上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

学士学位论文

THESIS OF BACHELOR

论文题目： Temporal Modeling in Action Recognition

学生姓名： 朱鑫祺

学生学号： 5140219240

专　　业： 计算机科学与技术

指导教师： Dacheng Tao　卢策吾

学院(系)： 计算机科学与工程系

行为识别中的时序建模

摘　要

近年来，视频中的行为识别成为计算机视觉领域一个饱受关注的问题。在这篇本科毕业论文中，我们从时序建模的角度来研究基于视频的行为识别问题。本论文的动机源自于最近一些对现有模型时序建模能力的疑问，以及我们对探究更好的时序利用方法的好奇。我们首先调研了近五年内流行的基于深度学习的行为识别方法，并对它们按照不同的处理时序动态的方式进行分类。鉴于我们希望能量化地描述一个模型在一个数据集上的时序建模能力，我们提出了能够检测模型对时间顺序性的敏感程度，和时间顺序对模型训练的影响程度的评测方法。我们所提出的敏感因子和影响因子的计算无需样本的标注信息。接下来我们在 UCF101 和 Something-Something 两个数据集上对所选出的有代表性的已有模型用我们的时序评测方法进行量化评测，并提供我们的讨论与分析。从实验中，我们的得出了一些很有意义和启发的结论，其中一些甚至有违直觉。例如相比于其他方法，基于长短时记忆网络的模型并不能太好地利用时序信息；3 维的卷积模型在视频分类时会着重依赖于时序的顺序性；我们也发现一个数据集对时序建模的要求可以用我们提出的敏感因子来量化估计。除了调研已有的方法，我们也在论文中提出了一个新的模型来处理行为识别中的时序建模，即近似双线性融合模型。我们的模型由双线性池化模型拓展而来，通过低秩近似来显著减少所需训练的参数，并用以接受多帧图片来捕捉视频中的长时序信息。我们给出低秩近似的推导过程，并解释如何实现有效的多帧融合。最后我们展示了所提出模型的探究实验，并在 Something-Something 和 UCF101 数据集上检验了模型的识别效果以及对时序建模的能力。实验结果显示我们的模型可以超过或者追赶上其他模型，同时保持低复杂度和快速的收敛。

关键词：行为识别，时序建模，评测，双线性模型

TEMPORAL MODELING IN ACTION RECOGNITION

ABSTRACT

Nowadays, action recognition in videos has become more and more popular in computer vision. In this thesis, we study the video based action recognition from the perspective of temporal modeling. Our motivation comes from some recent questions about temporal modeling, and our curiosity for better ways to exploit temporal information in videos. We first review the most promising deep learning based action recognition methods proposed within the last five years, and categorize them based on their main contributions for modeling temporal dynamics. Since we target at quantitatively describe a model's capability of temporal modeling on a dataset, we propose our evaluation protocols to measure a model's sensitivity about temporal coherency, and how temporal coherency influences a model's training. The calculations of the proposed Sens-Index and Inf-Index do not require the revealing of labels. We then evaluate representative reviewed models using our protocols on UCF101 and Something-Something datasets, and provide our discussions and analyses. There are many meaningful and inspiring conclusions drawn from the experiments, and some are even counter-intuitive, e.g. LSTM models do not take much temporal information into account compared with other models; 3D Convolutions rely heavily on temporal coherency to do classification; we also find that a dataset's requirement for temporal modeling can be roughly approximated using our Sens-Index. Besides investigating existing methods, we propose a novel model to exploit temporal information for action recognition, namely Approximated Bilinear Fusion. Our model is extended from bilinear pooling, adopting low-rank approximation technique to significantly reduce trainable parameters, and can model temporal dynamics by taking multiple frames as input. We show the derivation of the low-rank approximation, and how to effectively fuse across multiple frames. Finally, we present our exploration study on our proposed model, and validate our model's performance and temporal modeling capacity on Something-Something and UCF101 datasets. The experimental results show that our model can outperform or compete with other methods, while keeping low complexity and fast convergence.

KEY WORDS: Action Recognition, Temporal Modeling, Evaluation, Bilinear Model

Contents

List of Figures

List of Tables

Nomenclature

$\otimes$	Outer Product Operator
$\odot$	Element-Wise Product Operator
$*$	Convolution Operator
$Conv_{2d}(\,)$	2D Convolutional Architecture
$Conv_{3d}(\,)$	3D Convolutional Architecture
$\tanh(\,)$	Hyperbolic Tangent Function
$Vec(\,)$	Vectorization Operation
$tr(\,)$	Trace Operation
$\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T\}$	T Input Frames
$\boldsymbol{X} = \{\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, ..., \boldsymbol{\alpha}_S; \boldsymbol{\alpha}_s \in \mathbb{R}^C\}$	S Local Descriptors with C Channels

Chapter 1 Introduction

1.1 Introduction

With the rapid development of the Internet, after seeing the domination of text and images in the last few decades, we have entered an era in which videos become the main media used for communication and information sharing, e.g. video calling in WeChat, video chat in Facebook Messenger, and YouTube (300 hours of video are uploaded to YouTube every minute![1]). Following this trend, action recognition, a fundamental and vital task in video understanding, which could also be seen as the counterpart of the classification task in image processing, has reached a new level in the research area of computer vision, thanks to the growing explosion of video data and increasing development of deep learning.

The general goal of action recognition is to enable machines to automatically analyze ongoing actions from a video consisting of multiple frames of images[2]. For a simpler case, on which our study focuses, a trimmed video contains only a single action or activity that can be described by a single annotation out of a fixed number of predefined categories, and our objective is to correctly classify this video instance into a category.

Action recognition is of vital importance in our daily life. There are various applications demanding accurate, reliable and robust action recognition systems to make our life safer, happier, and easier. Video surveillance is an important and effective way to keep our society safe. Today we can see video surveillance cameras in almost every corner in our city, e.g. university, street, bank, supermarkets, and etc. Imagine if these cameras can not only do recording, but analyzing people's behavior, e.g. recognizing abnormal activities in the crowd, then people suffer in accidents can save a lot of time reaching hospital, and we can easily locate thieves in supermarkets. Action recognition is also very useful for easier or more comfortable human-machine interactions. There are many disabled people in the world who could not handle tools in a normal way, but when equipped with action recognition systems, their intentions could be efficiently detected by their gestures or body movements, thus things could be done more easily, leading them to an easier and more comfortable lifestyle. For sports, a large quantity of recording systems have already been there for reporting and analyzing competitions for a very long time. If these hardwares can automatically accurately analyze motions and poses in the sports, there could be more efficient and accurate real-time reporting and camera switching during games, and a fairer evaluation system in the competition may emerge since the peoples' judgement may not be very consistent or reliable. Action recognition methods may also dramatically benefit autonomous driving systems. Like videos, the sensors on self-driving cars are continuously recording surrounding information, while more accurate and robust recognition algorithms are essential for both pedestrians' and drivers' safety.

