question lengths for each dataset

$$\mathbf{r}^t = \sigma(W_{xr}\mathbf{x}^t + W_{hr}\mathbf{h}^{t-1}), \quad (4.1)$$

$$\mathbf{z}^t = \sigma(W_{xz}\mathbf{x}^t + W_{hz}\mathbf{h}^{t-1}), \quad (4.2)$$

$$\bar{\mathbf{h}}^t = \tanh(W_{x\bar{h}}\mathbf{x}^t + W_{h\bar{h}}(\mathbf{r}^t \odot \mathbf{h}^{t-1})), \quad (4.3)$$

$$\mathbf{h}^t = (1 - \mathbf{z}^t) \odot \mathbf{h}^{t-1} + \mathbf{z}^t \odot \bar{\mathbf{h}}^t, \quad (4.4)$$

where  $\mathbf{x}^t$  is the input,  $\mathbf{r}^t$  is the reset gate,  $\mathbf{z}^t$  is the update gate,  $\mathbf{h}^t$  is the proposed state and  $\odot$  is element-wise multiplication. We use the same architecture for the decoder as for the encoder, but its hidden state of  $\mathbf{h}^0$  is initialized with the hidden state of the last time step  $N$  in the encoder. We construct our GRU encoder-decoder model (Fig. 4.7) in a similar fashion to [96]. Besides reconstructing the input frames, we also train another two models which are asked to reconstruct the future frames (Fig. 4.7 left) and past frames (Fig. 4.7 right), respectively. Our proposed models

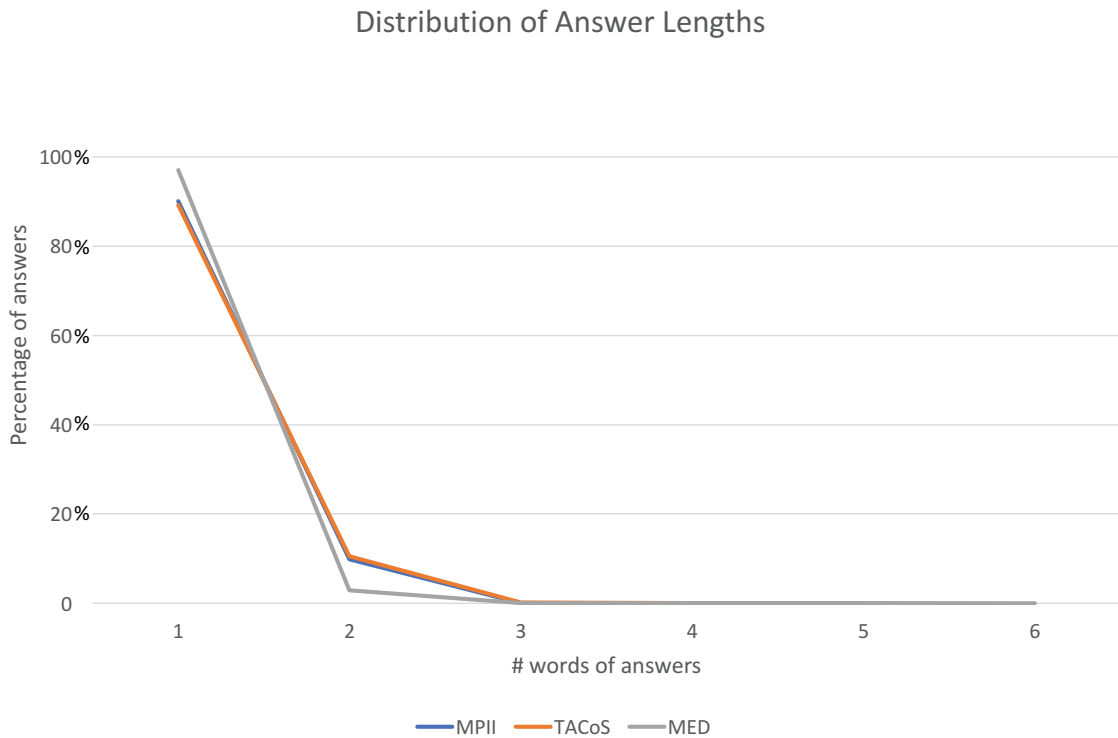


Figure 4.6 : Distribution of answer lengths for each dataset

are capable of learning good features as the network is optimized by minimizing the reconstruction error. To achieve good reconstruction, representation passed to the decoder should retain high level abstraction of the target sequence. Note that our three models are learned separately, and the encoder and decoder weights are not shared across models of past, present and future.

**Training.** We first train the encoder-decoder models in an unsupervised way using videos collected from a subset of the MED dataset [2] (excluding the MEDTest 13 and MEDTest 14 videos) which consists of 35,805 videos having a duration of 1,300 hours. The reason for choosing MED dataset as a source for temporal context learning is that videos in the MED dataset are longer in duration and contain complex and profound events, actions and objects for learning. We collect additional data to our target task datasets for the purpose of learning a more powerful model, and practically, it is difficult to train a model from scratch using such a small

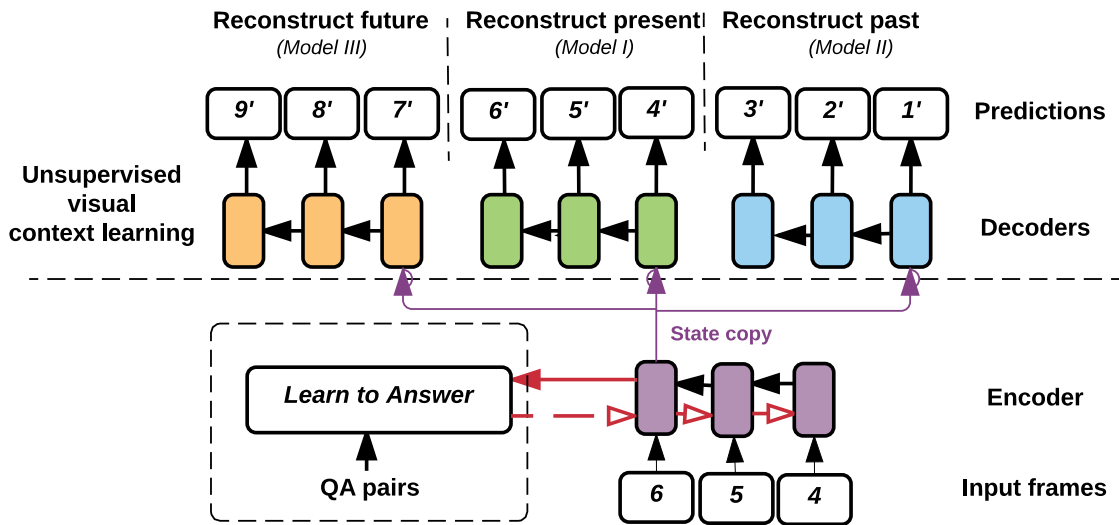


Figure 4.7 : The encoder-decoder model (top): encoder state of last time step is passed to three decoders for reconstruction. Learn to answer questions (bottom): encoder state of last time step is passed to the ranking module which selects an answer based on the visual information

dataset as TACoS, which has only 127 cooking videos. As the video frames have high correlations in short range, we sample frames at the frame rate of 1 fps. We use a time span of 30 seconds and set the unroll length  $T$  to 30 for the present model (Model 1), and 15 for both the past model (Model 2) and the future model (Model 3).

For the input to the GRU model, we use ConvNet features extracted from GoogLeNet [99] with Batch Normalization [39], which was trained from scratch with the ImageNet 2012 dataset [87] and we keep the ConvNets part frozen during RNN training.

We now explain our network structures and training process in detail. As three models are trained with the same hyper-parameters, we take Model 1 as an example. In our case, reconstruction error is measured by  $\ell_2$  distance between the predicted representation and the target sequence. We reverse the target sequences in the

present reconstruction scenario which, as indicated in [98], reduces the path of the gradient flow. We set the size of the GRU units to 1,024 and two GRU layers are stacked. Our decoders are conditioned on the inputs, and we apply Dropout with the rate of 0.5 at connections between the first GRU layer and the second GRU layer as suggested by [137] to improve the generalization of the neural network. We initialize  $\mathbf{h}^0$  for the encoder with zeros, while the weights in the input transformation layer are initialized with a uniform distribution in  $[-0.01, 0.01]$  and recurrent weights have uniform distribution in  $[-0.05, 0.05]$ . We set the mini-batch size to 64 and clip gradient element-wise at the norm of  $1e-4$ . Frame sequences from different videos are sampled in each mini-batch. The network is optimized by RMSprop [103], which scales the gradient by a running average of gradient norm. The model is trained by the Torch library [17] on a single NVIDIA Tesla K20 GPU, and it takes approximately one day for the models to converge and complete the training.

**Inference.** At inference time, we feed the ConvNet features extracted from GoogLeNet to the encoder, and obtain the video features from the hidden states. For each video clip, we initialize  $\mathbf{h}^0$  to zeros, and pass the current hidden state to the next step until the last input. We then average the hidden states at each time step as the final representation.

### 4.3.2 Dual-Channel Learning to Rank

Visual information and textual information are mutually beneficial. We present the proposed dual-channel learning to rank algorithm by appropriately integrating information, including sentences, words, and visual cues, within a joint learning framework to maximize the mutual benefit. We jointly model two channels, i.e., word channel and sentence channel, for learning.

[49] recently proposed skip-thought vectors to encode a sentence into a compact vector. The model uses an RNN encoder to encode a sentence and another two RNN

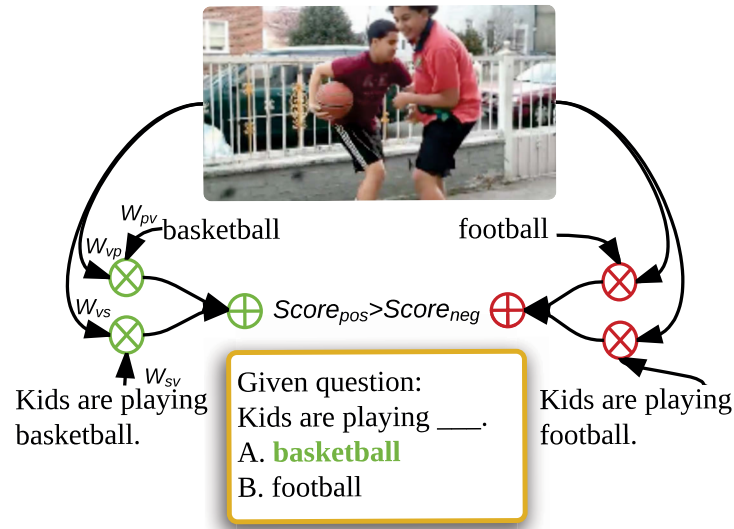


Figure 4.8 : Illustration of dual-channel learning to rank

decoders are asked to reconstruct the previous sentence and the next sentence. It was trained using BookCorpus dataset [143] which consists of 11,038 books, 74,004,228 sentences and 984,846,357 words. The skip-thought vectors model performs well on many different natural language processing (NLP) tasks. We utilize the combine-skip model to encode sentences. For more details, please refer to [49].

We first formulate the problem of multiple-choice question answering. Given  $N$  questions with blanks together with corresponding videos, and  $K$  candidate answers for each question, we denote each question as  $q_i, i \in 1, \dots, N$ , candidate answers for question  $q_i$  as  $p_{ij}, j \in 1, \dots, K$  and the ground truth for question  $q_i$  as  $p'_i$  with index  $j'_i$ . For each question  $q_i$ , let  $s_{ij}$  be the sentence formed by filling the blank of question  $q_i$  with candidate  $p_{ij}$ . For example, filling in the template of “A/An \_\_\_\_ swims in a pool” shown in Fig. 4.3 with candidate “dog”, we can form the sentence “A dog swims in a pool”, and the false description “A horseback swims in a pool” is generated by “horseback”.

Given  $q_i$ , we introduce a dual-channel ranking loss (also illustrated in Fig. 4.8)



that is trained to produce higher similarity for the visual context and representation vector of the correct answer  $p'_i$  than other distractors  $p_{ij}, j \neq j'_i$ . We define our loss as:

$$\min_{\boldsymbol{\theta}} \sum_{\mathbf{v}} \sum_{j \in K, j \neq j'} \lambda \ell_{word} + (1 - \lambda) \ell_{sent}, \lambda \in [0, 1], \quad (4.5)$$

with

$$\begin{aligned} \ell_{word} &= \max(0, \alpha - \mathbf{v}_{\mathbf{p}}^T \mathbf{p}_{j'} + \mathbf{v}_{\mathbf{p}}^T \mathbf{p}_j), \\ \ell_{sent} &= \max(0, \beta - \mathbf{v}_{\mathbf{s}}^T \mathbf{s}_{j'} + \mathbf{v}_{\mathbf{s}}^T \mathbf{s}_j), \end{aligned} \quad (4.6)$$

where  $\mathbf{v}_{\mathbf{p}} = W_{vp} \mathbf{v}$ ,  $\mathbf{v}_{\mathbf{s}} = W_{vs} \mathbf{v}$  and  $\mathbf{p}_j = W_{pv} \mathbf{y}_j$ ,  $\mathbf{s}_j = W_{sv} \mathbf{z}_j$  (for simplicity we drop the subscript  $i$ ).  $\mathbf{v}$  is the vector learning from our GRU encoder-decoder model for video clip  $v_i$ ,  $\mathbf{y}_j$  is the average of word2vec vectors for each word in candidate  $p_{ij}$ ,  $\mathbf{z}_j$  is the skip-thought vector for description  $\mathbf{s}_{ij}$ . We constrain these feature representations to be in unit norm.  $\boldsymbol{\theta}$  denotes all the transformation parameters that need to be learned in the model,  $W_{vs}$  and  $W_{vp}$  are transformations that map the visual representation to the semantic joint space, while  $W_{sv}$  and  $W_{pv}$  transform the semantic representation. Note that  $W_{xx}$  can be a linear transformation or multi-layer neural network with hidden units.

**Training.** During the training procedure, we sample false terms from negative candidates and practically stop summing after finding the first margin-violating term [27]. Empirically, we select a sentence embedding dimension of 500 and word embedding of 300. The model is trained by stochastic gradient descent (SGD) by simply setting the learning rate  $\eta$  as 0.01 and the momentum as 0.9. In practice, we set the margins  $\alpha$  and  $\beta$  to 0.2, and  $\lambda$  is cross-validated on the held-out validation set.

**Inference.** We learn the weight of the transformations at the training stage and at

inference time, we calculate the following score for each candidate,

$$score = \lambda \mathbf{v}_p^T \mathbf{p}_j + (1 - \lambda) \mathbf{v}_s^T \mathbf{s}_j \quad (4.7)$$

The candidate with the highest score is our answer.

## 4.4 Results

### 4.4.1 Evaluation of Describing the Present

In this section, we evaluate our model in the task of describing the present. We first present text-only baselines to show that visual information is an important cue in this task. We also demonstrate the effectiveness of our ranking method by comparing it with Canonical Correlation Analysis (CCA), normalized CCA (nCCA) and then conduct evaluations of dual-channel learning. We then show the biases in the answers.

**Text-only baselines.** We show that visual information is an important cue by presenting text-only baselines where only question templates and answers are provided. In the first baseline, we choose the candidate which is most similar to the question as our answer. We average word2vec representation for each word to get the phrase representation for both questions and candidates. We measure the similarity by using the dot product. The candidate with the highest score will be the answer. We call it the *Simple word2vec* baseline. We also encode the question and candidates with skip-thought vectors which is our *Simple skip-thought* baseline.

In the second baseline, we use a LSTM decoder to generate the probability of each word in the vocabulary while the encoder encodes words in the question [49, 83, 65, 28]. We choose the answer with the highest probability among the candidates. We call it the *LSTM Generator* baseline. A *GRU Generator* can also be introduced by replacing the LSTM cell with GRU cell. We use RNN cell size of 512 and set the number of RNN layer to 1 in the experiment. We early stop the

training process when the performance on the validation set does not improve.

We compare the baselines with our model in the *Present-Easy* task and the results are shown in Table 4.3. It shows our visual model outperforms the text-only baselines with a large margin. The *Simple* baseline performs much worse than other methods as our dataset is constructed by choosing similar candidates. The results show that visual information is important in the task. We also observe that *Simple word2vec* performs better than *Simple skip-thought* which indicates that skip-thought might not encode short phrases effectively. *LSTM Generator* and *GRU Generator* have comparable results when modeling language.

**Comparison with models with visual cue.** We first compare our dual-channel ranking approach with CCA which computes the direction of maximal correlation between a pair of multi-dimensional variables. [32] normalized the correlation by dimension reduction before linear CCA and [135] found it also beneficial to image QA. To learn CCA, we separately train two embedding layers. The first CCA maps the sentence description to the visual semantic joint-embedding space and the second CCA maps the correct answer to the joint space. To answer multiple-choice questions, we embed each candidate and select the answer that is most similar to the video clip using Equation 4.7. We conduct cross-validation for choosing the weight to combine two embeddings. For nCCA, we also cross validated the dimension of the projected space on the validation set.

We restrict the input features to be the same for both methods. For visual representation, we average the frame-level features extracted from the last fully connected layer of GoogLeNet. For semantic representation, we use the same method as described in Section 4.3.2, where sentences are encoded by skip-thought vectors, and word2vec is used for word representation.

Note that in CCA, the two embedding matrices are learned separately at training

Methods	TACoS (%)	MPII-MD (%)	MEDTest 14 (%)
Simple skip-thought	35.4	25.8	26.4
Simple word2vec	36.9	34.3	35.1
LSTM Generator	54.6	60.3	69.3
GRU Generator	54.4	61.0	68.9
Question type Prior	63.3	47.7	63.0
CCA	65.1	41.6	63.2
nCCA	69.2	42.8	64.2
Post-hoc	67.0	35.1	55.7
Visual + Answer-only	63.1	53.8	73.0
Our dual-channel loss	76.3	71.9	81.0

Table 4.3 : Comparison between our model and other baselines in the *Present-Easy* task. Except the text-only baselines, “Mean-GoogLeNet” is used as visual features for all approaches

time while the weights of two embeddings are introduced at the validation stage. The method of late fusing sentence and word descriptions is different from our dual-channel ranking approach, which integrates sentence and word representations during training time and learns to adjust the embeddings accordingly. We demonstrate the effectiveness of our dual-channel ranking method in Table 4.3.

As can be seen, nCCA consistently better than CCA and our method outperforms nCCA by a large margin. We believe it is because our objective function learns to integrate two representations, while nCCA uses a fixed embedding matrix during semantic weight learning. In addition, nCCA eliminates negative terms during training, and as multiple-choice question answering is required to select an

		Past			Present			Future		
		Conv	Ours	Improv	Conv	Ours	Improv	Conv	Ours	Improv
TACoS	Easy	74.8	<b>78.3</b>	3.5	76.3	<b>79.7</b>	3.4	76.4	<b>78.7</b>	2.3
	Hard	62.7	<b>64.7</b>	2.0	65.5	<b>67.1</b>	1.6	64.5	<b>67.3</b>	2.8
MPII-MD	Easy	66.8	<b>72.1</b>	5.3	72.0	<b>74.2</b>	2.2	68.7	<b>73.6</b>	4.9
	Hard	45.6	<b>47.0</b>	1.4	47.3	<b>48.2</b>	0.9	46.9	<b>48.0</b>	1.1

Table 4.4 : Comparisons between ConvNets (Conv) and our model for past, present and future modeling. Relative improvements for each task are also listed

answer from candidates at testing time, ranking loss is more suitable for modeling the problem.

Following [135], we also use the video captioning model to generate the description of the video. We then compute the cosine similarity between the skip-thought representation of the description generated and the description filled by all the candidates. The most similar candidate is the our answer. We call it the *Post-hoc* baseline. We use GRU cell of cell size 512 in the captioning model and the results are shown in Table 4.3. We observe that the *Post-hoc* baseline is comparable to CCA on TACoS dataset but worse on MPII-MD and MEDTest 14 datasets.

**Effectiveness of dual-channel loss.** We now show the effectiveness of using two channels for learning. The result of how the integration of two representations influences performance is shown in Fig. 4.9. As can be seen, it is beneficial to integrate word representations during training, and sentences are weighted more than words. This is because our visual features represent more global abstraction, which corresponds to sentence representation, whereas specific object features corresponding to word representation have not been considered in this work. We will explore this direction in detail in future works.

**Biases in answers.** [40] found that there is a strong bias in Visual7W dataset [142].

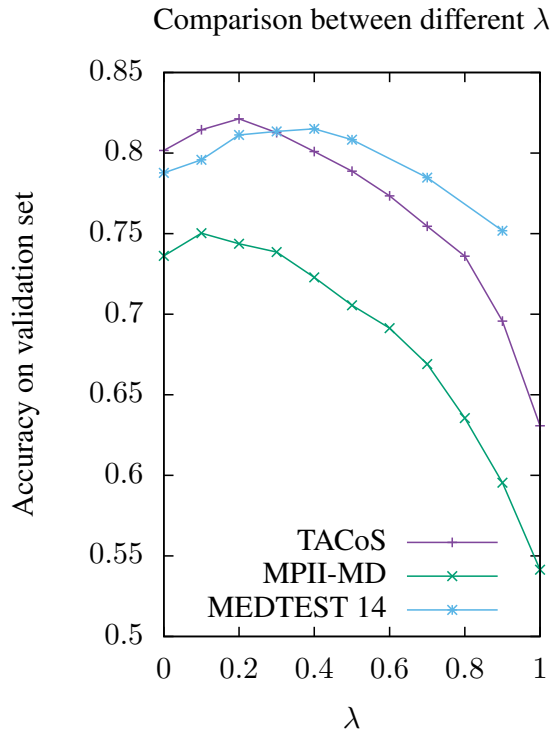


Figure 4.9 : The effectiveness of dual-channel learning to rank. We conduct experiments on the *Present-Easy* task to showcase.  $\lambda = 0$  corresponds to using the sentence channel only and  $\lambda = 1$  corresponds to using the word channel only

$\lambda = 1$  in Fig. 4.9 indicates that there are biases in the answers. We report the results of  $\lambda = 1$  in the *Present-Easy* task in Table 4.3 (*Visual + Answer-only*). We also conduct another experiment by introducing the question type prior where question type is known before answering the question. We answer the questions by selecting the most common candidate (measured by the answer frequency in the training set) of the question type. If the candidate does not belong to the question type, we will not choose it as the answer. We call it the *Question type Prior* baseline. The results show that *Question type Prior* baseline has high performance, however, the question type is a strong bias as it could be a lot easier after knowing what the question is asking about.

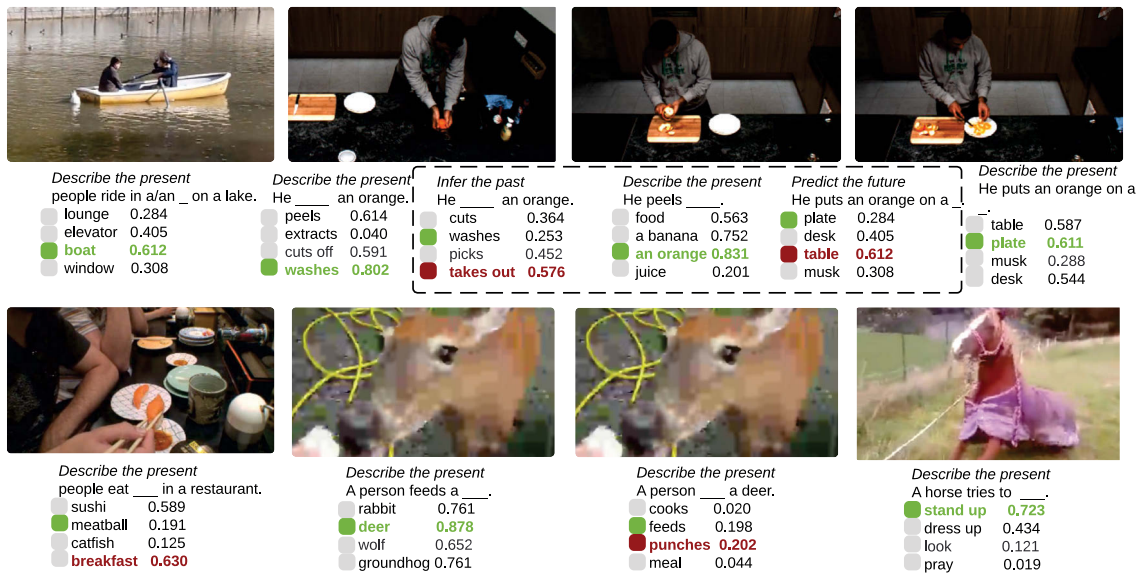


Figure 4.10 : Example results obtained from our model. Each candidate has a score corresponding to a clip. Correct answers are marked in green while failed cases are in red

#### 4.4.2 Evaluation of Inferring the Past and Predicting the Future

We first show the results of our GRU model in all tasks in Table 4.5. We illustrate some of the experiment results using our GRU model in Fig. 4.10 and show a number of wrong answers as well.

To show the effectiveness of our encoder-decoder approach in modeling video sequences, we compare our present model with a strong baseline - averaging frame-level features from GoogLeNet (Mean-GoogLeNet). We compare two representations by placing the visual input to our dual-channel ranking objective with the Mean-GoogLeNet or our GRU features. Note that the comparison is reasonable as both features have the same dimension of 1,024 and we use the same transformation layer and same hyper-parameters during training. The results are shown in Table 4.4. From the results, we make the following observations:

- (1) The GRU model outperforms Mean-GoogLeNet model in all cases, and per-

forms relatively better than Mean-GoogLeNet in the tasks of inferring the past and predicting the future compared to describing the present. For the MPII-MD Easy dataset, the GRU model performs better than the Mean-GoogLeNet model by 2.2% in describing the present, and around 5.0% improvements are achieved for the past and future inferring. It shows the effectiveness of our GRU encoder-decoder framework in modeling temporal structures in videos. As our models are trained to reconstruct past and future sequences, they can represent the past and future in a more reasonable way than the Mean-GoogLeNet models. Our results also demonstrate the ability of our GRU model to capture a wider range of temporal information than the Mean-GoogLeNet models. ConvNets trained from still frames can model temporal structures if the objects in a scene do not change too much in short intervals (one example would be in Fig. 4.1, where “cucumber” occurs in both the current clip and the future clip). However, in modeling longer sequences, ConvNets fail to make predictions due to lack of context.

(2) Our model can achieve better results for future prediction than for past inference. For future prediction, we feed the input frames in the order of 4, 5, 6 and the decoder is asked to reconstruct the frames in the order 7, 8, 9. We feed the same input for past inferring, but ask the decoder to reconstruct the target sequence of 1, 2, 3. As the future prediction model has shorter term dependencies than the past inferring model, it will be easier for the future prediction model to learn the temporal dependencies, which is consistent with the observations and hypothesis in [98].

#### 4.4.3 Limitations and Future Work

Although our results on question answering for video temporal context are encouraging, our model has multiple limitations. First, our model is only aware of context for at most 30 seconds (the longest unroll length). An alternative flexible



and promising approach would be to incorporate the attention mechanism [7] to learn longer sequences of context in videos. Additionally, our model sometimes fails to answer questions about detailed objects, due to lack of local visual features, i.e., region-level, bounding boxes based representation. We would like to integrate object detection ingredients to localize objects for better visual understanding. Lastly, we fixed the sentence and word representation learning in this work. Learning visual and language representations simultaneously remains an open problem, as indicated in [27].

## 4.5 Conclusion

Unlike video captioning tasks which generate a generic and single description for a video clip, we introduced a temporal structure modeling approach for video question answering. We utilized an encoder-decoder model trained in an unsupervised way for visual context learning and propose a dual-channel learning-to-rank method to answer questions. The proposed method is capable of modeling video temporal structure in a longer time range. We evaluated our approach on three datasets which have a large number of videos. The new approach outperforms the compared baselines, and achieves encouraging question answering results.

	TACoS			
	split 1	split 2	split 3	mean
Past-Easy	78.1	78.3	78.5	78.3
Past-Hard	65.8	64.4	63.9	64.7
Present-Easy	79.1	81.9	78.1	79.7
Present-Hard	66.9	66.2	68.2	67.1
Future-Easy	76.9	79.6	79.7	78.7
Future-Hard	66.1	65.8	69.9	67.3
	MPII-MD			
	split 1	split 2	split 3	mean
Past-Easy	72.4	72.0	72.0	72.1
Past-Hard	47.0	47.0	46.9	47.0
Present-Easy	75.5	74.6	72.4	74.2
Present-Hard	47.4	49.0	48.3	48.2
Future-Easy	75.9	73.3	71.7	73.6
Future-Hard	47.1	48.8	48.1	48.0
	MED			
	split 1	split 2	split 3	mean
Past-Easy	-	-	-	-
Past-Hard	-	-	-	-
Present-Easy	83.7	83.0	82.8	83.2
Present-Hard	63.0	63.9	62.3	63.1
Future-Easy	-	-	-	-
Future-Hard	-	-	-	-

Table 4.5 : Results of our GRU models on inferring past and predicting the future for TACoS and MPII-MD datasets

## Chapter 5

# Few-Shot Object Recognition from Machine-Labeled Web Images

### 5.1 Introduction

In this chapter, I present an “abstraction memory” based framework for few-shot learning, building upon machine-labeled image annotations. The method takes large-scale machine-annotated dataset (e.g., OpenImages) as an external memory bank. In the external memory bank, the information is stored in the memory slots in the form of key-value, in which image feature is regarded as the key and the label embedding serves as the value. When queried by the few-shot examples, the model selects visually similar data from the external memory bank and writes the useful information obtained from related external data into another memory bank, i.e. abstraction memory. Long Short-Term Memory (LSTM) controllers and attention mechanisms are utilized to guarantee the data written to the abstraction memory correlates with the query example. The abstraction memory concentrates information from the external memory bank to make the few-shot recognition effective. In the experiments, we first confirm that the model can learn to conduct few-shot object recognition on clean human-labeled data from the ImageNet dataset. Then, we demonstrate that with the model, machine-labeled image annotations are very effective and abundant resources for performing object recognition on novel cate-

---

This chapter is based on the joint work with Zhongwen Xu and Yi Yang (Xu et al. 2017 [129]), presented primarily as it appears in the CVPR 2017 proceedings. Zhongwen Xu and Linchao Zhu are equally contributed to this work as indicated in the original paper.

gories. Experimental results show that the proposed model with machine-labeled annotations achieves great results, with only a 1% difference in accuracy between the machine-labeled annotations and the human-labeled annotations.

Innovations in the architecture of Convolutional Neural Networks (ConvNets) [60, 99, 92, 35] have resulted in tremendous improvements in image classification in the past few years. With the increase in the capacity of neural networks, the demand for more labeled data in richer categories is rising. However, it is impractical and very expensive to manually label a dataset 10 times larger than ImageNet. This prompts us to design a new paradigm that can utilize the machine-labeled image annotations to enable rapid learning from novel object categories. Figure 5.1 illustrates the proposed task. Our major question in this work is as follows: Can we use machine-labeled web image annotations to rapidly conduct object recognition for *novel categories* with *only a handful of examples*?

We propose a new memory component in neural networks, namely *abstraction memory*, to concentrate information from the external memory bank, e.g., large-scale object recognition datasets like ImageNet [87] and OpenImages [53], based on few-shot image queries. Previous methods which attempt to learn from different categories or datasets usually use a larger dataset for pre-training and then conduct fine-tuning on a relatively small dataset. The information of the large datasets is encoded in the learnable weights of the neural networks. In contrast to previous works, our model utilizes a content-based addressing mechanism with a Long Short-Term Memory (LSTM) controller to automatically decide where to read from and where to write into the memory. The neural network applies a soft attention mechanism [7] to the query image to find the appropriate information to read from the external memory and write into another memory. The abstraction memory records helpful information for the specific few-shot object recognition, so that the classification network can utilize readouts from the abstraction memory to recognize the objects

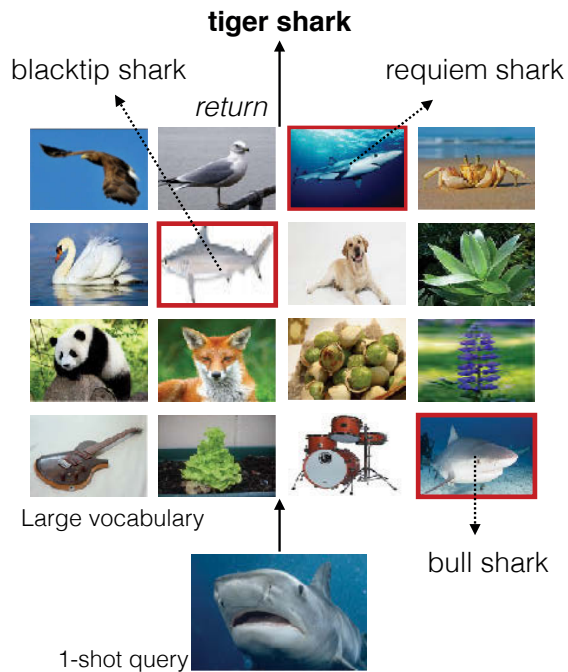


Figure 5.1 : Given a large vocabulary of labels and their corresponding images, we conduct few-shot learning on a novel category which is not in the vocabulary and only has a handful of positive examples. The image examples in the vocabulary are stored in the external memory of our model, and the image example from the novel category queries the external memory. Our model returns helpful information according to visual similarity and LSTM controllers. The retrieved information, i.e., visual features and their corresponding labels, are combined to classify this query image example.

from novel categories.