In order to solve the problem of action recognition, there have been many methods proposed, from early hand-crafted methods[3-5] to recent deep learning based methods[6-10]. A detailed literature review is presented in Chapter 2. However, along with this big general goal, a more specific problem, which is also

what many of the state-of-the-art action recognition algorithms targeting at solving, is how to exploit temporal information in videos. Though many methods are proposed to capture temporal structures in videos, none of them actually quantitatively show their capacity for temporal modeling but just showing their final accuracy boosts on benchmark datasets. Although we are all interested in the final accuracy for the classification task, we want to offer a decomposed viewpoint of action recognition to analyze temporal modeling's effect in detail, in order to clarify some problems such as if temporal modeling actually accounts for the accuracy a method achieved, if there is any requirement of temporal modeling for high accuracy on a dataset, and how temporal modeling is related to action recognition on a specific dataset, etc. In order to achieve this, in this thesis we propose a method to give a quantitative evaluation on existing methods about their sensitivity to temporal coherency and how temporal coherency influences their training. Then we evaluate representative action recognition methods and discuss their results in detail, trying to summarize each model's strength on temporal modeling. Then we ask ourselves the question of how to improve the temporal modeling and increase the performance of a model. In the second half of this thesis, we propose a new model to novelly exploit temporal information by bilinear pooling with approximated implementation, which is validated on benchmark datasets. The investigation scope of this thesis mainly focuses on deep learning based models, with a few exceptions such as dense trajectory models[4, 5].

1.2 Motivation

Video action recognition is a problem closely related to image classification, but differ in a way that rather than dealing with static 2D patches, we are faced with 3D streams containing a directed time axis, leading to strong causality and correlation information in consecutive frames. Because of it, almost all the proposed methods attempt to design their algorithms and build their models to handle temporal information, explicitly or implicitly. Among recent deep learning based techniques, there are three of them widely used for temporal modeling[11-13]:

- **Optical Flow** is a modality which highlights the movements between consecutive RGB frames and was shown to be very powerful for action recognition when it was introduced along with two-stream architecture by Simonyan and Zisserman[7]. In their work, a stack of optical flow fields were fed into a 2D convolution neural network to produce a prediction, and further fused with a score from RGB branch. This method inspired many subsequent works using optical flow as an explicit way to handle temporal information[14-19].
- **Recurrent Neural Network** and its variants are well-known models dealing with sequential information in natural language processing, and it is very natural to incorporate them into video modeling since videos are sequential information of images. The recurrent architectures are usually applied to a sequence of image features extracted by a deep convolutional neural network[8, 18, 20-22]. There are also convolutional based RNNs which use convolution as transition operators in the hidden units[23-25] which are supposed to capture both spatial and temporal information recurrently.
- **3D Convolutions** resemble 2D convolutions in image processing, but use an additional dimension in filters to convolve along time axis. The hierarchical structures in 3D convolutions are supposed

to capture spatio-temporal patterns implicitly. 3D convolutions are introduced to handle action recognition by Ji et.al. in[26], and a more representative model by Tran et.al. in[9]. Then a bunch of investigation on 3D convolutional structures have been conducted[12, 27-31] and some architectures are very promising recently[12, 13, 18, 31].

Although these techniques were introduced to handle temporal information in videos and their performance on benchmark datasets like UCF101[32] has more or less shown their strength in temporal modeling, there are still some problems discovered recently[33, 34] which also motivate this thesis to do a further investigation on them. We highlight some of the problems in the following paragraphs:

- During the development of action recognition algorithms, except for the approaches mentioned above, there are some other factors that dramatically influence the performance of recognition accuracy, e.g. backbone 2D CNN architectures, and different data augmentations techniques. These techniques do not take any temporal information into account, but are very influential in accuracy and has been increasingly adopted in many proposed models, leading us to think about how temporal modeling is essential for good action recognition. Are the performance boosts are just caused by the advanced CNN architectures or actually the novel approaches for temporal modeling?
- There has been an investigation on optical flow in[33] by Sevilla-Lara et.al. in which the authors argue that the value of optical flow is its invariance to appearance of input images instead of motion information it contains. They have validated it by shuffling the optical flow fields, image frames, and altering the input color ranges. The results indicate that optical flow is not drastically influenced by the collapse in temporal coherency, and the flow net is much more robust than the RGB net about color shift. This investigation revealed some potential benefits and drawbacks of optical flow which we haven't considered earlier, and motivated us to do a further investigation on more methods to see their value in temporal modeling. We are also interested in the vanilla performance of optical flow without any fancy testing schemes like "10-crop 25-segment" to see how it performs, which hasn't been shown in[33].
- In[34], Zhou et.al. investigated the performance change of their model on UCF101[32] and Something-Something[22] datasets with shuffled and ordered frames. The results show that Something-Something is a dataset requiring strong temporal modeling to get high action recognition accuracy while UCF101 doesn't. This also indicates that the methods working well on temporal modeling do not guarantee good performance on UCF101, and vice versa. This stimulates our curiosity about how well other methods perform on Something-Something dataset to see if they can also actually capture temporal information.

These phenomena and unsolved questions implicitly present some potential flaws in the existent approaches, and the way to model temporal dynamics in action recognition is far from ending. Questioning these problems, in this thesis we do a thorough investigation on existent action recognition approaches focusing on their capability to model temporal information. We hope this work can help future researchers to design new models to handle temporal dynamics, and propose novel ways to think about action recognition problems. Besides investigating existent methods, we propose a new model using a new way to exploit

temporal information in the second half of this thesis.