Previous methods only discover the relationship of the word embeddings [43, 69] between the category labels, whereas we fully utilize the visual similarity between the examples of few-shot categories and the external memory bank to make the proposed framework more robust to noisy labels. If the external memory data is inconsistent with its label, this sample will be rejected during the visual matching

process. This property makes the use of a large-scale machine annotated dataset, e.g., OpenImages [53] feasible. The machine-labeled annotations for images could be predicted by off-the-shelf ConvNet models (e.g., ResNets [35]), but although these annotations are reasonably good, they are not perfect. In this scenario, the external dataset can also consist of images obtained by querying keywords in search engines (e.g., Google Images), and images crawled from social image sharing sites (e.g., Flickr). In the experiment section, we show that the results of our proposed method using machine-annotated data differ from human-labeled data by a minor gap  $\approx 1\%$ .

When novel categories arrive, the network accesses and queries the external memory, retrieves the related information, and writes into abstraction memory. We organize the memory in the data structure **key:value**, which was first proposed in Key-Value Memory Networks (KV-MemNNs) [70]. We note that we have the implementation of our model, including LSTM controllers, abstraction memory, and reading mechanisms, differs significantly from KV-MemNNs. Moreover, KV-MemNNs were developed in natural language understanding area, and their memory access is limited to the most recent few sentences. We extend the key-value storage concept into computer vision applications by novel modifications to enable scalability. We formulate the image embedding as the **key** and the word embedding of the annotated label as the **value**. The additional memory for abstraction extracts information from the external memory and learns task-specific representation for the few-shot learning while maintaining efficiency.

Our contributions are as follows.

1. We propose a novel task for learning few-shot object recognition on machine-labeled image annotations. We demonstrate that with sufficiently reliable machine-labeled annotations, it is possible to achieve excellent performance

with a only a minor deviation in accuracy (about 1%) compared to learning from human-labeled annotations;

2. We propose the incorporation of a novel memory component, namely abstraction memory, into the Memory Networks [123] structure. The abstraction memory alleviates the time-consuming content-based addressing of the external memory, enabling the model to be scalable and efficient;
3. We utilize both visual embeddings and label embeddings in a form of key-value to make the system robust to imperfect labeling. This enables the model to learn from the machine-labeled web images to obtain rich signals for visual representation, which is very suitable for real-world vision application. We conduct few-shot learning of unseen visual categories, making rapid and accurate predictions without extensive iterations of positive examples.

We demonstrate the advantages of our method over state-of-the-art models such as Matching Networks [115], KV-MemNNs [70], Exemplar-SVMs [66], and Nearest Neighbors [10] on few-shot object recognition tasks.

## 5.2 Proposed Approach

### 5.2.1 Preliminaries

We briefly introduce some technical preliminaries on Memory Network variants before discussing our proposed model.

Memory Networks (MemNNs) [123] are a new family of learning models which augment neural networks with *external memory*. The major innovation of Memory Networks is the long-term memory component  $\mathcal{M}$ , which enables the neural networks to reason and access the information from long-term storage. End-to-End Memory Networks (MemN2N) [97] implement Memory Networks in a continuous

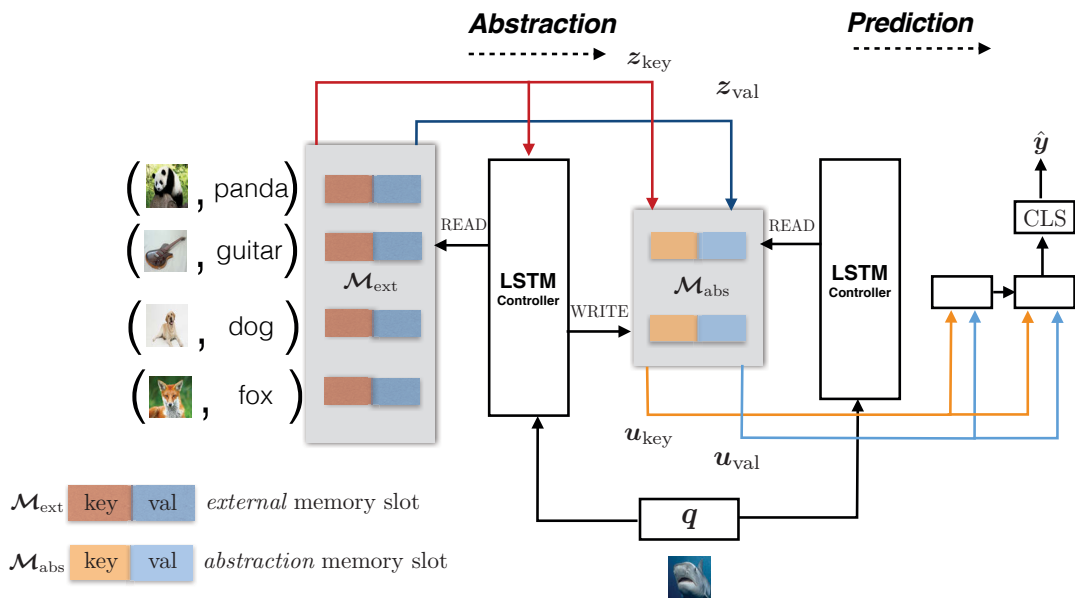


Figure 5.2 : An illustration of our proposed model. Best viewed in color.

form, so that end-to-end training becomes feasible. The recently proposed Key-Value Memory Networks (KV-MemNNs) [70] extend MemNNs [123] and MemN2N [97] with structural information storage in the memory slots. Instead of having only single vector representation in the memory component, as in MemN2N, KV-MemNNs make use of pairs of vectors in the memory slots, i.e., **key**: **value**. The incorporation of the structural storage of the Key-Value form into the memory slots brings much more flexibility, which enriches the expressive power of the neural networks. The Key-Value property makes information retrieval from the external memory natural.

The Memory Network variants (MemNNs, MemN2N, and KV-MemNNs) have been proposed for natural language understanding, and researchers often only validate these models on question answering tasks such as bAbI tasks [122].

### 5.2.2 Model Overview

In this work, we propose a novel Memory Networks architecture to tackle the few-shot visual object recognition problem. It retains the key-value structure, but



in contrast to KV-MemNNs, we utilize Long Short-Term Memory (LSTM) as a “controller” when accessing and writing to memory. Moreover, we introduce a novel memory component, namely *abstraction memory*, to enable task-specific feature learning and obtain scalability. The distinct nature of our proposed abstraction memory makes the neural network “remember” the ever present external memories, analogous to the memory cell  $\mathbf{c}$  in LSTMs but much more expressive. The incorporation of abstraction memory enables stochastic external memory training, i.e., we can sample batches from the a huge external memory pool. In contrast to our work, existing Memory Networks limit their access to external memory to a very small number, e.g., MemN2N limit their access to external memory to the most recent 50 sentences [97].

The overview of our model is shown in Figure 5.2. The whole procedure of our proposed model is illustrated as follows. Note that we re-formulate `key: value` as (key, value) in the rest of this work.

$$\mathbf{q}, \mathcal{M}_{\text{ext}} = \text{EMBED}(I, \{\mathcal{I}_{\text{web}}, \mathcal{L}_{\text{web}}\}) \quad (5.1)$$

$$(\mathbf{z}_{\text{key}}, \mathbf{z}_{\text{val}}) = \text{READ}(\mathbf{q}, \mathcal{M}_{\text{ext}}), \quad (5.2)$$

$$\mathcal{M}_{\text{abs}} \leftarrow \text{WRITE}(\mathbf{q}, (\mathbf{z}_{\text{key}}, \mathbf{z}_{\text{val}}), \mathcal{M}_{\text{abs}}), \quad (5.3)$$

$$(\mathbf{u}_{\text{key}}, \mathbf{u}_{\text{val}}) = \text{READ}(\mathbf{q}, \mathcal{M}_{\text{abs}}), \quad (5.4)$$

$$\hat{\mathbf{y}} = \text{CLS}([\mathbf{u}_{\text{key}}, \mathbf{u}_{\text{val}}]). \quad (5.5)$$

We elaborate on each of the operation in the procedure, and all of the following operations are parameterized by neural networks:

1. `Embed` is a transformation from the raw inputs to their feature representation.

We denote the network to extract image feature as  $\Phi_{\text{img}}$  and the one to extract vector representation for label as  $\Phi_{\text{label}}$ . Given an image  $I$  from a novel category, and a group of web images with labels, denoted as  $\mathcal{I}_{\text{web}}$  and  $\mathcal{L}_{\text{web}}$ , where

$\mathcal{I}$  is the image set and  $\mathcal{L}$  is the label set, the input image  $I$  is sampled from unseen categories, and the embedded feature for the query image is referred to as query  $\mathbf{q}$  following the notation in Memory Networks. The web images are embedded in the external memory  $\mathcal{M}_{\text{ext}}$  through the same embedding networks  $\Phi_{\text{img}}$  and  $\Phi_{\text{label}}$ ;

2. **READ** takes the query  $\mathbf{q}$  as input and conducts content-based addressing on the external memory  $\mathcal{M}_{\text{ext}}$ , to find related information according to a similarity metric with  $\mathbf{q}$ . The external memory is also called the support set in Memory Networks. The output of **READ** is a pair of vectors in key-value form, i.e.,  $(\mathbf{z}_{\text{key}}, \mathbf{z}_{\text{val}})$ , as shown in Eqn. (5.2);
3. **WRITE** takes a query  $\mathbf{q}$  and key-value pair  $(\mathbf{z}_{\text{key}}, \mathbf{z}_{\text{val}})$  as inputs to conduct a write operation. The content-based addressing is based on matching input with  $\mathcal{M}_{\text{abs}}$ , and then updating the content of the corresponding abstraction memory slots as in Eqn. (5.3);
4. **READ** from abstraction memory (Eqn. (5.4)) is for the classification stage. Take the input query  $\mathbf{q}$  to match the abstraction memory  $\mathcal{M}_{\text{abs}}$ . The obtained pairs of vectors (i.e.,  $(\mathbf{u}_{\text{key}}, \mathbf{u}_{\text{val}})$ ) are concatenated to be fed into the classification network;
5. **CLS** operation takes the readout key-value  $(\mathbf{z}_{\text{key}}, \mathbf{z}_{\text{val}})$  and concatenates them into one vector  $\mathbf{z}_{\text{cls}} = [\mathbf{z}_{\text{key}}, \mathbf{z}_{\text{val}}]$ . Then  $\mathbf{z}_{\text{cls}}$  goes through a Fully-Connected (FC) layer where:  $\text{FC}(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + \mathbf{b}$ , and a **Softmax** layer.

Section 5.2.3 shows an LSTM variant of the **CLS** operation.

### 5.2.3 Model Components

#### *Long Short-Term Memory*

In our model, Long Short-Term Memory (LSTM) [37] plays an important role in the READ, WRITE and CLS procedures and serves as the *controller* of the memory addressing. LSTM is a special form of Recurrent Neural Networks (RNNs). LSTM addresses the vanishing gradient problem [9] of RNNs by introducing an *internal* memory cell to encode information from the previous steps. LSTM has resurged due to the success of sequence to sequence modeling [98] on machine translation [7], image captioning [116, 45, 127], video classification [136], video captioning [112, 74]. Following the notations of Zaremba et al. [137] and Xu et al. [127] and assuming  $\mathbf{x}_t \in \mathbb{R}^D$ ,  $\mathbf{T}_{D+d,4d} : \mathbb{R}^{D+d} \rightarrow \mathbb{R}^{4d}$  denotes an affine transformation from  $\mathbb{R}^{D+d}$  to  $\mathbb{R}^{4d}$ , LSTM is implemented as:

$$\begin{pmatrix} \mathbf{i}_t \\ \mathbf{f}_t \\ \mathbf{o}_t \\ \mathbf{g}_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \mathbf{T}_{D+d,4d} \begin{pmatrix} \mathbf{x}_t \\ \mathbf{h}_{t-1} \end{pmatrix} \quad (5.6)$$

$$\mathbf{c}_t = \mathbf{f} \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \quad (5.7)$$

$$\mathbf{h}_t = \mathbf{o} \odot \tanh(\mathbf{c}_t), \quad (5.8)$$

where  $\mathbf{i}_t, \mathbf{f}_t, \mathbf{c}_t, \mathbf{o}_t$  are the input, forget, memory, output gates respectively,  $\sigma$  and  $\tanh$  are element-wise activation functions,  $\mathbf{x}_t$  is the input to the LSTM in the  $t$ -th step and  $\mathbf{h}_t$  is the hidden state of the LSTM in the  $t$ -th step.

For simplicity of notation, we denote one computational step of the LSTM recurrence as a function LSTM, defined as:

$$\mathbf{h}_t = \text{LSTM}(\mathbf{x}_t, \mathbf{h}_{t-1}). \quad (5.9)$$

### *Reading from the Memory*

In this section, we describe the mechanism for reading information from the memory. Given an external memory with buffer size  $N_1$ ,  $\mathcal{M} = \{(\mathbf{m}_{\text{key}}^1, \mathbf{m}_{\text{val}}^1), (\mathbf{m}_{\text{key}}^2, \mathbf{m}_{\text{val}}^2), \dots, (\mathbf{m}_{\text{key}}^{N_1}, \mathbf{m}_{\text{val}}^{N_1})\}$ , where each memory slot  $\mathbf{m}^i$  is encoded as a key-value structure, i.e.,  $(\mathbf{m}_{\text{key}}^i, \mathbf{m}_{\text{val}}^i)$ , or equivalently  $\mathbf{m}_{\text{key}}^i : \mathbf{m}_{\text{val}}^i$ .  $\mathbf{m}_{\text{key}}^i \in \mathbb{R}^{d_1}$ ,  $\mathbf{m}_{\text{val}}^i \in \mathbb{R}^{d_2}$ , where  $d_1$  is the dimension of the image embedding (i.e., the **key** part) in the memory slot, and  $d_2$  denotes the dimension of the label embedding (i.e., the **val** part) in the memory slot. We use the tuple notation  $(\mathbf{m}_{\text{key}}^i, \mathbf{m}_{\text{val}}^i)$  in the rest. We apply the reading mechanisms from the set-to-set framework [114] on the memory bank. For each time step  $t$ , we have:

$$\mathbf{q}_t = \text{LSTM}(\mathbf{0}, \mathbf{q}_{t-1}^*) \quad (5.10)$$

$$e_{i,t} = \mathbf{q}_t^\top \mathbf{m}_{\text{key}}^i \quad (5.11)$$

$$a_{i,t} = \text{Softmax}(e_{i,t}) \quad (5.12)$$

$$\mathbf{z}_{\text{key}}^t = \sum_i a_{i,t} \mathbf{m}_{\text{key}}^i \quad (5.13)$$

$$\mathbf{z}_{\text{val}}^t = \sum_i a_{i,t} \mathbf{m}_{\text{val}}^i \quad (5.14)$$

$$\mathbf{q}_t^* = [\mathbf{q}_t, \mathbf{z}_{\text{key}}^t]. \quad (5.15)$$

$(\mathbf{m}_{\text{key}}^i, \mathbf{m}_{\text{val}}^i)$ ,  $i = 1, 2, \dots, N_1$ , are all the memory slots stored in  $\mathcal{M}$ . When the query  $\mathbf{q}_t$  comes, it conducts dot product with all of the **key** parts of the memory slot  $\mathbf{m}_{\text{key}}^i$  (Eqn. (5.11)), to obtain the similarity metric  $e_{i,t}$  between query image  $\mathbf{q}_t$  and the image in the memory slot  $\mathbf{m}_{\text{key}}^i$ . The **Softmax** operation of Eqn. (5.12) generates an attention weight  $a_{i,t}$  over the whole memory  $\mathcal{M}$ . Then, Eqn. (5.13) and Eqn. (5.14) utilize the learned attention weight  $a_{i,t}$  to read the **key** part and the **value** part, i.e., label embedding, from the external memory. The readout operation blends all of the key/value vectors  $\mathbf{m}_{\text{key}}^i / \mathbf{m}_{\text{val}}^i$  with the attention weight  $a_{i,t}$  to obtain the readout vectors  $\mathbf{z}_{\text{key}}^t$  and  $\mathbf{z}_{\text{val}}^t$ . Lastly,  $\mathbf{z}_{\text{key}}^t$  is concatenated with query  $\mathbf{q}_t$ , producing  $\mathbf{q}_t^*$  to

be fed into the next step as the input of LSTM (Eqn. (5.10)). The above reading procedure loops over the memory for  $T$  timesteps, obtaining  $T$  readout pairs of vectors, i.e.,  $\{(\mathbf{z}_{\text{key}}^1, \mathbf{z}_{\text{val}}^1), (\mathbf{z}_{\text{key}}^2, \mathbf{z}_{\text{val}}^2), \dots, (\mathbf{z}_{\text{key}}^T, \mathbf{z}_{\text{val}}^T)\}$ . The LSTM controller takes no input but computes the recurrent state to control the reading operation. For more details, please refer to the vector version (the memory slot is in the form of a vector instead of a key-value) of this reading mechanism [114].

After  $T$ -step READ operations over memory  $\mathcal{M}$  (which could be either  $\mathcal{M}_{\text{ext}}$  or  $\mathcal{M}_{\text{abs}}$ ), we can obtain:

$$\mathcal{Z} = \{(\mathbf{z}_{\text{key}}^1, \mathbf{z}_{\text{val}}^1), (\mathbf{z}_{\text{key}}^2, \mathbf{z}_{\text{val}}^2), \dots, (\mathbf{z}_{\text{key}}^T, \mathbf{z}_{\text{val}}^T)\}. \quad (5.16)$$

### ***Abstraction Memory***

We propose to utilize a novel memory component, namely *abstraction memory*, in our implementation of Memory Networks. The abstraction memory has the following properties:

1. It learns task-specific representation for the few-shot object recognition task;
2. It attempts to tackle the problem of efficiency of content-based addressing over a large external memory pool.

Abstraction memory is a *writable* memory bank  $\mathcal{M}_{\text{abs}}$ , with buffer size  $N_2$ . It satisfies  $N_2 < N_1$ , where  $N_1$  is the buffer size of the external memory bank  $\mathcal{M}_{\text{ext}}$ . We denote  $\mathcal{M}_{\text{abs}} = \{(\tilde{\mathbf{m}}_{\text{key}}^1, \tilde{\mathbf{m}}_{\text{val}}^1), (\tilde{\mathbf{m}}_{\text{key}}^2, \tilde{\mathbf{m}}_{\text{val}}^2), \dots, (\tilde{\mathbf{m}}_{\text{key}}^{N_2}, \tilde{\mathbf{m}}_{\text{val}}^{N_2})\}$ , where  $\tilde{\mathbf{m}}_{\text{key}}^i \in \mathbb{R}^{\tilde{d}_1}$ ,  $\tilde{\mathbf{m}}_{\text{val}}^i \in \mathbb{R}^{\tilde{d}_2}$ ,  $\tilde{d}_1$  is the dimension of the **key** vector stored in the memory slot, and  $\tilde{d}_2$  is the dimension of the **value** part stored in the memory slot.

**Writing.** Unlike the *external* memory bank, the *abstraction* memory bank is “writable”, which means the neural networks can learn to update the memory slots in the storage by remembering and abstracting what is important for specific tasks.

The memory update is according to an embedding (i.e., through an FC layer) of the readout ( $\mathbf{z}_{\text{key}}, \mathbf{z}_{\text{val}}$ ) from the larger external memory bank  $\mathcal{M}_{\text{ext}}$ .

Following the writing operation proposed in Neural Turing Machines (NTMs) [33], we conduct the differentiable WRITE operation on the abstraction memory bank  $\mathcal{M}_{\text{abs}}$ . The LSTM controller produces *erase* vectors  $\mathbf{e}_{\text{key}} \in \mathbb{R}^{\tilde{d}_1}$ ,  $\mathbf{e}_{\text{val}} \in \mathbb{R}^{\tilde{d}_2}$ , and *add* vectors  $\mathbf{a}_{\text{key}} \in \mathbb{R}^{\tilde{d}_1}$ ,  $\mathbf{a}_{\text{val}} \in \mathbb{R}^{\tilde{d}_2}$ . Note that each element of the erase vector satisfies  $0 < \mathbf{e}_{\text{key}}^i < 1$  and  $0 < \mathbf{e}_{\text{val}}^i < 1$ , which can be implemented by passing through a Sigmoid function  $\sigma(x)$ .

For each memory slot  $\tilde{\mathbf{m}}^i$ , the WRITE operation conducts the following updates in the abstraction memory bank  $\mathcal{M}_{\text{abs}}$ . For each timestep  $t$ , we have

$$\tilde{\mathbf{m}}_{\text{key}}^i \leftarrow \tilde{\mathbf{m}}_{\text{key}}^i (\mathbf{1} - w_{i,t} \mathbf{e}_{\text{key}}) + w_{i,t} \mathbf{a}_{\text{key}}, \quad (5.17)$$

$$\tilde{\mathbf{m}}_{\text{val}}^i \leftarrow \tilde{\mathbf{m}}_{\text{val}}^i (\mathbf{1} - w_{i,t} \mathbf{e}_{\text{val}}) + w_{i,t} \mathbf{a}_{\text{val}}. \quad (5.18)$$

The vector  $\mathbf{w}_t$  is used for addressing mechanisms in the WRITE operation [33]. However, in contrast to NTMs, we do not utilize location-based addressing but only content-based addressing over the abstraction memory  $\mathcal{M}_{\text{abs}}$ . The vector  $\mathbf{w}_t$  can be calculated as in Eqn. (5.11) and Eqn. (5.12), by replacing  $\mathbf{m}_{\text{key}}$  of  $\mathcal{M}_{\text{ext}}$  into  $\tilde{\mathbf{m}}_{\text{key}}$  of  $\mathcal{M}_{\text{abs}}$ .

### ***Label Prediction***

When it reaches the prediction stage, our model reads  $(\mathbf{u}_{\text{key}}, \mathbf{u}_{\text{val}})$  from the abstraction memory  $\mathcal{M}_{\text{abs}}$ , as shown in Eqn. (5.4). The reading mechanism has been illustrated in Section 5.2.3. Reading from the memory is a recurrent process, with  $T$  timesteps, and we can fetch readouts  $\mathcal{U} = \{[\mathbf{u}_{\text{key}}^1, \mathbf{u}_{\text{val}}^1], [\mathbf{u}_{\text{key}}^2, \mathbf{u}_{\text{val}}^2], \dots, [\mathbf{u}_{\text{key}}^T, \mathbf{u}_{\text{val}}^T]\}$  to obtain enough information for few-shot classification, where  $[\mathbf{u}_{\text{key}}^i, \mathbf{u}_{\text{val}}^i]$  denotes the concatenation of two vectors into one. We then run an LSTM on top of the sequence  $\mathcal{U}$ , obtain the final state output  $\mathbf{h}_T$  from the LSTM, and then feed  $\mathbf{h}_T$  into

an FC layer and a `Softmax` layer to output the prediction  $\hat{\mathbf{y}}$ .

In this way, our model fully utilizes the readout vectors with both visual information and label embedding information to conduct classification. These readout vectors are from abstraction memory, which learns to adapt in specific tasks, e.g., few-shot object recognition.

#### 5.2.4 Training

We apply a standard cross entropy loss between the prediction  $\hat{\mathbf{y}}$  and the groundtruth  $\mathbf{y}$ , where  $\mathbf{y}$  is the one-hot representation of the groundtruth label.

All of the operations and components in our model are fully differentiable, which means we can train our model with stochastic gradient descent (SGD) in an end-to-end way.

#### 5.2.5 Inference

In the inference (testing) stage, we do not make the external memory  $\mathcal{M}_{\text{ext}}$  available, since the abstraction memory  $\mathcal{M}_{\text{abs}}$  has stored all of the required information in the form of key-value in the memory slots. Thus, on the inference stage, we only run the prediction process (Section 5.2.3) on the fetched vectors from  $\mathcal{M}_{\text{abs}}$ . The predicted label is obtained by an `argmax` operation over the softmax probability output  $\hat{\mathbf{y}}$ .

### 5.3 Experiments

We evaluate our proposed model using two different external image sources, i.e., ImageNet [87] dataset and OpenImages [53] dataset. In this section, we describe the specific model configurations used in the experiments, and show the results of the few-shot recognition model trained from clean human-labeled annotations and machine-labeled annotations. Our model is implemented using TensorFlow [3].

### 5.3.1 Preprocessing

We use features from top convolutional layers as image embeddings. In all our experiments, we use the last layer activations before the final classification from the ResNet-200 [36] model pretrained on ImageNet [87]. This single model achieved top-5 error of 5.79% on the ILSVRC 2012 validation set. Following standard image preprocessing practice, images are first resized to 256 on the short side and the central  $224 \times 224$  subregion is cropped; we thus obtain an image embedding with the feature dimension of 2,048. We apply the word embedding from the state-of-the-art language modeling model [43] in our label to word embedding mapping. We follow the instructions provided by the authors to extract embeddings for each word in the vocabulary, and embeddings are averaged if there are multiple words for one category. The embedding length is 1,024, thus we have the embedding matrix of  $|V|$  by 1,024, where  $|V|$  is the size of the vocabulary  $V$ . The ResNet for visual feature extraction and the label embedding matrix will *not* be updated during training.

### 5.3.2 Model Specifications

For all the LSTM models, we use one-layer LSTM with hidden unit size of 1,024. In particular, we utilize Layer Normalization [6] for the gates and states in the cell, which we found was crucial to train our model. Layer Normalization helps to stabilize the learning procedure in RNNs, without which we could not train the network successfully. Dropout is used in the input and output of LSTM and we set the Dropout probability to 0.5. The default model parameters are described as follows. We use  $N_1 = 1,000$  memory slots for the external memory bank and  $N_2 = 500$  memory slots for the abstraction memory. Both **key** and **value** vectors stored in the abstraction memory have the dimensionality of 512. The controller iterates  $T = 5$  times when abstracting information from the external memory banks. We use the default model parameters in all the experiments unless otherwise stated.



Our model is trained with an ADAM optimizer [48] with learning rate of  $1 \times 10^{-4}$  and clip the norm of the global gradients at 10 to avoid the gradient exploding problem [98]. Weights in the neural network are initialized with Glorot uniform initialization [31] and weight decay of  $1 \times 10^{-4}$  is applied for regularization.

### 5.3.3 Datasets

**russakovsky2014imagenet:** ImageNet is a widely used image classification benchmark. There are two sets in the ImageNet dataset. One part is used in the ILSVRC classification competitions, namely ILSVRC 2012 CLS. This part contains exactly 1,000 classes with about 1,200 images per class, with well-verified human-labeled annotations. The other set of ImageNet is the whole set, which consists of about 21,000 categories.

**OpenImages.** The recently released OpenImages dataset [53] consisting of web images with machine-labeled annotations. We utilize the validation set with 167,057 images to conduct our experiment, since both machine-labeled annotations and human-labeled annotations are provided only in the validation set. There are 7,844 distinct labels in OpenImages, whose label vocabulary and diversity is much richer than the ILSVRC 2012 CLS dataset. Since this dataset is relatively new, we provide example images in Figure 5.3. We can see that the OpenImages dataset has a wider vocabulary than the ImageNet ILSVRC 2012 dataset, which is beneficial for generalization to novel categories.

### 5.3.4 Few-shot Learning with Human-labeled annotations

We first validate our model on the task of few-shot classification using human-labeled clean data.

For few-shot image classification, the training set has only a small number of examples, and the basic task can be denoted as  $N$ -way  $k$ -shot classification (following

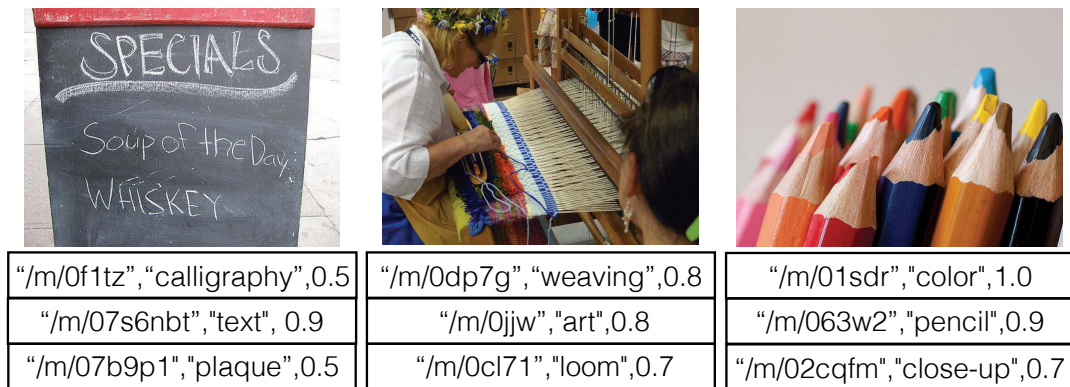


Figure 5.3 : Sample images from the OpenImages dataset. Annotations on the images are shown in the bottom. The annotations listed are “label id”, “label name”, “confidence” tuples.

the notation of Matching Networks [115]), in which  $N$  class images need to be classified and each class is provided with  $k$  labeled examples ( $k$  is usually less than 10).

**Dataset.** We now construct our dataset for few-shot learning. We select 100 classes for learning by randomly choosing 100 categories from the entire 21,000 categories in the ImageNet dataset, excluding the 1,000 categories in the ILSVRC 2012 CLS vocabulary. For testing, there are 200 images per category and the training set has  $k$  examples per category. We use settings of  $k = 1$ ,  $k = 5$ ,  $k = 10$ , i.e., there are 1 example, 5 examples and 10 examples in the training set.

### *Comparison with other methods*

In this experiment, we use image-label pairs from the ILSVRC 2012 CLS dataset as external memory. We use all 1,000 categories for learning. We conduct experiments on 1-shot, 5-shot, 10-shot tasks and compare our results with several algorithms. The results are shown in Table 5.1.

**$k$ -NN and Exemplar-SVMs.**  $k$ -Nearest Neighbors ( $k$ -NN) is a simple but effective classifier when very few training examples are provided. We utilize ResNet-200 features and consider two distance metrics, i.e.,  $l_1$  and  $l_2$ , for pairwise distance calculation. And we set  $k = 5$ . Exemplar-SVMs (E-SVM) [66] train an SVM for each positive example and ensemble them to obtain the final score. The method was widely used in object detection in the pre-ConvNet era. We use the same ResNet-200 features and set  $C = 0.1$ . The results show that our method outperforms  $k$ -NN for both  $l_1$  and  $l_2$  distance with a large margin and it also outperforms E-SVMs. Note that on 5-shot and 10-shot tasks, our model achieves better performance than the E-SVMs with larger margin. The results show that our model takes advantage of the large number of image-label pairs in the external memory by learning relationships between the examples and the external data.

**KV-MemNNs.** By utilizing the interpretation of image embedding as **key** and label embedding as **value** as in our model, KV-MemNNs can also be trained to conduct few-shot learning. However, due to the design of KV-MemNNs, few-shot prediction has to rely on the external memory, while the image classification datasets used in our work are too large to be stored in. This property means that KV-MemNNs conduct non-deterministic classification prediction, which is not desirable. It is unrealistic to search over all image-pairs in the external memory during each training iteration. In the testing, it is also time-consuming to traverse the whole external memory. As a workaround, we randomly sample 1,000 pairs from the external memory for matching during both training and testing. We report the mean classification results and the standard deviation in 20 runs. The result shows that our abstraction memory extracts valuable information from the large external memory and is much more compact than the original memory banks.

**Matching Networks.** We also compare our method with the recently proposed Matching Networks [115]. Matching Networks use two embedding functions that

Methods	1-shot	5-shot	10-shot
$k$ -NN ( $l_1$ )	38.8	57.0	62.9
$k$ -NN ( $l_2$ )	38.6	56.4	62.1
E-SVM	45.1	62.3	68.0
KV-MemNNs	43.2 ( $\pm 0.4$ )	66.6 ( $\pm 0.2$ )	72.8 ( $\pm 0.2$ )
Ours	<b>45.8</b>	<b>68.0</b>	<b>73.5</b>

Table 5.1 : Comparison between our model and other methods. Results are reported on our 100-way testing set.

consider set context. However, as LSTM is used for the embedding, the size of the support set is limited. In [115], the number of categories is usually set to 5 for ImageNet experiments (5-way). For fair comparison, we conduct our experiment on the 5-way 1-shot task. We randomly choose 5 categories from the previously used 100-category set. The testing set has the same number of instances per category. The result is shown in Table 5.2, which demonstrates that our method outperforms the Matching Network. Our model builds an explicit connection between the few training examples and the external memory, which benefits greatly from a large vocabulary.