1.3 Challenges

Action recognition is a problem far from solved in computer vision, and how to exploit temporal information is a question frequently asked in literatures[7, 15, 18, 35-37]. In this section, we present some challenges of temporal modeling in action recognition, and we introduce the difficulties in quantitatively measuring a model's capability on temporal modeling.

Thought it is very intuitive to think about exploiting temporal information in action recognition problems since the only difference between video classification and image classification is the length of the input data, e.g. the former consists of sequential frames while the latter doesn't, it is not intuitive to come up with the best way to handle it. We present some difficulties here.

- A key aspect of temporal information is *motion*, and motions in a video usually characterize the type of action. For human, noticing the motion by our visual system is very natural and we are born to be good at it. However, the input data for a computer vision model is usually consecutive of RGB frames whose quality heavily depends on the cameras, and in most cases, camera motion can dominate the (human) motions in a video. This may cause that temporal information in a video is not informative enough to be a strong cue for action recognition.
- We are not sure how to obtain the temporal information. There are multiple ways proposed to do sampling along time to get a fix length of input frames for processing, e.g. consecutive frames as a clip[9], or seperated segments[10]. None of them serve as a main stream, and we are not even sure why fixed length of temporal information is necessary since for human we can accept various length of videos as input. Generally speaking, accessing dynamic temporal information is not as intuitive as accessing static 2D image information.
- Another important point that obstructs the development of temporal modeling in videos may come from the benchmark datasets. For the last 5 years, the most popular datasets in video action recognition UCF101[38], basically resemble the image classification datasets that provide only a coarse annotation of a video instance based on its "appearance", making action classification rely heavily on scene classification. This results in a phenomenon that using very small number of frames without temporal modeling (single frame classification or average among individual frames) can compete or even outperform models with explicit temporal modeling. A bad consequence of this is the feeling of unnecessariness of modeling of time. Fortunately, some most recent datasets try to alleviate this problem by providing finer annotations on time[22] or increasing the intra-class variations[39] to weaken the dependence on appearance.

In order to quantitatively show how the methods model the temporal dynamics, we need a general protocol which could be used to evaluate a model's capacity in exploiting temporal information. To the best of our knowledge, there is no such a protocol introduced previously that can be applied to recent deep learning based methods. Therefore in this thesis, we are going to propose a protocol to focus on measuring temporal modeling. We believe this work is beneficial for new methods' proposal. Here we introduce the challenges

of this work.

- There are very few parameters shared among existing methods for us to monitor. Typically when a model was first introduced, the only goal of it was to reach as high accuracy as possible, therefore no internal parameters are shared among different approaches but the last layer which usually contains scores for each class. It is not easy to employ these limited scores to give a usable, generic, and quantitative evaluation for each distinct method.
- Temporal modeling sometimes is not independently testable because some methods do not explicitly consider it but still have the capacity to capture it in an implicit way. For example, 3D convolution can capture temporal structure but since there are temporal pooling operations and the receptive field of units in deep layers increases, the temporal information may become not clear and hard to monitor.
- A fair comparison is hard because different methods have their own optimal hyperparameters and some are even dynamic, e.g. the number of frames to use for each example may depend on the input image size and the depth of backbone CNN architecture. Therefore the goal of our work is not to find the optimal parameter for each method, but to give a rough evaluation about each selected method's capacity on modeling temporal information.
- Since we need to train and test many models, while the time and resources are limited, it is quite challenging to complete all these works in such a short time.

However, despite all these challenges, we make all-out of effort to give a thorough literature review on action recognition, propose protocols on temporal modeling, evaluate the representative approaches, propose a novel method for better temporal modeling, and validate it on two datasets.

1.4 Contributions

The main contributions of this thesis are fivefold. We summarize them as follows:

- We do a literature review on recent deep learning based action recognition models, focusing on presenting a survey on how recent methods exploit temporal information in action recognition. Reviewed methods are categorized into Fusion Methods, Motion Representations, Temporal Convolutions, and Other methods, which are based on their main contributions on temporal modeling. The literature review is presented in Chapter 2.
- We propose new protocols to quantitatively evaluate a model's sensitivity about temporal coherency, and how a model's training is influenced by temporal coherency. This evaluation method works with models that are trained using CrossEntropy Loss.
- We selecte seven representative methods from our review and evaluated them using our proposed protocols on UCF101 and Something-Something datasets. Using our protocols, we have found some interesting points which haven't been emphasized in the previous literatures. For instance, LSTM models exploit no temporal information on UCF101 datasets, while Temporal Relation and 3D Convolution models do; 3D Convolution seems to be less robust to temporal coherency; optical flow can exploit temporal information to achieve advanced performance even the flow fields are shuffled. Experiments also show that our proposed Sens-Index could be used to quantitatively describe a

dataset's requirement for temporal modeling to achieve good performance.

- We introduce a novel temporal modeling method called approximated bilinear fusion model for action recognition. This model uses low-rank approximation to approximate the bilinear pooling, and could efficiently fuse multiple frames while taking second order temporal information into account. We show the derivation of the approximation, and the detailed architecture of our model in Chapter 5.
- We conduct experiments to do exploration study on our proposed model. Our model is evaluated using our proposed protocol to confirm its temporal modeling capability. The model is validated on UCF101 and Something-Something datasets, and the results show our model can outperform all reviewed methods except the 3D Convolution model (which is pretrained on Kinetics and not directly comparable). We also show that our model contains the lowest number of parameters to be learned among the compared models. The experiments proved the effectivity and efficiency of our approach.

1.5 Thesis Structure

We organize the remaining parts of this thesis as follows:

- **Chapter 2** provides an overall literature review on deep learning based action recognition approaches. The reviewed methods include novel architectures, techniques to fuse multiple frame features, and representations used to capture temporal information in video action recognition. We have divided the surveyed methods into four categories by their main contributions on temporal modeling: Fusion Methods, Motion Representations, Temporal Convolutions and Other Methods.
- **Chapter 3** proposes several temporal evaluation protocols to quantitatively measure how much temporal coherency influences the learning of a model, and how sensitive a trained model is about temporal information. We adopt accuracy and prediction vector as our parameters to monitor. Root-Mean-Square deviation is used on prediction vectors to produce the *Inf* index and *Sens* index.
- **Chapter 4** evaluates representative methods selected from Chapter 2 using the protocols proposed in Chapter 3. This chapter introduce the used datasets, experimental details, experimental results, and conclusions drawn from the experiments.
- **Chapter 5** introduces our proposed Approximated Bilinear Fusion model. We show the bilinear pooling can be efficiently implemented by inner-products of attention maps produced by convolution with 1×1 kernels, through low rank approximation of the high dimensional parameters, and we extend the model to accept multiple frames to capture long term temporal dynamics.
- **Chapter 6** presents the exploration study of our model on Something-Something dataset, and evaluate our model using the protocols we proposed. We show that our model can definitely benefit from temporal modeling, and outperform all 2D convolution based models, and compete with 3D convolution model, while containing much fewer parameters and converging much faster than others, indicating its advanced effectivity and efficiency.
- **Chapter 7** summarizes and concludes the thesis. In this chapter, we highlight our contributions of each chapter, and introduce potential future directions of our work.