We visualize the query results between the external memory and the query in Figure 5.4.

### 5.3.5 Few-shot Learning with Machine-labeled Annotations

In this experiment, we replace the external memory source with the OpenImages dataset. The machine-labeled images are much easier to obtain but are noisier. We train our model to learn from such noisy web images.

Methods	5-way 1-shot classification
Matching Networks	90.1
Ours	<b>93.9</b>

Table 5.2 : Comparison between our model and Matching Networks on the 5-way 1-shot task.

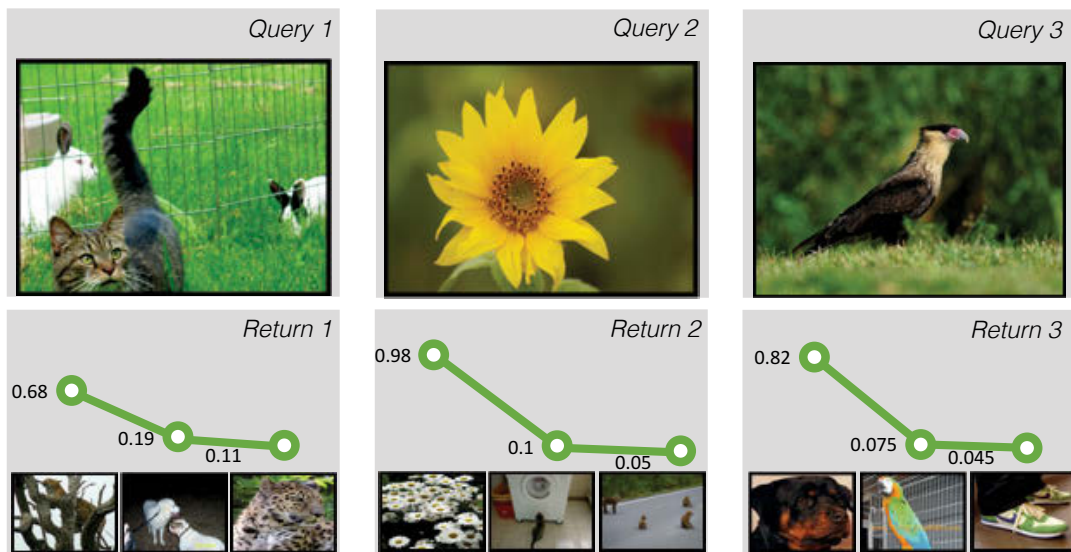


Figure 5.4 : We show the query results returns from the external memory. The scores are the softmax probabilities. Only top-3 results are shown.

We construct the external memory using the OpenImages dataset. We use four different settings, which are: 1,000 vocabulary with human-labeled images, 1,000 vocabulary with machine-labeled images, 6,000 vocabulary with human-labeled images, and 6,000 vocabulary with machine-labeled images. Note that although the OpenImages dataset is machine-labeled, the validation set in the original dataset is also validated by human raters. The results are shown in Table 5.3, which demonstrates that machine-labeled external memory can serve as a good source for few-shot learning, which is less accurate than human-labeled external memory by only about

Methods	1,000	6,000
Machine-labeled	66.6	67.4
Human-labeled	67.7	68.2

Table 5.3 : Results on the OpenImages dataset. The results are reported on the 100-way 5-shot task.

$N_1 : N_2$	Accuracy (%)
500 : 500	67.6
1000 : 250	67.9
1000 : 500	68.0
1000 : 1000	67.7
2000 : 500	68.1

Table 5.4 : Comparisons among the numbers of memory slots.

1%.

As the vocabulary size grows, we observe that performance improves. This shows that given a large vocabulary, our model is able to reason among the external memory in a more effective way. Larger vocabulary will be explored in the future.

### 5.3.6 Hyperparameter Study

We conduct the hyperparameter study on the memory slots numbers, i.e.,  $N_1$  for the external memory and  $N_2$  for the abstraction memory. Table 5.4 shows the comparisons among different combinations of memory slots in 5-shot recognition on ImageNet dataset, which demonstrates that our proposed model is robust to the change of memory slots.

## 5.4 Conclusion

We propose a novel Memory Networks architecture specifically tailored to tackle the few-shot learning problem on object recognition. By incorporating a novel memory component into the Key-Value Memory Networks, we enable rapid learning from seeing only a handful of positive examples by abstracting and remembering the presented external memory. We utilize LSTM controllers for reading and writing operations into the memory. We demonstrate that our proposed model achieves better performance than other state-of-the-art methods. Furthermore, we obtain similar performance by utilizing machine-labeled annotations compared to human-labeled annotations.

## Chapter 6

# Compound Memory Networks for Few-shot Video Classification

### 6.1 Introduction

In this chapter, I propose a new memory network structure for few-shot video classification by making the following contributions. First, I propose a compound memory network (CMN) structure under the key-value memory network paradigm, in which each key memory involves multiple constituent keys. These constituent keys work collaboratively for training, which enables the CMN to obtain an optimal video representation in a larger space. Second, I introduce a multi-saliency embedding algorithm which encodes a variable-length video sequence into a fixed-size matrix representation by discovering multiple saliencies of interest. For example, given a video of car auction, some people are interested in the car, while others are interested in the auction activities. Third, I design an abstract memory on top of the constituent keys. The abstract memory and constituent keys form a layered structure, which makes the CMN more efficient and capable of being scaled, while also retaining the representation capability of the multiple keys. I compare CMN with several state-of-the-art baselines on a new few-shot video classification dataset and show the effectiveness of the approach.

Deep learning models have been successfully applied to many tasks, e.g., image classification [54, 92, 101, 35], image detection [84], video classification [46, 91] and

---

This chapter is based on joint work with Yi Yang (Zhu and Yang, 2018 [129]), presented primarily as it appears in the ECCV 2018 proceedings.



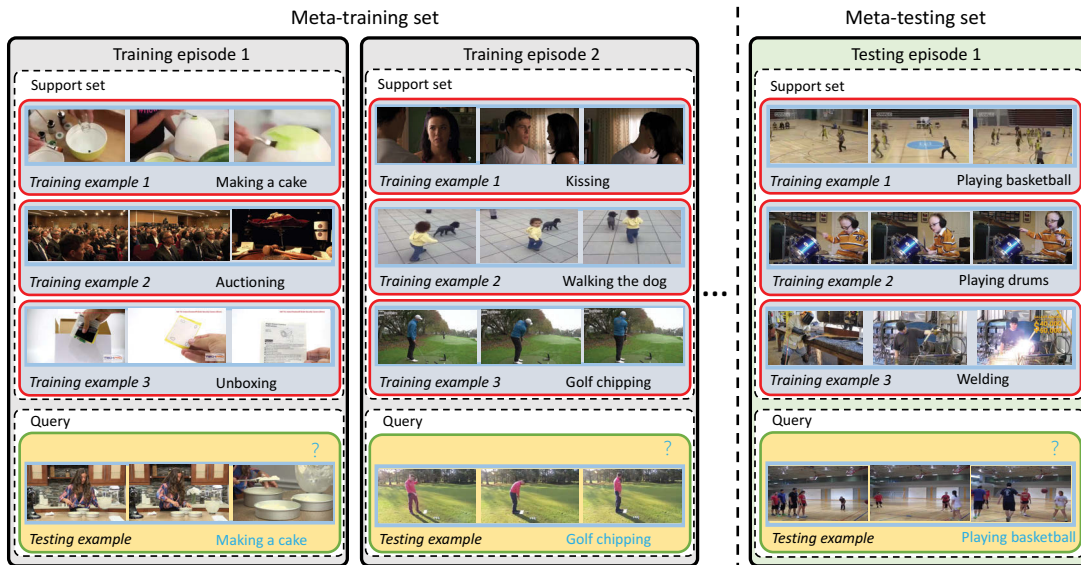


Figure 6.1 : The setting of the few-shot video classification. There are two non-overlapping datasets in this figure, i.e., meta-training and meta-testing. The meta-training set is for meta-learning and the meta-testing set is for evaluating the generalization performance on novel categories. The network is trained in an episodic way and each episode has a support set and a query example.

machine translation [98, 125]. We aim to enable a system to learn how to classify video data into a new category by exploiting a meta-training set. As shown in Figure 6.1, the meta-training set consists of a number of episodes which mimic the few-shot learning task. In this example, there is only one positive exemplar per class in each episode, indicated by a red rectangle. There is no overlapping category between the training phase and testing phase. During the training phase, the system learns an optimal mechanism that best recognizes queries in all training episodes. When testing, the system directly adopts the learned optimal mechanism to classify each query in testing episodes.

We focus on few-shot video representation learning. Videos have more complex structures than images, involving temporal information and more noise, e.g., camera

motion, object scales, viewpoints. It is a more challenging task than few-shot image classification. Many videos usually contain hundreds of frames containing various scene dynamics. It could be difficult to understand the concept in a video when only few examples are provided.

We thus propose a compound memory network (CMN) structure for few-shot video classification. Our CMN structure is designed on top of the key-value memory networks [123] for the following two reasons. First, new information can be readily written to memory, which provides our model with better ‘memorization’ capability. In other words, MANNs are able to store and memorize an example long-term, even though the example has been seen only once. Second, information stored in the memory module can be memorized for a longer period and can be easily accessed. During training, information in each training episode is gradually accumulated into CMN, which is then used as the learned few-shot video classification mechanism for testing. It is worthwhile highlighting the following aspects of our CMN model.

First, we propose a new notion of compound memory networks with a much stronger representation capability by extending the key memory elements from a 1D vector to a 2D matrix. Standard key-value memory networks use a single vector as the key in each memory slot [70]. Videos are more complex in structure than images and have richer semantic information. Accordingly, we propose to use multiple vectors to enhance the video representation, with each vector being a constituent key. The constituent keys are stacked to a matrix to generate the video representation in CMN. These stacked constituent keys work collaboratively in the training phase, providing a larger search space from which to obtain an optimal video representation.

Second, we introduce a series of hidden saliency descriptors as constituent keys in the memory slots of CMN. In many cases, user may be interested in different salient parts of a video. For example, given a video of a birthday party, some

users may be more interested in the dancing scene, while others focus on the food and drinks. We propose a multi-saliency embedding algorithm which automatically detects multiple saliencies of interest in any given video. We extend the self-attention mechanism [63, 109] by integrating a newly designed learnable variable to adaptively detect hidden salient genres within a video. The multi-saliency embedding algorithm learns a hidden saliency descriptor for each genre, which is then stacked as a video representation in CMN.

Third, we design a layered memory structure, which vastly improves efficiency while retaining the strong representation capability of CMN. The first layer stores the stacked constituent keys. We design an abstract memory on top of the first layer, which is equipped with reading and writing operations for retrieving and updating the constituent keys. The abstract memory compresses the stacked constituent keys into a vector and vastly improves training and testing efficiency. At the same time, the communication between the two layers ensures that abstract memory is able to retain the information from all constituent keys.

## 6.2 Few-shot Video Classification Setup

In the few-shot video classification setting, we aim to train a network that can generalize to new episodes over novel classes. Each episode in a mini-batch mimics a few-shot classification task, which consists of a support set and a query set. The support set contains training videos and labels, while the query set is for evaluating the generalization performance. In an  $n$ -way,  $k$ -shot problem, the goal of each episode is to classify query videos into  $n$  classes with only a small number of support examples per class ( $k$ ). Videos and labels in an episode are sampled from a meta set. The meta set has  $N$  classes ( $N > n$ ), and each class has  $K$  examples ( $K > k$ ). In our setup, there are three meta sets, i.e., meta-training set, meta-validation set and meta-testing set with  $N_{training}$ ,  $N_{validation}$  and  $N_{testing}$  classes, respectively. The meta-

training set is for meta-learning which minimizes the loss over training episodes. The meta-validation set is for hyper-parameter tuning. We report the accuracy on the meta-testing set. The three meta sets do not have overlapping categories. Following [89, 115], we construct an episode by randomly choosing  $n$  classes from  $N$  categories in the meta set. For each class,  $k$  videos are selected from  $K$  examples. The label indices for  $n$  classes are randomly shuffled across different episodes, which prevents the model from memorizing the association between the input and the label.

In a standard video classification problem, there is a single training dataset  $D_{single}$  with fixed categories. Given an input/output pair  $(\mathbf{x}, y)$  sampled from  $D_{single}$ , the goal is to minimize the estimated loss over all training examples, i.e.,

$$\min_{\theta} \mathbb{E}_{(\mathbf{x}, y) \sim D_{single}} [\mathcal{L}(\mathbf{x}, y)], \quad (6.1)$$

where  $\theta$  represents the trainable parameters in a model.

In the few-shot video classification problem, training is conducted over a number of different episodes. An episode  $T_i$  sampled from meta-set  $\mathcal{T}$  involves an episode length  $l$ , inputs  $\mathbf{x}_t$ , outputs  $y_t$  and a loss function  $\mathcal{L}(x_t, y_t)$ , where  $t = \{1, 2, \dots, l\}$ . During meta-training, the network is trained to predict the label of  $\mathbf{x}_t$  at each step given previous input pairs  $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_{t-1}, y_{t-1})\}$ . The objective is to minimize the expected loss over mini-batches of episodes, i.e.,  $\min_{\theta} \mathbb{E}_{T_i \sim \mathcal{T}} [\sum_{t=1}^l \mathcal{L}_i(\mathbf{x}_t, y_t)]$ .

### 6.3 Compound Memory Network

We first illustrate the multi-saliency embedding function that learns a fixed-size matrix representation for a variable-length video sequence. We then show the detailed structure of our Compound Memory Network, and introduce the novel components, i.e., the constituent keys, abstract memory, together with the accessing

and updating operations.

### 6.3.1 Multi-saliency Embedding Function

Videos have variable lengths and should be encoded into a fixed-size matrix before being stored in memory. Given a query video  $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_{m'}\}$ , where  $m'$  is the number of video frames and  $\mathbf{p}_i$  is a frame-level representation extracted by a ConvNet, video  $\mathbf{P}$  should be aggregated into a fixed-size matrix  $\mathbf{Q}$ . The representation  $\mathbf{Q}$  consists of  $m$  stacked hidden descriptors  $\{\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_m\}$ , and the size of each hidden descriptor is `hidden-size`. Note that the number of video frames  $m'$  varies across different videos, but  $m$  is a fixed number.

We design the multi-saliency embedding function (MEF) by introducing a hidden variable  $\mathbf{H}$  with  $m$  components  $\{\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_m\}$ . Each component  $\mathbf{h}_j$  is used to detect one saliency in a video. For each input  $\mathbf{p}_i$ , a soft weight  $\mathbf{a}_{ij}$  over  $\mathbf{h}_j$  will be calculated which measures the relevance between the input and the component. The hidden descriptor  $\mathbf{q}_j$  will be the weighted sum over the residual between  $\mathbf{P}$  and  $\mathbf{h}_j$ . Thus, the MEF function can be formulated by

$$\mathbf{a}_i = \text{softmax}\left(\frac{\mathbf{p}_i \mathbf{H}^T}{\sqrt{d_{\text{hidden-size}}}}\right), \quad \mathbf{q}_j = \sum_{i=1}^m \mathbf{a}_{ij} (\mathbf{p}_i - \mathbf{h}_j), \quad (6.2)$$

where `softmax` is defined as,  $\text{softmax}(\mathbf{e}) = \frac{\exp(e_i)}{\sum_i \exp(e_i)}$ .

To calculate the relevance score between  $\mathbf{p}_i$  and  $\mathbf{h}_j$ , we simply use dot-product but include a scaled factor  $\frac{1}{\sqrt{d_{\text{hidden-size}}}}$  [109] followed by a `softmax` function. The original sequence  $\mathbf{P}$  is mapped to our multi-saliency descriptor  $\mathbf{Q}$ , *i.e.*,  $\mathbf{Q} = \text{MEF}(\mathbf{P}, \mathbf{H})$ .  $\mathbf{Q}$  is then flattened and normalized to a vector, which will be discussed in Section 6.4 (Figure 6.2). [63, 109] introduced multi-hops attention to calculate multiple weighted sums over the inputs. In contrast, we introduce a hidden variable  $\mathbf{H}$  to explicitly model the relation between the input and each hidden vector, which learns

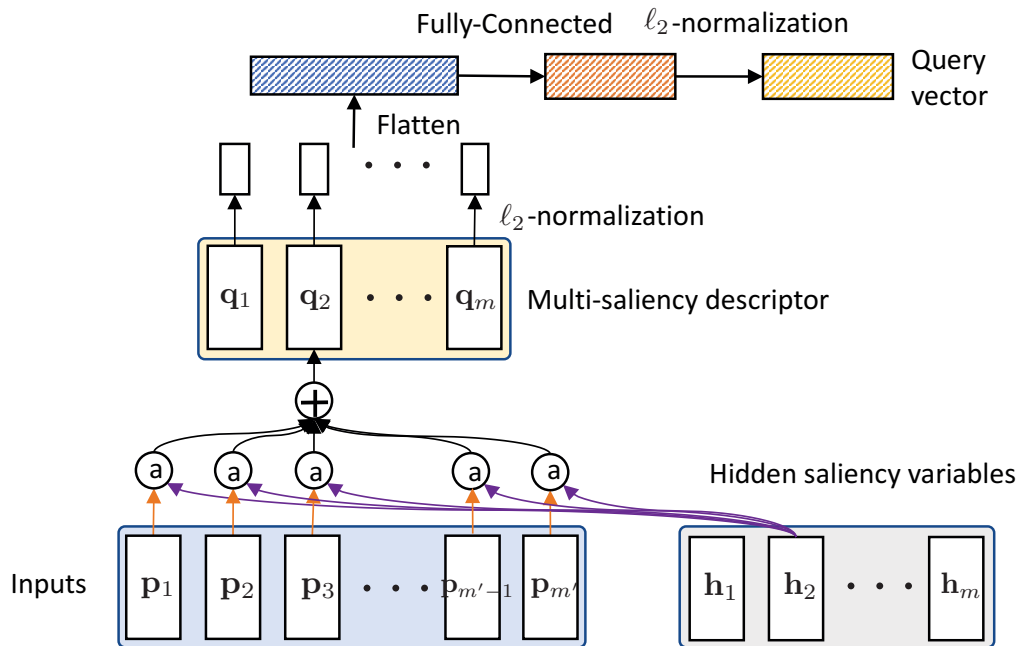


Figure 6.2 : Illustration of the input embedding model. The embedding function generates the multi-saliency descriptor  $\mathbf{Q}$ , which is flattened and normalized to a query vector.

multiple descriptors for different salient parts in a video.

### 6.3.2 Compound Memory Structure

Our Compound Memory Network is a variant of the Key-Value Memory Networks, which has the key memory ( $\mathcal{K}$ ) and the value memory ( $\mathcal{V}$ ). Visual information is stored in the key part, while the label information is stored in the value part. Our key memory is a layered structure in which the first layer stores the constituent keys ( $\mathcal{C}$ ) and the second layer is the abstract memory ( $\mathcal{A}$ ). We also track the usage of each slot with an age memory ( $\mathcal{U}$ ). Thus, the compound memory module ( $\mathcal{M}$ ) can be represented by the following tuple,

$$\mathcal{M} = ((\mathcal{C}_{ns \times nc \times cs}, \mathcal{A}_{ns \times as}), \mathcal{V}_{ns}, \mathcal{U}_{ns}), \quad (6.3)$$

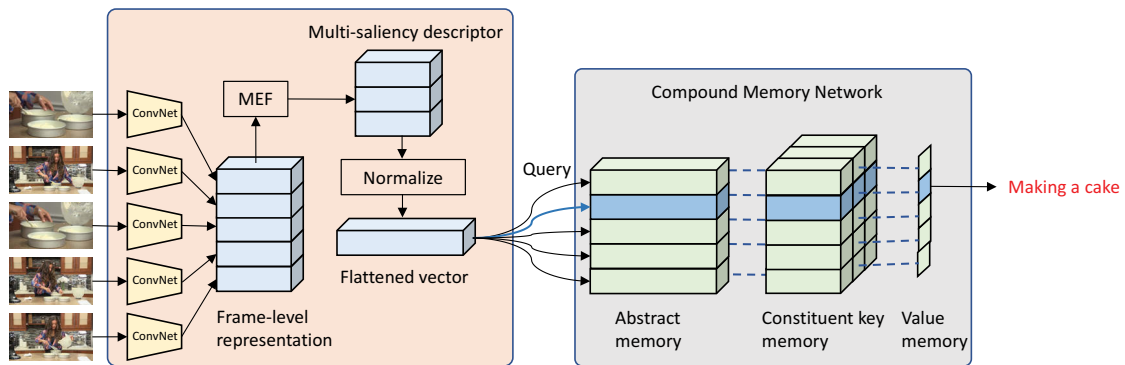


Figure 6.3 : Our CMN structure. A video is first mapped to a matrix representation via the multi-saliency embedding function. This hidden representation is then vectorized and normalized as a query vector, which performs a nearest neighbour search over the abstract memory. The most similar memory slot is retrieved and the label stored in the value memory will be used as the prediction. The constituent key memory contains the matrix representations of the inputs, while the abstract memory is constructed on top of the stacked constituent keys.

where  $\mathbf{ns}$  is the memory size,  $\mathbf{nc}$  is the number of constituent keys,  $\mathbf{cs}$  is the key size and  $\mathbf{as}$  is the abstract memory size.

### *Two-layer Key Memory*

In the constituent key memory, we use multiple stacked constituent keys, which have stronger capability than a single vector, as the visual representation. In CMN, each constituent key is represented by a multi-saliency descriptor.

Note that  $\mathbf{Q}$  is a matrix with shape  $(m, \text{hidden-size})$  and there are  $\mathbf{nc}$  keys in each slot of the constituent key memory. We let  $m$  be equal to  $\mathbf{nc}$ , thus each descriptor in  $\mathbf{Q}$  can be directly saved in the constituent key memory.

To enable fast nearest neighbour query, we introduce an abstract memory on top of the constituent key memory. The stacked keys are compressed to a vector

and it is cached in the abstract memory. The abstract memory can be seen as a snapshot of the constituent key memory. The two sub-memory modules have the same number of slots, but they represent information at different levels.

We denote the stacked matrix representation in  $\mathcal{C}$  as  $\mathbf{C}$ , and each constituent key is  $\mathbf{c}_i$ ,  $i \in \{1, \dots, nc\}$ . We first normalize each constituent key with  $\ell_2$  normalization, *i.e.*,  $\|\mathbf{c}_i\| = 1$ . We then flatten the normalized  $\mathbf{C}'$  to a vector followed by a Fully-Connected (FC) layer, which is then  $\ell_2$ -normalized to a compressed representation. We denote the procedure as the `normalize` function,

$$\mathbf{c}_i' = \frac{\mathbf{c}_i}{\|\mathbf{c}_i\|}, \quad \mathbf{d}' = \text{FC}(\text{flatten}(\mathbf{C}')), \quad \mathbf{d} = \frac{\mathbf{d}'}{\|\mathbf{d}'\|}, \quad (6.4)$$

where a FC layer is simply a linear transformation layer, *i.e.*,  $\text{FC}(\mathbf{x}) = \mathbf{w}\mathbf{x} + b$ . The compressed representation  $\mathbf{d}$  is stored in the abstract memory, which will only be updated when the value in constituent key memory changes. The abstract memory keeps a one-to-one mapping to the constituent key memory, which will accelerate the query process.

### ***Reading***

Given a query vector  $\mathbf{z} = \text{normalize}(\mathbf{Q})$ , nearest neighbour search is conducted over the abstract memory. We select the memory slot that is closest to the query  $\mathbf{z}$  by,  $\text{NN}(\mathbf{z}, \mathcal{A}) = \text{argmax}_i \mathbf{z} \cdot \mathcal{A}[i]$ .  $k$ -nearest slots (ordered by decreasing similarity) can be returned by,

$$(n_1, \dots, n_k) = \text{NN}_k(\mathbf{z}, \mathcal{A}), \quad (6.5)$$

where  $n_1$  is the memory slot that is most similar to the query. At the inference phase,  $\mathcal{V}[n_1]$  will be our prediction for query  $\mathbf{z}$ .

### ***Writing***

The new information should be recorded in the memory to reflect the relation of new query  $\mathbf{z}$  and the corresponding label  $y$ . The memory will not be updated via



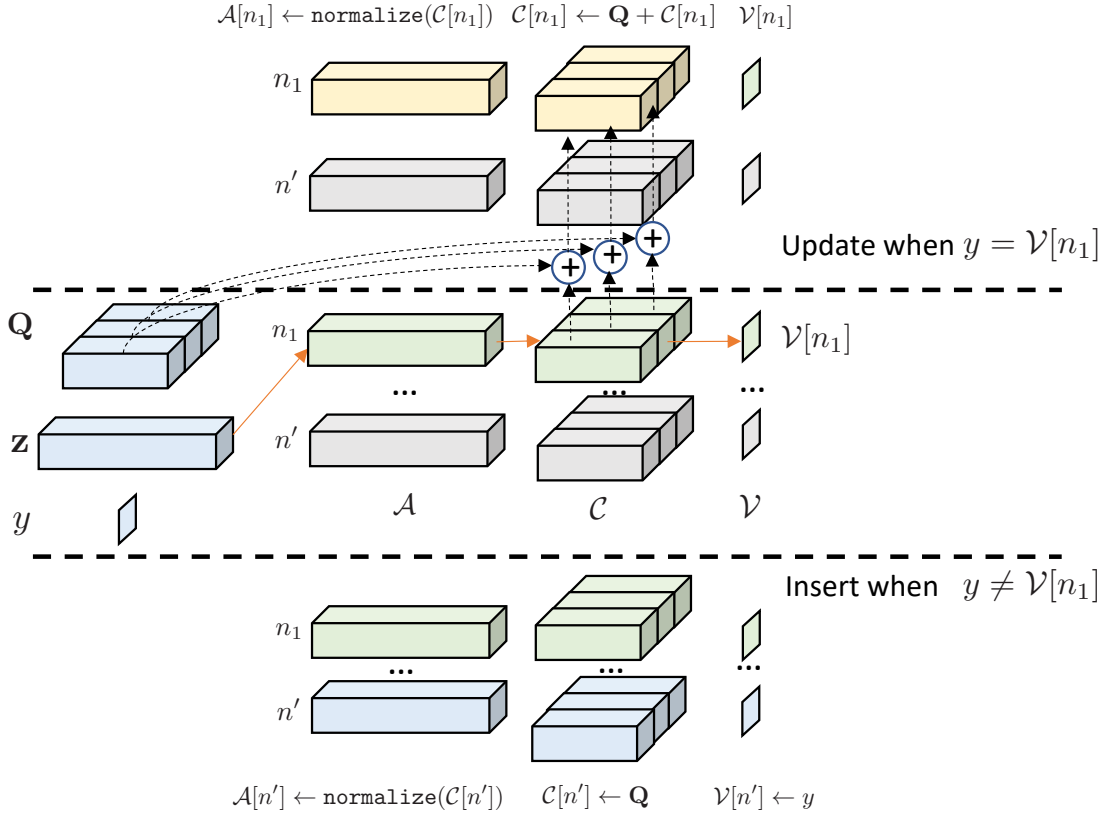


Figure 6.4 : Illustration of the update rule for CMN.

backpropagation which may catastrophically modify the information, but it will be refreshed with the following rule. Note that  $n_1$  is the index of the nearest memory slot, and if the memory already returns the correct label, *i.e.*,  $\mathcal{V}[n_1] = y$ , we only update the  $n_1$  memory slot.  $\mathcal{A}[n_1]$ ,  $\mathcal{U}[n_1]$  and  $\mathcal{C}[n_1]$  will be updated, and leave  $\mathcal{V}[n_1]$  unchanged.

$$\begin{aligned} \mathcal{C}[n_1][i] &\leftarrow \mathbf{q}_i + \mathcal{C}[n_1][i], \quad \text{for } i = 1, \dots, \text{nc}, \\ \mathcal{A}[n_1] &\leftarrow \text{normalize}(\mathcal{C}[n_1]), \quad \mathcal{U}[n_1] \leftarrow 0. \end{aligned} \tag{6.6}$$

The constituent key memory is updated by averaging the constituent keys  $\mathcal{C}[n_1]$  and the multi-saliency descriptors  $\mathbf{Q}$ . The abstract memory  $\mathcal{A}[n_1]$  is updated correspondingly. We also set  $\mathcal{U}[n_1]$  to 0, which shows that slot  $n_1$  has just been updated.

When  $\mathcal{V}[n_1] \neq y$ , the  $(\mathbf{Q}, y)$  pair is stored in another memory slot to record the information. We choose the oldest memory slot  $n'$  that has not been updated for a long time,

$$n' = \arg \max_i (\mathcal{U}[i] + r_i), \quad (6.7)$$

where  $r_i$  is a random number to introduce randomness during slot selection. The memory will be updated by,

$$\begin{aligned} \mathcal{C}[n'][i] &\leftarrow \mathbf{q}_i, \quad \text{for } i = 1, \dots, \mathbf{nc}, \\ \mathcal{A}[n'] &\leftarrow \text{normalize}(\mathcal{C}[n']), \quad \mathcal{V}[n'] \leftarrow y, \quad \mathcal{U}[n'] \leftarrow 0. \end{aligned} \quad (6.8)$$

In this case,  $\mathcal{V}[n']$  is also updated with the new label  $y$ . We illustrate the procedure in Figure. 6.4.

### 6.3.3 Training

Given a query  $\mathbf{z}$  and a corresponding ground-truth label  $y$ , we retrieve top- $k$  key-value pairs on memory indices  $(n_1, \dots, n_k)$  by Eq. 6.5. Let **i-pos** be the smallest index that  $\mathcal{V}[n_{\mathbf{i-pos}}] = y$  and **i-neg** be the smallest index that  $\mathcal{V}[n_{\mathbf{i-neg}}] \neq y$ . We train the query vector  $\mathbf{z}$  to be more similar to  $\mathcal{A}[n_{\mathbf{i-pos}}]$  than  $\mathcal{A}[n_{\mathbf{i-neg}}]$  with the following ranking loss,

$$\mathcal{L}(\mathbf{z}, y, \mathcal{A}) = \max(\alpha - \mathbf{z} \cdot \mathcal{A}[n_{\mathbf{i-pos}}] + \mathbf{z} \cdot \mathcal{A}[n_{\mathbf{i-neg}}], 0). \quad (6.9)$$

The similarity between the query and the positive key should be larger than the similarity between the query and the negative key by margin  $\alpha$ . The loss will be 0 when the difference between the two similarities is beyond margin  $\alpha$ .

The memory in each episode is cleared before operations are conducted. The clear operation simply initializes all memory variables to 0. During mini-batch training, information from multiple episodes are stored in the global memory. To avoid confliction in the label space, label ids across episodes should be different. The

global label id can be calculated by,

$$\text{global-label-id} = \text{label-id} + \text{index} \times \mathbf{k}, \quad (6.10)$$

where  $\mathbf{k}$  is the number of classes, `label-id` is the shuffled label id in an episode and `index` is the index of the episode in the mini-batch. At the inference phase, the weights of the network are fixed except for the memory module, which will be updated with the support set examples.

## 6.4 Experiments

### 6.4.1 Datasets

There are no existing datasets for few-shot video classification, thus we collected the first dataset for few-shot video classification evaluation, which we will release for future research. We used videos from the recently released Kinetics dataset [47], which consists of 400 categories and 306,245 videos, covering videos from a wide range of actions and events, e.g., “dribbling basketball”, “robot dancing”, “shaking hands”, “playing violin”. We randomly selected 100 classes from the Kinetics dataset, each of which contains 100 examples. The 100 classes were split into 64, 12 and 24 non-overlapping classes for use as the meta-training set, meta-validation set and meta-testing set, respectively.

### 6.4.2 Implementation Details

In an  $n$ -way,  $k$ -shot problem, we randomly sampled  $n$  classes and each class has  $k$  examples, while an additional unlabeled example belonging to one of the  $n$  classes is used for testing. Thus each episode has  $nk + 1$  examples. We calculated the mean accuracy by randomly sampling 20,000 episodes in all experiments.