Chapter 2 Literature Review

2.1 Introduction

There have been tremendous works trying to solve the problem of action recognition, and various types of temporal modeling methods have been proposed recently. In this chapter, we briefly survey existent methods by focusing on investigating their efforts on temporal modeling in videos. Because of the limitation of time, our review focused on the recently published methods based on the most promising deep learning techniques, with an exception of the most successful handcrafted method IDT[4, 5]. We categorize the reviewed methods into four subsections by their main contributions in handling temporal dynamics:

- **Fusion Methods** include models that rely on 2D convolutions performing on multiple frames to extract a sequence of high level features of a video, and some special fusion techniques are used to produce a video level prediction, e.g. average pooling, max pooling, and LSTMs.
- **Motion Representations** include representations that are proposed to be useful to capture temporal information. These representations usually summarize consecutive frames to delicately encode their motion information, e.g. optical flow and dense trajectory.
- **Temporal Convolutions** are designed to retrace the success of 2D convolutions in image processing. These methods include 3D convolutions and some alternative architectures (mixed 2D and 1D convolutions) which capture temporal dynamics by convolution operation.
- **Other Methods** include some other techniques which could not be summarized by the other three main categories but also try to exploit temporal information in videos such as explicitly modeling state transformation in videos and temporal coherency.

Before we dive into the literature review, we define a few basic annotations here which are generally used among all methods. We define $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T\}$ as the representation of the input frames (a video or a clip) to a model, and $\boldsymbol{y}$ as the prediction made by this model on the input video. $Conv_{2d}(\ )$ denotes a 2D convolutional architecture outputting high-level feature maps for a frame. We also use f as a layer (or multiple layers) above the base convolution architecture which could be seen as a classifier, and its meaning may vary for different algorithms.

2.2 Fusion Methods

In this section we review methods which usually rely on pretrained 2D convolutional networks. Each frame is passed through this pretrained model and a sequence of feature (maps) are extracted for further processing, and finally a video level prediction will be produced.

2.2.1 Single Frame

We consider this method as the most basic implementation as it downscalse the video action recognition back to image classification, which can be formulated as:

$$\boldsymbol{y} = f(Conv_{2d}(\boldsymbol{x}_i)) \quad i \sim [1, T] \tag{2–1}$$

This method has been adopted in[6, 20] as a baseline model. Surprisingly, in[6] this model didn't show large inability compared with its multi-frame counterparts. This may be caused by the lack of temporal modeling in compared methods, or the benchmark datasets didn't show much requirement for multi-frame consensus.

2.2.2 Average Pooling

In[8], Donahue et.al. utilized an average pooling along time axis as a baseline model to compare with their proposed LRCN counterpart. Wang[10] experimentally showed that using averaging consensus function across temporal segments performs better than max and weighted average encoding. This basic fusion method also played as a baseline in[22], in which the proposed dataset Something-Something requires very careful modeling in temporal dimension to achieve good results, which is not what average pooling good at.

$$\boldsymbol{y} = f(\frac{1}{T}\sum_{i \in T} Conv_{2d}(\boldsymbol{x}_i)) \tag{2–2}$$

Intuitively, this fusion method do not exploit temporal information because averaging consider no order of the frames.

2.2.3 Max Pooling

Ng et.al. introduced several pooling methods along time in[20], and they showed that using max-pooling over the final convolutional layer across the video's frames outperform other compared pooling methods:

$$\boldsymbol{y} = f(MAX(Conv_{2d}(\boldsymbol{x}_1), ..., Conv_{2d}(\boldsymbol{x}_T))) \tag{2–3}$$

Same as average pooling, this method also take no temporal information into account. But it should show better performance than single frame model because it can ignore less informative frames.

2.2.4 Element-Wise Multiplication

Diba et.al.[40] proposed temporal linear encoding to learn a representation from entire videos. They adopted element-wise multiplication to fuse features from multiple frames, and the fused feature is passed through a bilinear encoding to learn interaction across spatial locations. The element-wise multiplication fusion could be formulated as:

$$\boldsymbol{w} = Conv_{2d}(\boldsymbol{x}_1) \odot ... \odot Conv_{2d}(\boldsymbol{x}_T) \tag{2–4}$$

$$\boldsymbol{y} = f(\boldsymbol{w}) \tag{2–5}$$

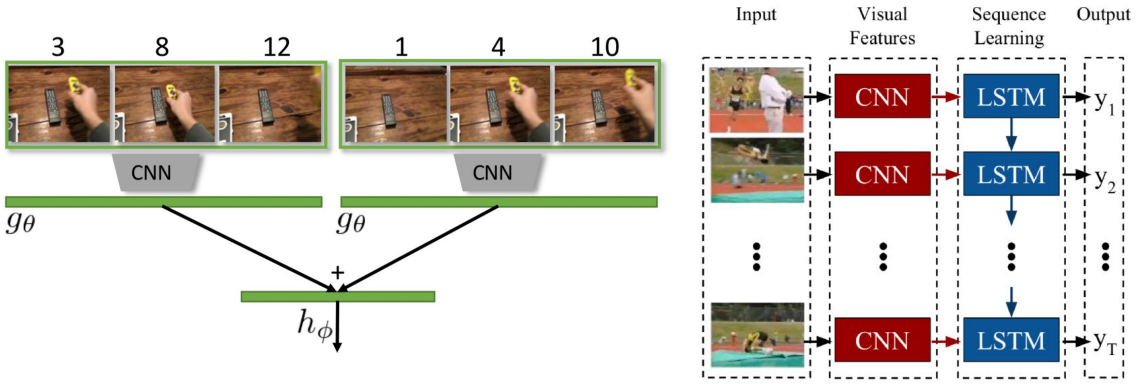

Figure 2–1 The architecture of Temporal Relation Network and Long-term Recurrent Convolutional Network. On the left is TRN which fuses multiple features of frames using multi-layer perceptrons across frames. On the right is the LRCN which utilizes LSTMs to fuse multiple frames. Figures are taken from[34] and[8].

where $\odot$ denotes element-wise product. Their proposed bilinear encoding is:

$$\boldsymbol{y} = f(\boldsymbol{w} \otimes \boldsymbol{w}) \tag{2–6}$$

where $\otimes$ denotes outer product. Note that the bilinear encoding in this method is not used to capture temporal information while element-wise multiplication is.