To obtain the frame-level feature representation, we forwarded each frame to a ResNet-50 [36] network that was pre-trained on ImageNet. We followed the basic

Model	1-shot	2-shot	3-shot	4-shot	5-shot
RGB w/o mem	28.7	36.8	42.6	46.2	48.6
Flow w/o mem	24.4	27.3	29.8	32.0	33.1
LSTM(RGB) w/o mem	28.9	37.5	43.3	47.1	49.0
Nearest-finetune	48.2	55.5	59.1	61.0	62.6
Nearest-pretrain	51.1	60.4	64.8	67.1	68.9
MatchingNet [115]	53.3	64.3	69.2	71.8	74.6
MAML [26]	54.2	65.5	70.0	72.1	75.3
Plain CMN [44]	57.3	67.5	72.5	74.7	76.0
LSTM-emb	57.6	67.9	72.8	74.8	76.2
Ours	<b>60.5</b>	<b>70.0</b>	<b>75.6</b>	<b>77.3</b>	<b>78.9</b>

Table 6.1 : Results of 5-way few-shot video classification on the meta-testing set. The numbers are reported in percentages. Our CMN achieves state-of-the-art results.

image preprocessing procedure, whereby the image was first rescaled by resizing the short side to 256 and a  $224 \times 224$  region was randomly cropped from the image. We cropped the central region during the inference phase.

We optimized our model with Adam [48] and fixed the learning rate to  $1.0 \times 10^{-4}$ . The margin  $\alpha$  was set to 0.5 in all experiments. We tuned the hyper-parameters on the meta-validation set, and stopped the training process when the accuracy on the meta-validation set began to decrease. The model was implemented with the TensorFlow framework [3].

### 6.4.3 Evaluation

We compare our model with several baselines. We report 1-shot, 2-shot, 3-shot, 4-shot and 5-shot results on the 5-way classification task. In the first baseline, we utilize all the training data to pre-train the ResNet-50 network. At the testing stage, we fine-tune the network for each episode. The network is initialized with the pre-trained weights up to the last layer. The weights in the last layer is randomly initialized. We test the performance with different inputs. For “RGB w/o mem”, we take RGB frames as inputs to train the network. For “Flow w/o mem”, stack flows images are stacked as inputs to the network. To encode video with more sophisticated embedding function upon the frame-level features, we use an LSTM to aggregate temporal dynamics in a video. The LSTM takes the RGB features as inputs. It is fine-tuned for each episode. We denote this baseline as “LSTM (RGB) w/o mem”. Another baseline is a nearest neighbour baseline (“Nearest-finetune”). We first finetune the ResNet-50 network to classify all classes in the meta-training set. We feed each frame as the input image and the video-level label is used as the label for each frame. Frames are first preprocessed with the procedure described above. We initialize the weights of the ResNet-50 network with the ImageNet pre-trained model. We train the network via stochastic gradient descent (SGD) with momentum 0.9. We set the initial learning rate to 0.01. We decrease the learning rate by 0.1 every 10 epochs. The batch size is 128. During inference, we feed the video frames to the finetuned ResNet-50 network and extract the activations from the last layer before final classification. We average the frame-level features and obtain a video-level representation of 2,048 dimension. We also apply  $\ell_2$  normalization before nearest neighbour search.

In the next baseline (“Nearest-pretrain”), we do not finetune the ResNet-50 network on the meta-training dataset, but directly utilize the pre-trained weights without modification. We embed the video with the same procedure in “Nearest-

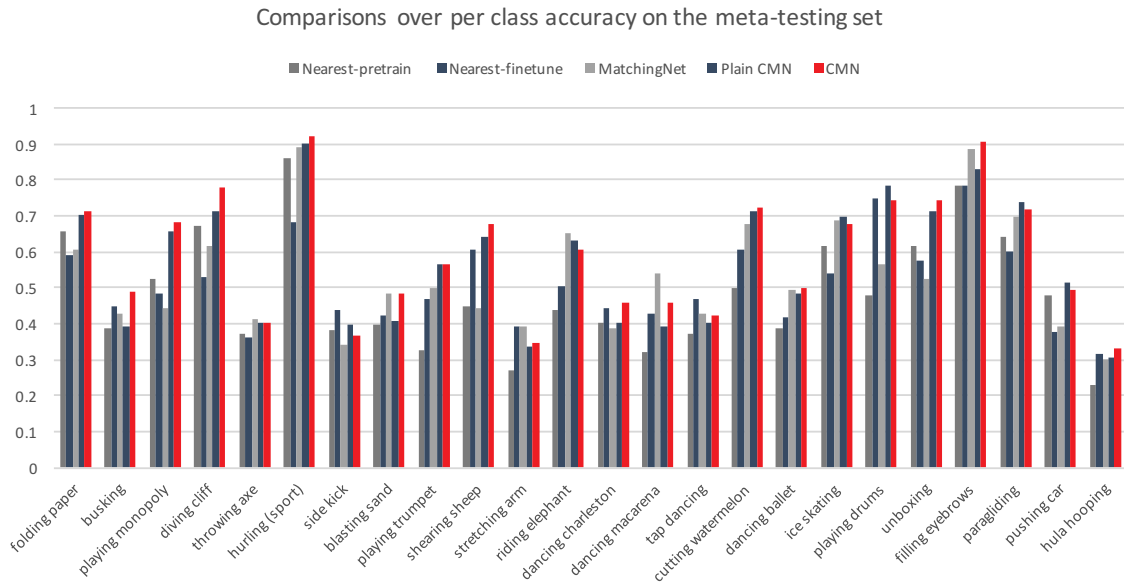


Figure 6.5 : Per class accuracy on the 5-way 1-shot setting. We show the accuracies of 24 classes on the meta-testing set.

finetune”, and then apply nearest neighbour search.

We also show the result of the Matching Network [115] (“MatchingNet”) on this dataset, which achieves state-of-the-art performance on the few-shot image classification task. We implement the Matching Network algorithms ourselves. We first feed the frames to a ResNet-50 network without fine-tuning. We average frame-level features to obtain a video-level feature. We then use the fully-conditional embedding (FCE) function proposed in [115] to embed the training examples. The FCE uses a bidirectional-LSTM and each training example is a function of all the other examples. To train MAML [26], we average the frame-level features and follow the default hyper-parameters in [26].

Another baseline is “Plain CMN” where we remove the constituent key memory from the model and use a video-level vector as video representation. We replace our embedding module with an LSTM function, while keeping the other settings the same. We denote this baseline as “LSTM-emb”. We conduct this baseline to show

Model	1-shot	2-shot	3-shot	4-shot	5-shot
Mem-64	52.0	61.9	66.5	69.4	71.2
Mem-128	53.4	63.7	68.9	71.5	73.5
Mem-512	<b>55.1</b>	<b>65.3</b>	<b>70.1</b>	72.0	<b>74.2</b>
Mem-2048	55.0	65.0	69.7	<b>72.4</b>	74.1

Table 6.2 : Results of different memory sizes.

the effectiveness of our compound memory network structure.

The results are shown in Table 6.1. We can see from Table 6.1 that our CMN improves the baselines in all shots. We observe that fine-tuning the ResNet-50 network on the meta-training set does not improve the few-shot video classification performance, but significantly harms performance. As there are no overlapping classes between the meta-training set and the meta-testing set, it is very likely that the model will overfit the meta-training set. Our CMN structure also outperforms the Matching Networks by more than 4% across all shots. Furthermore, our CMN structure outperforms the “Plain CMN”, which demonstrates the strong representation capability of the constituent key memory. About 10% improvement is obtained between the 1-shot setting and the 2-shot setting, by only adding one example per class. The relative improvement decreases when more examples are added, e.g., the improvement from 3-shot to 4-shot is only 1.7%. This shows that one-shot classification is still a difficult problem which can be further improved in the future.

The 1-shot accuracy for each class is shown in Figure 6.5. We report the mean accuracy for class  $c$  over all episodes where the query label is  $c$ . The “hurling (sport)” category have the highest accuracy, while “hula hooping” and “stretching arms” achieve the worst performance with about 30% accuracy.

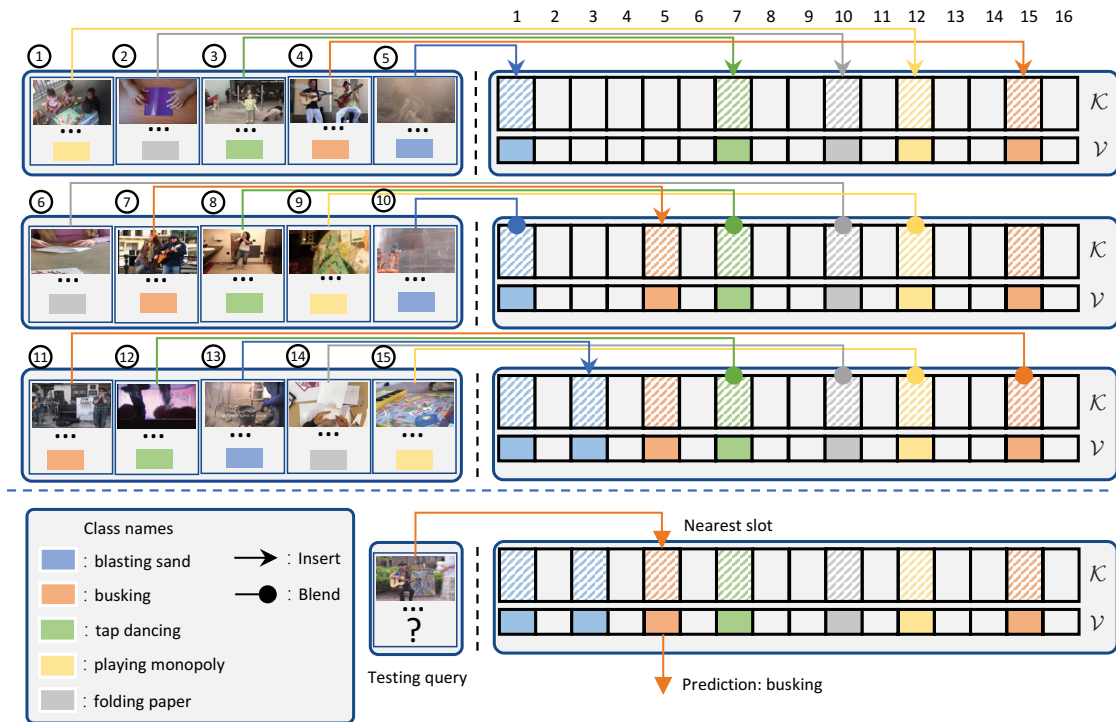


Figure 6.6 : We illustrate the inference procedure. There are 5 classes and the memory has 16 slots. Two different update rules will be used depending on the query results.

We illustrate the inference procedure in an episode in Figure 6.6. In this 5-way 3-shot setting, the support set has 15 examples. Each example is sequentially fed to the network. This episode is divided into three groups, each of which has five examples with distinct labels. We arrange the episode in this way for better illustration. In row 1, all inputs are inserted into the memory. In row 2, the 7th example is inserted into a new slot in the memory, while other videos are blended into existing slots of the same category. In row 3, the 13th example is inserted. For the 11th example, the closest slot is the 15th slot and the two representations are averaged.



Model	1-shot	2-shot	3-shot	4-shot	5-shot
Desc-1	53.7	63.5	68.3	70.9	73.3
Desc-5	<b>55.1</b>	<b>65.3</b>	<b>70.1</b>	<b>72.0</b>	<b>74.2</b>
Desc-10	53.2	62.9	68.2	70.0	72.3

Table 6.3 : Results of different numbers of multi-saliency descriptors.

#### 6.4.4 Ablation Study

We perform ablation experiments to explain our selections for the final model. The default setting is the 5-way few-shot classification. We show the performance of different memory sizes in Table 6.2, and the results of different numbers of constituent keys are shown in Table 6.3. We also report the results of other few-shot video classification tasks with different numbers of categories. We report the results on the meta-validation set, and choose only 10 frames during evaluation.

**Memory size.** The results of different memory sizes are shown in Table 6.2. When the memory has a small number of slots, the performance is worse because some information has to be wiped out when new data arrives. Memory size of 512 achieves the best results. Increasing the memory size does not improve performance when the memory is large enough to record all the information.

**The number of multi-saliency descriptors.** The result is shown in Table 6.3. It shows that multi-saliency descriptors with stronger representation capability obtain better performance than a single descriptor. The performance decreases when too many descriptors are used, because more parameters are introduced in the network.

**$N$ -way classification.** In all previous experiments, evaluations were conducted on the 5-way classification setting.  $n$ -way classification with larger  $n$  is a similar task to 5-way classification, but can be more difficult. As can be seen, the performance

Model	1-shot	2-shot	3-shot	4-shot	5-shot
5-way	55.0	65.0	69.7	72.4	74.1
6-way	51.7	61.8	66.4	69.3	71.2
7-way	49.5	59.6	64.3	67.1	68.9
8-way	46.0	56.1	61.0	64.0	65.8

Table 6.4 : Results of different way few-shot video classification.

decreases when  $n$  increases.

## 6.5 Conclusion

We have proposed a compound memory network for few-shot video classification. This module stores matrix representations, which can be easily retrieved and updated in an efficient way. Our future work is to leverage multiple memory banks of different modality representations.

## Chapter 7

### Future Works

In this dissertation, I investigated methods for video representation learning under different settings. I have shown the importance of leveraging sequence relationships for video temporal modeling. Recurrent Neural Networks have been heavily used in this dissertation, which mainly focus on learning global representation from local features. More research studies can conduct on using 3D convolutional neural networks for spatio-temporal modeling on clip-level feature learning. Another future direction for video-level aggregation is to leverage 1D convolution on top of the local features. For better aggregation, attention mechanism can also be exploited for global feature encoding. Apart from video classification, future studies can be tried on action detection using our multi-rate GRU structure.

The better abstract knowledge from web data or episode sequences, the abstraction structure can be tailored to be a universal controller for retrieving knowledge from relevant or irrelevant candidates. For few-shot video classification, more gradient-based methods can be studied, where self-supervised signals and optical flow correspondence are also important cues to the success of few-example temporal modeling.

## Bibliography

- [1] “TRECVID MED 13.” <http://nist.gov/itl/iad/mig/med13.cfm>, 2013.
- [2] “TRECVID MED 14.” <http://nist.gov/itl/iad/mig/med14.cfm>, 2014.
- [3] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: A system for large-scale machine learning,” in *OSDI*, 2016.
- [4] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh, “VQA: Visual question answering,” in *ICCV*, 2015.
- [5] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, “Dbpedia: A nucleus for a web of open data,” in *The semantic web*. Springer, 2007, pp. 722–735.
- [6] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization,” *arXiv preprint arXiv:1607.06450*, 2016.
- [7] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” in *ICLR*, 2015.
- [8] N. Ballas, L. Yao, C. Pal, and A. Courville, “Delving deeper into convolutional networks for learning video representations,” *ICLR*, 2016.
- [9] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” *TNN*, vol. 5, no. 2, pp. 157–166, 1994.

- [10] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.
- [11] J. Carreira and A. Zisserman, “Quo vadis, action recognition? a new model and the kinetics dataset,” in *CVPR*, 2017.
- [12] X. Chang, Y. Yang, E. P. Xing, and Y.-L. Yu, “Complex event detection using semantic saliency and nearly-isotonic svm,” in *ICML*, 2015.
- [13] D. L. Chen and W. B. Dolan, “Collecting highly parallel data for paraphrase evaluation,” in *ACL*, 2011.
- [14] 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.
- [15] K. Cho, B. van Merriënboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” in *EMNLP*, 2015.
- [16] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” *arXiv preprint arXiv:1412.3555*, 2014.
- [17] R. Collobert, K. Kavukcuoglu, and C. Farabet, “Torch7: A matlab-like environment for machine learning,” in *Conference on Neural Information Processing Systems Workshops (NIPS Workshops)*, 2011.
- [18] N. Dalal, B. Triggs, and C. Schmid, “Human detection using oriented histograms of flow and appearance,” in *ECCV*, 2006.
- [19] M. Denkowski and A. Lavie, “Meteor Universal: Language specific translation evaluation for any target language,” in *EACL*, 2014.