2.2.5 Bilinear Encoding

Wang et.al.[17] proposed to use compact bilinear encoding as a general fusion technique. Their proposed Spatio-temporal compact bilinear (STCB) performed roles of fusing snippets of optical flow features, producing attention maps, and fusing all feature pathways. This could be formulated as:

$$\boldsymbol{v}_i = Conv(\boldsymbol{x}_i) \tag{2–7}$$

$$\boldsymbol{v}'_i = CountSketch(\boldsymbol{v}_i) \quad i \in T \tag{2–8}$$

$$\boldsymbol{y} = f(FFT^{-1}(\odot_{i=1}^{T} FFT(\boldsymbol{v}'_i)) \tag{2–9}$$

where the $CountSketch()$ means Count Sketch projection function[41], and FFT denotes fast Fourier transform. Note that in the original paper, the inputs were optical flows rather than RGB frames to avoid overfitting issue. It turned out that compact bilinear encoding outperforms other basic fusion methods (e.g. average, concatenation, element-wise sum) by a large margin.

2.2.6 Temporal Relation

Zhou et.al.[34] introduced a temporal reasoning module which learns representations to summarize multiple samples of sorted frames, and could be extended to capture multi-scale temporal relation.

$$\boldsymbol{v}_i = f(Conv(\boldsymbol{x}_i)) \quad i \in T \tag{2–10}$$

$$\boldsymbol{y} = h(\sum_{i<j} g(\boldsymbol{v}_i, \boldsymbol{v}_j)) \quad i, j \in T \tag{2–11}$$

The results confirmed its advantage on datasets such as Something-Something[22], Jester[42], Charades[43] and Moments in Time[39], but not on UCF101[32]. This may be due to that recognizing activities in UCF101 does not necessarily require much temporal relational reasoning. The illustration is shown in Figure 2–1 left.

2.2.7 RNN

Recurrent neural networks are usually applied to the CNN extracted features (or feature maps) to capture temporal structures in videos. Donahue et.al. introduced a long-term recurrent convolutional network[8] to exploit temporal information by recurrently processing the extracted feature vectors by CNNs from each frame of a video clip. The temporal features are extracted by LSTMs and averaged along each time step (see Figure 2–1 right for an illustration). Ng et.al.[20] proposed to apply multiple layers of LSTMs to CNN features of each frame to capture long-term temporal structure, but the gain was limited. Srivastava et.al.[44] proposed to use LSTMs as a pretraining method by encouraging the model to reconstruct input frames and predict future frames. They showed that this pretraining can boost performance on action recognition. In[21], Sharma et.al. introduced a three-layer LSTM architecture that can produce soft attention maps for next-frame CNN feature cubes, which is supposed to more effectively exploit temporal structure. However, only limited gain has been achieved. The formulation for generic LSTM model could be summarized below:

$$\boldsymbol{v}_t = f(Conv(\boldsymbol{x}_t)) \tag{2–12}$$

$$\boldsymbol{i}_t = \sigma(W_{vi}\boldsymbol{v}_t + W_{hi}\boldsymbol{h}_{t-1} + b_i) \tag{2–13}$$

$$\boldsymbol{f}_t = \sigma(W_{vf}\boldsymbol{v}_t + W_{hf}\boldsymbol{h}_{t-1} + b_f) \tag{2–14}$$

$$\boldsymbol{o}_t = \sigma(W_{vo}\boldsymbol{v}_t + W_{ho}\boldsymbol{h}_{t-1} + b_o) \tag{2–15}$$

$$\boldsymbol{g}_t = \tanh(W_{vg}\boldsymbol{v}_t + W_{hg}\boldsymbol{h}_{t-1} + b_g) \tag{2–16}$$

$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \boldsymbol{g}_t \tag{2–17}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t) \quad t \in T \tag{2–18}$$

where $\boldsymbol{v}$ denotes the input to the LSTM architecture, and should be vector representations. tanh denotes the hyperbolic tangent function. In[22], Goyal et.al. showed that using LSTMs to process CNN extracted features was shown to perform better than average pooling on Something-Something dataset. Resembling[10], Ma et.al.[18] proposed to apply LSTMs on temporally segmented videos to sequentially process pooled features for each segment. Du et.al.[36] introduced a more sophisticated RNN based architecture which can learn a summary of spatial-temporal encoding at each time step about the whole context and apply it to the hidden

states. In[45], Zheng et.al. adopted the Simple Recurrent Unit[46] as the hidden units and trained their model with three independent streams containing multiple levels' of convolutional features to boost the performance by fusion.

2.2.8 Convolution Based RNN

Unlike usual recurrent unit implementations with vectors as hidden states, Shi et.al.[23] introduced convolutional LSTMs to use convolution operation as hidden transition function, and this implementation can jointly model spatial and temporal structure. The basic idea behind convolution based LSTMs is to replace the linear transformation in regular LSTMs by convolution operations. Taking input gate as an example, the convolutional version could be formulated as:

$$\boldsymbol{v}_t = Conv(\boldsymbol{x}_t) \tag{2–19}$$

$$\boldsymbol{i}_t = \sigma(W_{vi} * \boldsymbol{v}_t + W_{hi} * \boldsymbol{h}_{t-1} + b_i) \tag{2–20}$$

where $\boldsymbol{v}$ denotes feature maps produced by a ConNet and $*$ is convolution operation. Similarly, Ballas et.al.[24] proposed GRU-RCN between states as well, but they leverage Gated Recurrent Units[47] instead of LSTMs. GRU-RCNs also exploit spatio-temporal information by constructing recurrent convolutional neural networks at different levels of abstraction of input frames. Li et.al.[37] also proposed convolution based LSTMs with attention mechanism which convolve to generate gates and attention maps. They also incorporated optical flow modality to obtain the hidden states so as to further model temporal structure.

Sun et.al.[25] introduced Lattice-LSTM to address the issue in typical convolution based LSTMs that hidden state transitions of memory cells are shared across spatial locations thus lack capability to characterize complex motion patterns. Their local superposition summation performs local filtering operations with different filters at different spatial locations which dramatically enhances the representation of motion dynamics. The local superposition summation only operates on the hidden transition of cell memory within the recurrence:

$$\boldsymbol{g}_t = \tanh(W_{vg} * \boldsymbol{v}_t + W_{hg} \textcircled{\Omega} \boldsymbol{h}_{t-1} + b_g) \tag{2–21}$$

$$W_{hg} \textcircled{\Omega} \boldsymbol{h}_{t-1} = \forall_{k,i,j}\Big(\sum_{d,m,n} W_{hg}(d,i,j,k,m,n)\boldsymbol{h}_{t-1}(d, i+m-1, j+n-1)\Big) \tag{2–22}$$

where (i, j) indicates the position in the feature map, m and n index position in the kernel, and d and k index the channels of input and output feature maps.

2.3 Motion Representations

2.3.1 Optical Flow

Many methods utilized optical flow modality to exploit motion information[7, 14-19, 25]. Simonyan and Zisserman first introduced two-stream network in[7] to do action recognition. They use optical flow modality to explicitly bring motion information into 2D deep neural networks. Their deep model first outperformed

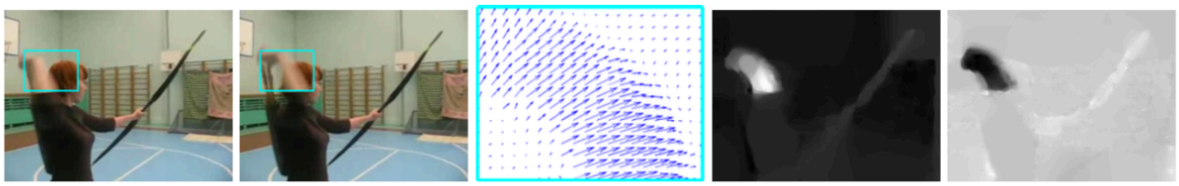

Figure 2–2 (a)(b) are two consecutive frames. (c) represents the dense sampled optical flow in the highlighted area. (d)(e) denote the displacement on x and y directions to visually represent the optical flow. Positive areas mean strong movements. Figures are taken from[7].

strong handcrafted method[5] and other deep models by a large margin. The used optical flow representation is shown in Figure 2–2. Feichtenhofer et.al. sought alternative fusion schemes of flow and RGB modalities in[14]. They proposed to fuse two streams at last convolution layer as well as the softmax layer. They found that for spatial fusion, bilinear fusion and conv fusion are more effective, and for temporal fusion, 3D conv plus 3D pooling performed best. Later in[15] and[16], Feichtenhofer introduced residual additional and multiplicative interaction between two streams to better fuse spatialtemporal features. In[16], temporal convolutional layers are also employed to provide two-stream nets with better temporal support.

2.3.2 Estimated Optical Flow

Because of the expensive computational overhead to calculate the optical flow, some methods sought to estimate optical flow using RGB inputs. In[48], Ng et.al. introduced a multi-task model, ActionFlowNet, to estimate optical flow and predict video labels simultaneously. This method significantly outperformed other methods that are not pretrained on external large datasets, and even outperformed C3D[9] which is pretrained on Sports-1M dataset[6].

2.3.3 RGB Difference and Warped Optical Flow

In[10], Wang et.al. studied two extra input modalities of RGB difference and warped optical flow to see if they can boost the recognition performance. The warped flow was inspired by the work of improved dense trajectory to compensate camera motion. The results showed that warped flow can be combined with original two modalities to further improve accuracy.

2.3.4 Optical Flow Guided Feature

Sun et.al.[49] introduced a novel compact motion representation, optical flow guided feature (OFF), to exploit temporal information. This feature is derived from the definition of optical flow but in an orthogonal space. OFF is obtained by applying Sobel operator and element-wise subtraction on two consecutive frames. OFF is complementary to optical flow and RGB inputs and thus could be combined to further boost performance.

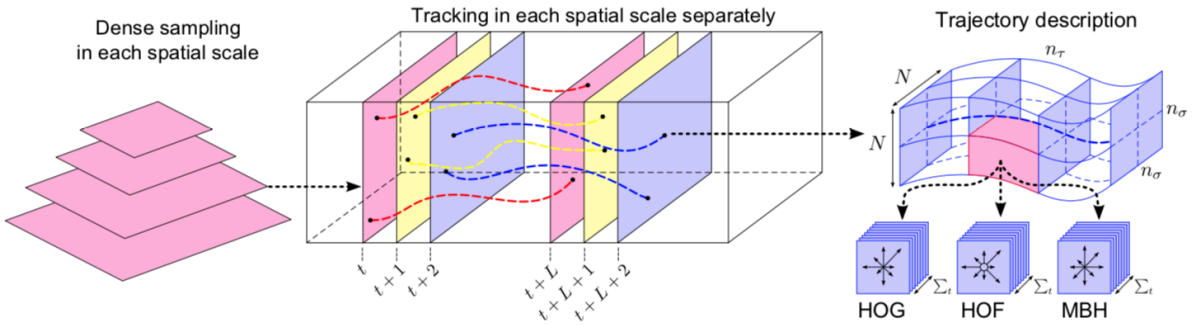

Figure 2–3 This figure illustrates how the dense trajectory features are extracted. First multi-scale feature points are densely sampled. Second, trajectories scaling N frames are tracked and delivered to further usage. Third, local descriptors (HOG, HOF, MBH) are extracted from the volumes based on the tracked trajectories. Figures are taken from[4].

2.3.5 Dense Trajectory

In[4], Wang et.al. introduced the dense trajectory representation for action recognition. The model captures temporal information and local appearance feature along extracted trajectories (volumnes) with hand crafted descriptors (HOG, HOF, MBH). The procedure to extract dense trajectory features is in Figure 2–3. Later, Wang et.al.[5] improved the model by explicitly estimating camera motion, while they also removed inconsistent matches cause by human motion during estimation using a human detector.