- [20] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, “Long-term recurrent convolutional networks for visual recognition and description,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [21] 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 *CVPR*, 2015.
- [22] D. Elliott and F. Keller, “Comparing automatic evaluation measures for image description,” in *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2014.
- [23] B. G. Fabian Caba Heilbron, Victor Escorcia and J. C. Niebles, “ActivityNet: A large-scale video benchmark for human activity understanding,” in *CVPR*, 2015.
- [24] L. Fei-Fei, R. Fergus, and P. Perona, “One-shot learning of object categories,” *TPAMI*, vol. 28, no. 4, pp. 594–611, 2006.
- [25] C. Feichtenhofer, A. Pinz, and A. Zisserman, “Convolutional two-stream network fusion for video action recognition,” in *CVPR*, 2016.
- [26] C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” in *ICML*, 2017.
- [27] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, M. A. Ranzato, and T. Mikolov, “DeViSE: A deep visual-semantic embedding model,” in *Conference on Neural Information Processing Systems (NIPS)*, 2013.
- [28] H. Gao, J. Mao, J. Zhou, Z. Huang, L. Wang, and W. Xu, “Are you talking to a machine? Dataset and methods for multilingual image question

- answering,” in *Conference on Neural Information Processing Systems (NIPS)*, 2015.
- [29] R. Girshick, “Fast R-CNN,” in *ICCV*, 2015.
- [30] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.
- [31] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks.” in *AISTATS*, 2010.
- [32] Y. Gong, Q. Ke, M. Isard, and S. Lazebnik, “A multi-view embedding space for modeling internet images, tags, and their semantics,” *International Journal of Computer Vision (IJCV)*, vol. 106, no. 2, pp. 210–233, 2014.
- [33] A. Graves, G. Wayne, and I. Danihelka, “Neural turing machines,” *arXiv preprint arXiv:1410.5401*, 2014.
- [34] B. Hariharan and R. Girshick, “Low-shot visual recognition by shrinking and hallucinating features,” in *ICCV*, 2017.
- [35] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *CVPR*, 2016.
- [36] ———, “Identity mappings in deep residual networks,” in *ECCV*, 2016.
- [37] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [38] M. Hodosh, P. Young, and J. Hockenmaier, “Framing image description as a ranking task: Data, models and evaluation metrics,” *Journal of Artificial Intelligence Research (JAIR)*, pp. 853–899, 2013.

- [39] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *ICML*, 2015.
- [40] A. Jabri, A. Joulin, and L. van der Maaten, “Revisiting visual question answering baselines,” in *European Conference on Computer Vision (ECCV)*. Springer, 2016.
- [41] H. Jegou, M. Douze, C. Schmid, and P. Pérez, “Aggregating local descriptors into a compact image representation,” in *CVPR*, 2010.
- [42] S. Ji, W. Xu, M. Yang, and K. Yu, “3d convolutional neural networks for human action recognition,” *TPAMI*, 2013.
- [43] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu, “Exploring the limits of language modeling,” *arXiv preprint arXiv:1602.02410*, 2016.
- [44] L. Kaiser, O. Nachum, A. Roy, and S. Bengio, “Learning to remember rare events,” in *ICLR*, 2017.
- [45] A. Karpathy and L. Fei-Fei, “Deep visual-semantic alignments for generating image descriptions,” in *CVPR*, 2015.
- [46] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, “Large-scale video classification with convolutional neural networks,” in *CVPR*, 2014.
- [47] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev *et al.*, “The kinetics human action video dataset,” *arXiv preprint arXiv:1705.06950*, 2017.
- [48] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *ICLR*, 2015.



- [49] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler, “Skip-thought vectors,” in *NIPS*, 2015.
- [50] D. Klein and C. D. Manning, “Accurate unlexicalized parsing,” in *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2003.
- [51] G. Koch, “Siamese neural networks for one-shot image recognition,” Ph.D. dissertation, University of Toronto, 2015.
- [52] J. Koutnik, K. Greff, F. Gomez, and J. Schmidhuber, “A clockwork RNN,” in *ICML*, 2014.
- [53] I. Krasin, T. Duerig, N. Alldrin, A. Veit, S. Abu-El-Haija, S. Belongie, D. Cai, Z. Feng, V. Ferrari, V. Gomes, A. Gupta, D. Narayanan, C. Sun, G. Chechik, and K. Murphy, “OpenImages: A public dataset for large-scale multi-label and multi-class image classification,” *Dataset available from <https://github.com/openimages>*, 2016.
- [54] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *NIPS*, 2012.
- [55] G. Kulkarni, V. Premraj, S. Dhar, S. Li, Y. Choi, A. C. Berg, and T. L. Berg, “Baby talk: Understanding and generating image descriptions,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.
- [56] B. Lake, R. Salakhutdinov, J. Gross, and J. Tenenbaum, “One shot learning of simple visual concepts,” in *CogSci*, 2011.
- [57] Z. Lan, M. Lin, X. Li, A. G. Hauptmann, and B. Raj, “Beyond Gaussian pyramid: Multi-skip feature stacking for action recognition,” in *CVPR*, 2015.

- [58] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, “Learning realistic human actions from movies,” in *CVPR*, 2008.
- [59] R. Lebrecht, P. O. Pinheiro, and R. Collobert, “Phrase-based image captioning,” in *International Conference on Machine Learning (ICML)*, 2015.
- [60] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [61] 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 (ECCV)*, 2014.
- [62] X. Lin and D. Parikh, “Don’t just listen, use your imagination: Leveraging visual common sense for non-visual tasks,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [63] Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, “A structured self-attentive sentence embedding,” in *ICLR*, 2017.
- [64] M. Malinowski and M. Fritz, “A multi-world approach to question answering about real-world scenes based on uncertain input,” in *Conference on Neural Information Processing Systems (NIPS)*, 2014.
- [65] M. Malinowski, M. Rohrbach, and M. Fritz, “Ask your neurons: A neural-based approach to answering questions about images,” in *International Conference on Computer Vision (ICCV)*, 2015.
- [66] T. Malisiewicz, A. Gupta, and A. A. Efros, “Ensemble of exemplar-SVMs for object detection and beyond,” in *ICCV*, 2011.

- [67] J. Mao, J. Huang, A. Toshev, O. Camburu, A. Yuille, and K. Murphy, “Generation and comprehension of unambiguous object descriptions,” *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [68] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” in *ICLR*, 2013.
- [69] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *Conference on Neural Information Processing Systems (NIPS)*, 2013.
- [70] A. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston, “Key-value memory networks for directly reading documents,” in *EMNLP*, 2016.
- [71] E. G. Miller, N. E. Matsakis, and P. A. Viola, “Learning from one example through shared densities on transforms,” in *CVPR*, 2000.
- [72] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, “Beyond short snippets: Deep networks for video classification,” in *CVPR*, 2015.
- [73] V. Ordonez, X. Han, P. Kuznetsova, G. Kulkarni, M. Mitchell, K. Yamaguchi, K. Stratos, A. Goyal, J. Dodge, A. Mensch *et al.*, “Large scale retrieval and generation of image descriptions,” *International Journal of Computer Vision (IJCV)*, pp. 1–14, 2015.
- [74] P. Pan, Z. Xu, Y. Yang, F. Wu, and Y. Zhuang, “Hierarchical recurrent neural encoder for video representation with application to captioning,” in *CVPR*, 2016.
- [75] Y. Pan, T. Mei, T. Yao, H. Li, and Y. Rui, “Jointly modeling embedding and translation to bridge video and language,” in *CVPR*, 2016.

- [76] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “BLEU: A method for automatic evaluation of machine translation,” in *ACL*, 2002.
- [77] V. Pătrăucean, A. Handa, and R. Cipolla, “Spatio-temporal video autoencoder with differentiable memory,” in *ICLR Workshop*, 2016.
- [78] F. Perronnin, J. Sánchez, and T. Mensink, “Improving the Fisher kernel for large-scale image classification,” in *ECCV*, 2010.
- [79] V. Pham, T. Bluche, C. Kermorvant, and J. Louradour, “Dropout improves recurrent neural networks for handwriting recognition,” in *ICFHR*, 2014.
- [80] Z. Qiu, T. Yao, and T. Mei, “Learning spatio-temporal representation with pseudo-3d residual networks,” in *ICCV*, 2017.
- [81] S. Ravi and H. Larochelle, “Optimization as a model for few-shot learning,” in *ICLR*, 2017.
- [82] M. Regneri, M. Rohrbach, D. Wetzel, S. Thater, B. Schiele, and M. Pinkal, “Grounding action descriptions in videos,” *Transactions of the Association for Computational Linguistics (TACL)*, vol. 1, pp. 25–36, 2013.
- [83] M. Ren, R. Kiros, and R. S. Zemel, “Exploring models and data for image question answering,” in *Conference on Neural Information Processing Systems (NIPS)*, 2015.
- [84] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” in *NIPS*, 2015.
- [85] A. Rohrbach, M. Rohrbach, N. Tandon, and B. Schiele, “A dataset for movie description,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

- [86] M. Rohrbach, W. Qiu, I. Titov, S. Thater, M. Pinkal, and B. Schiele, “Translating video content to natural language descriptions,” in *International Conference on Computer Vision (ICCV)*, 2013.
- [87] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy *et al.*, “ImageNet large scale visual recognition challenge,” *International Journal of Computer Vision (IJCV)*, pp. 1–42, 2014.
- [88] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “ImageNet Large Scale Visual Recognition Challenge,” *IJCV*, vol. 115, no. 3, pp. 211–252, 2015.
- [89] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, “Meta-learning with memory-augmented neural networks,” in *ICML*, 2016.
- [90] D. L. Schacter, D. R. Addis, D. Hassabis, V. C. Martin, R. N. Spreng, and K. K. Szpunar, “The future of memory: Remembering, imagining, and the brain,” *Neuron*, vol. 76, no. 4, pp. 677–694, 2012.
- [91] K. Simonyan and A. Zisserman, “Two-stream convolutional networks for action recognition in videos,” in *NIPS*, 2014.
- [92] —, “Very deep convolutional networks for large-scale image recognition,” in *ICLR*, 2015.
- [93] J. Snell, K. Swersky, and R. S. Zemel, “Prototypical networks for few-shot learning,” in *NIPS*, 2017.
- [94] K. Soomro, A. R. Zamir, and M. Shah, “UCF101: A dataset of 101 human actions classes from videos in the wild,” *arXiv preprint arXiv:1212.0402*, 2012.

- [95] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *JMLR*, vol. 15, no. 1, pp. 1929–1958, 2014.
- [96] N. Srivastava, E. Mansimov, and R. Salakhutdinov, “Unsupervised learning of video representations using LSTMs,” in *ICML*, 2015.
- [97] S. Sukhbaatar, J. Weston, R. Fergus *et al.*, “End-to-end memory networks,” in *NIPS*, 2015.
- [98] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *NIPS*, 2014.
- [99] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [100] —, “Going deeper with convolutions,” in *CVPR*, 2015.
- [101] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *CVPR*, 2016.
- [102] M. Tapaswi, Y. Zhu, R. Stiefelhagen, A. Torralba, R. Urtasun, and S. Fidler, “MovieQA: Understanding stories in movies through question-answering,” in *CVPR*, 2016.
- [103] T. Tieleman and G. Hinton, “Lecture 6.5-RMSprop: Divide the gradient by a running average of its recent magnitude.” 2012.
- [104] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, “Learning spatiotemporal features with 3D convolutional networks,” in *ICCV*, 2015.
- [105] —, “Learning spatiotemporal features with 3d convolutional networks,” in *ICCV*, 2015.

- [106] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, “A closer look at spatiotemporal convolutions for action recognition,” in *CVPR*, 2018.
- [107] K. Tu, M. Meng, M. W. Lee, T. E. Choe, and S.-C. Zhu, “Joint video and text parsing for understanding events and answering queries,” *MultiMedia, IEEE*, vol. 21, no. 2, pp. 42–70, 2014.
- [108] L. Van der Maaten and G. Hinton, “Visualizing data using t-SNE,” *Journal of Machine Learning Research (JMLR)*, vol. 9, no. 2579-2605, p. 85, 2008.
- [109] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *NIPS*, 2017.
- [110] R. Vedantam, C. Lawrence Zitnick, and D. Parikh, “CIDEr: Consensus-based image description evaluation,” in *CVPR*, 2015.
- [111] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko, “Sequence to sequence – video to text,” in *ICCV*, 2015.
- [112] ———, “Sequence to sequence-video to text,” in *ICCV*, 2015.
- [113] S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. Mooney, and K. Saenko, “Translating videos to natural language using deep recurrent neural networks,” in *NAACL HLT*, 2015.
- [114] O. Vinyals, S. Bengio, and M. Kudlur, “Order matters: Sequence to sequence for sets,” in *ICLR*, 2016.
- [115] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra *et al.*, “Matching networks for one shot learning,” in *NIPS*, 2016.
- [116] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, “Show and tell: A neural image caption generator,” in *CVPR*, 2015.

- [117] C. Vondrick, H. Pirsiavash, and A. Torralba, “Anticipating the future by watching unlabeled video,” *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [118] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, “Action recognition by dense trajectories,” in *CVPR*, 2011.
- [119] H. Wang and C. Schmid, “Action recognition with improved trajectories,” in *ICCV*, 2013.
- [120] L. Wang, Y. Qiao, and X. Tang, “Action recognition with trajectory-pooled deep-convolutional descriptors,” in *CVPR*, 2015.
- [121] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, “Temporal segment networks: Towards good practices for deep action recognition,” in *ECCV*, 2016.
- [122] J. Weston, A. Bordes, S. Chopra, A. M. Rush, B. van Merriënboer, A. Joulin, and T. Mikolov, “Towards AI-complete question answering: A set of prerequisite toy tasks,” *arXiv preprint arXiv:1502.05698*, 2015.
- [123] J. Weston, S. Chopra, and A. Bordes, “Memory networks,” in *ICLR*, 2015.
- [124] Q. Wu, P. Wang, C. Shen, A. Dick, and A. van den Hengel, “Ask me anything: Free-form visual question answering based on knowledge from external sources,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [125] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, “Google’s neural machine translation system: Bridging the gap between human and machine translation,” *arXiv preprint arXiv:1609.08144*, 2016.



- [126] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy, “Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification,” in *ECCV*, 2018.
- [127] K. Xu, J. Ba, R. Kiros, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio, “Show, attend and tell: Neural image caption generation with visual attention,” in *ICML*, 2015.
- [128] Z. Xu, Y. Yang, and A. G. Hauptmann, “A discriminative CNN video representation for event detection,” in *CVPR*, 2015.
- [129] Z. Xu, L. Zhu, and Y. Yang, “Few-shot object recognition from machine-labeled web images,” in *CVPR*, 2017.
- [130] Z. Xu, L. Zhu, Y. Yang, and A. G. Hauptmann, “UTS-CMU at THUMOS 2015,” *THUMOS Challenge*, 2015.
- [131] L. Yao, A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, and A. Courville, “Describing videos by exploiting temporal structure,” in *ICCV*, 2015.
- [132] P. Young, A. Lai, M. Hodosh, and J. Hockenmaier, “From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions,” *Transactions of the Association for Computational Linguistics (TACL)*, vol. 2, pp. 67–78, 2014.
- [133] H. Yu and J. M. Siskind, “Grounded language learning from video described with sentences.” in *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2013.
- [134] H. Yu, J. Wang, Z. Huang, Y. Yang, and W. Xu, “Video paragraph captioning using hierarchical recurrent neural networks,” in *CVPR*, 2016.

- [135] L. Yu, E. Park, A. C. Berg, and T. L. Berg, “Visual Madlibs: Fill in the blank image generation and question answering,” in *International Conference on Computer Vision (ICCV)*, 2015.
- [136] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, “Beyond short snippets: Deep networks for video classification,” in *CVPR*, 2015.
- [137] W. Zaremba, I. Sutskever, and O. Vinyals, “Recurrent neural network regularization,” *arXiv preprint arXiv:1409.2329*, 2014.
- [138] S. Zha, F. Luisier, W. Andrews, N. Srivastava, and R. Salakhutdinov, “Exploiting image-trained CNN architectures for unconstrained video classification,” in *BMVC*, 2015.
- [139] Y. Zhou, X. Sun, Z.-J. Zha, and W. Zeng, “Mict: Mixed 3d/2d convolutional tube for human action recognition,” in *CVPR*, 2018.
- [140] L. Zhu, Z. Xu, and Y. Yang, “Bidirectional multirate reconstruction for temporal modeling in videos,” in *CVPR*, 2017.
- [141] L. Zhu, Z. Xu, Y. Yang, and A. G. Hauptmann, “Uncovering temporal context for video question and answering,” *arXiv preprint arXiv:1511.04670*, 2015.
- [142] Y. Zhu, O. Groth, M. Bernstein, and L. Fei-Fei, “Visual7w: Grounded question answering in images,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [143] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, “Aligning books and movies: Towards story-like visual explanations by watching movies and reading books,” in *International Conference on Computer Vision (ICCV)*, 2015.