2.4 Temporal Convolutions

Vanilla Convolution Inspired by the success of CNNs in image classification, Ji et.al.[26] proposed to apply 3D convolution to action recognition thus temporal information could be implicitly captured by convolution operation. As a early work, its temporal span is limited to 7 frames. Karpathy also used 3D convolution in his slow-fusion method[6], with a clip of 10 frames. A more representative 3D convolution based action recognition model is the later proposed C3D by Tran et.al.[9]. This model is supposed to extract generic and compact deep feature in action recognition, and experiments confirm that $3 \times 3 \times 3$ kernels work best. The temporal span has been extended to 16 frames per clip compare with[26]. In order to handle long-term dependency with 3D convolution networks, Varol et.al.[30] proposed to reduce the input spatial resolution and thus make it feasible to extend the temporal dimension beyond limited snippets (from 16 frames to 100 frames). Unlike other prior 3D convolution methods,[30] also conducted temporal convolution on optical flow modality for further temporal gain. The general 3D convolution (see Figure 2–4 left) on a clip could be summarized as:

$$\boldsymbol{y} = f(Conv_{3d}(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T)) \tag{2–23}$$

More recently, Carreira et.al.[31] proposed Inflated 3D ConvNet (I3D) which can benefit 3D convolution from ImageNet pretraining. They also compared the 3D convolution on RGB and optical flow, and the results show that when the dataset is large enough (e.g. Kinetics[50]), the performance of RGB inputs can surpass the

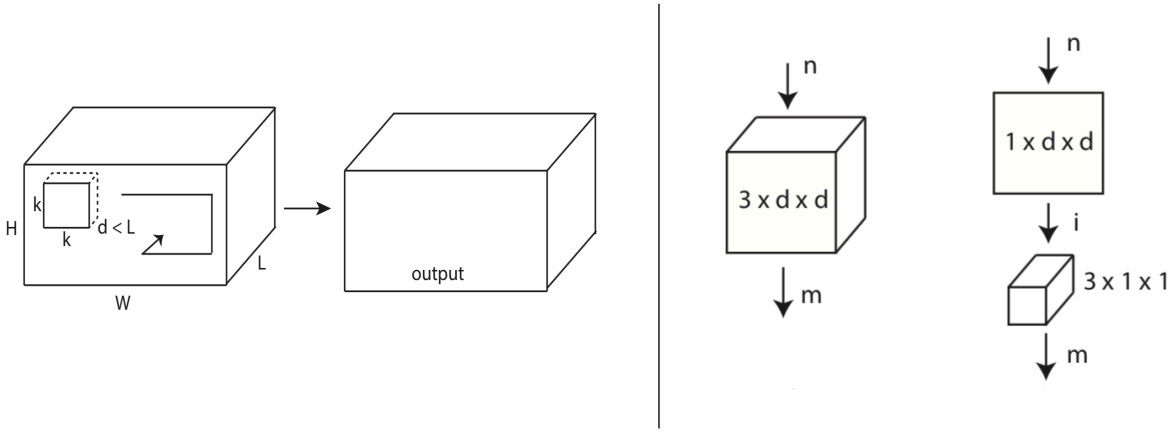

Figure 2–4 Left figure shows the 3D convolution operation on a group of feature maps. Right figure represents the replacement of 3D convolution by 2.5D convolution. Figures are taken from[9] and[29].

flow modality, which suggests the power of discovering temporal structure by 3D convolution on raw pixels. Hara[12] confirmed that very deep 3D CNNs (up to ResNet152[51]) can be trained from scratch on Kinetics dataset[50] without overfitting. In[52], Diba et.al. augmented the DenseNet[53] to 3D convolution, and proposed a new module called Temporal Transition Layer which concatenates feature maps produced by kernels with different temporal depths.

2.4.1 Modified Convolution

Considering the high complexity of 3D convolution, Sun et.al. introduced factorized 3D convolution in[27] which simulates a 3D convolution by separated spatial and temporal convolutions resulting in a reduction of kernel complexity. Specifically, temporal convolution is a 2D convolution over time and feature channels at high levels of the network. In[28], Tran et.al. also sought cheaper alternatives of 3D convolutions along temporal modeling, namely Mixed 3D-2D ConvNets (MCs) which replace higher level convolutions with 2D convolutions, and 2.5D ConvNets which replace a 3D convolution with a 2D followed by a 1D convolution (see Figure 2–4 right). But their results didn't show any advantage over 3D convolution. Later in[29], Tran et.al. showed that replacing the 3D convolutions with 2.5D convolutions can dramatically boost performance on Kinetics and Sports-1M with ResNet backbone. Xie et.al.[13] introduced a promising Top-heavy-S3D model which successfully replaced bottom few levels of heavy 3D convolutions with lighter 2D, and factorized top few 3D convolutions into 2D and 1D convolutions, which could be written as:

$$\boldsymbol{y} = f(Conv_{2.5d}(Conv_{2d}(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T))) \tag{2–24}$$

where $Conv_{2.5}$ denotes a 2D convolution followed by a 1D to replace the heavy 3D counterpart. This model not only reduced parameters to learn, but also achieved higher performance compared with its I3D counterpart due to its less likelihood of overfitting. They also proposed an attention mechanism by gating with spatial-temporal pooling to further exploit temporal information. Ma et.al.[18] introduced an Inception-like

convolutional module using multi-size kernels to do temporal convolving, but only a limited gain has been achieved.

2.4.2 Multiplicative Interaction

Instead of using plain convolution with linear transformation to learn the relation between frames, Wang et.al.[54] appended a square operation after the 3D convolution to estimate the multiplicative interaction between frames. This relation branch along with a normal 2D convolution branch formed a generic building block, SMART block, which has been shown more effective than plain 3D convolutions of the same parameters.

2.5 Other Methods

2.5.1 Attention

In[36], Du et.al. not only use an LSTM to model the temporal information, but introduced a temporal attention mechanism to further discover complex temporal features. Their model takes the whole context information into account when processing feature vector at each time step and output an intergraded representation after focusing on spatial and temporal salient parts. Girdhar and Ramanan[55] introduced bottom-up and top-down attention heatmaps to better capture fine-grained spatial information, but their temporal modeling relied on the consensus provided by temporal segment network[10].

2.5.2 Temporal Consistency

Some methods sought to explicitly model the temporal consistency in videos. In[56], Misra et.al. trained a Siamese Network on frame tuples to tell if the given frames are in natural order. This unsupervised pretraining can be used as an initialization for action recognition. As shown in paper, the pretrained network outperformed the randomly initialized baseline by a large margin.

2.5.3 Transformation

Wang et.al.[57] proposed to model the motion in a video as a transformation between preconditions and effects. They modeled the transformation as a matrix multiplication, and preconditions and effects are deep representations of early frames and late frames. The precondition and effect representations are:

$$\boldsymbol{F}_{pre} = f(\frac{1}{P}\sum_{i=1}^{P} Conv_{2d}(\boldsymbol{x}_i)) \tag{2–25}$$

$$\boldsymbol{F}_{eff} = f(\frac{1}{E}\sum_{i=T-E+1}^{T} Conv_{2d}(\boldsymbol{x}_i)) \tag{2–26}$$

Because their networks do not directly output classification scores, their training objective is to minimize the distance between the effect and the transformed precondition caused by the actual action, and maximize

that caused by other actions:

$$\min\ (D(\boldsymbol{A}_y(\boldsymbol{F}_{pre}), \boldsymbol{F}_{eff}) + \sum_{i \neq y}^{N} \max(0, m - D(\boldsymbol{A}_i(\boldsymbol{F}_{pre}), \boldsymbol{F}_{eff}))) \tag{2–27}$$

where $D()$ denotes a distance function, and m is the predefined margin value. $\{\boldsymbol{A}_1, \boldsymbol{A}_2, ..., \boldsymbol{A}_N\}$ are actions of N classes, and $\boldsymbol{A}_y$ is the target action. The action recognition is equivalent to identify which transformation is most suitable to relate the precondition and effect.

2.5.4 Position Codebook

In[35], Duta et.al. proposed a novel encoding method to incorporate CNN features over entire videos. They constructed a position codebook to encode spatial and temporal information in their final representation. Besides, optical flow and 3D convolutional features were also combined to boost performance.

Chapter 3 Proposed Temporal Evaluation Protocol

3.1 Introduction

In most of the papers where the models were introduced, improved performance on specific benchmark datasets can be seen, and it usually seems natural to attribute those advanced performance to the proposed new techniques that handle temporal information. But there is no such a convincing clue that can show if it is actually the temporal modeling accounts for the improvement. We see this as a potential problem that prevents us from exploring more effective temporal modeling methods, since the effectivity of existing methods on temporal modeling has been concealed. A very important reason of this is the lack of a general protocol that can be used to evaluate temporal modeling. The absence of such a evaluation method leads to a situation that, when a new action recognition approach is introduced, which usually accompanied with some new state-of-the-art record on some benchmark datasets, we can rarely tell if those improvements are achieved by more advanced 2D backbone architectures, or its increased capability to understand the temporal dynamics along time axis. Or to an extreme, we are asking if those models actually take any temporal information into account. In this chapter, we are going to present several protocols to focus on measuring a model's capability to exploit temporal information.

Before we start introducing the proposed methods, it's important to know why designing a usable protocol on this problem is challenging. Here we list three main points that make this problem so hard that there is still no widely used methods at present.

- First of all, in order to measure a property of models, we need to monitor some specific variables which should be semantically shared among different methods. But the fact is that the existing methods vary so drastically that there are not many stable variables we could rely on to quantitatively monitor the performance on temporal modeling.
- Second, temporal information can be jointly entangled with other cues (e.g. spatial information) for some methods which makes only monitoring shifts along time axis difficult or even not possible. A worse news is that for some models only fixing or modifying the temporal cue may result in collapse of the whole model leading to the valuelessness of the measurement.
- Third and also the most important is that, it is hard to tell if a model's awareness of temporal information is good or bad because unawareness of time sometimes could be seen as a good property since it indicates a level of robustness, but may also be bad because this model may be lack of reasoning capacity along time. Because of these, people may doubt the value of temporal modeling.

Though these challenges are still there and hard to solve, we try our best to introduce the following measurements to quantitatively describe a method's efforts on temporal modeling.

3.2 Accuracy

As the only shared and evaluable thing across different methods is the final prediction, it is straightforward to compare how the prediction accuracy changes on benchmark datasets when the temporal coherency is preserved or not for each model. Because achieving high accuracy is the only final goal of the recognition task, using it as a criteria could reveal if temporal exploiting is correlated to or necessary for this goal, e.g. if action recognition on a dataset truly require temporal modeling to achieve high accuracy. Previously, this measurement has been partially reported in[13, 33], and has been interpreted as "seeing the arrow of time" in[58]. In[33], Sevilla et.al. evaluated the influence of temporal coherency on optical flow modality. But both of them only reported the accuracy shift of the models when tested under different temporal conditions, but did not train them on abnormal sets to see how well a model adapt to the incoherent situation. Therefore we propose to not only evaluate the accuracy difference when testing the same model under multiple temporal conditions (like normal, reversed, and shuffled frames), but train a model under different temporal conditions. We believe this additional step can give a fairer comparison of how a model's accuracy rely on temporal coherency of a dataset. Considering the above papers did not compare multiple methods together, in this thesis, we more thoroughly evaluate the accuracy shift in a broader range of methods.

Specifically, for each method we train two versions of it on the normal training set and the shuffled training set (with input frames shuffled along time axis), respectively. Then we report the test accuracy on the test set with normal, reversed and shuffled frames for each version. Thus there should be 6 accuracy results for a method on a single dataset. In addition, we train these methods on both UCF101[32] and Something-Something[22] datasets, resulting 12 entries of reported accuracy for each method. Besides the single accuracy for a model, we report the top 10 class-level results that the accuracy are improved or dropped in some of the detailed discussions. In this way, we can more easily observe which classes need more time modeling to be recognized.

3.3 Distance Between Prediction Vectors

A drawback of measuring by accuracy is that the single final prediction score may conceal how the models respond to each instance since the accuracy is a high-level summary of each specific responses. Despite that the existent methods vary dramatically, most of them produce a vector at the final layer as a prediction of probabilities for each class of this instance. This is a more detailed parameter that we can monitor than the final prediction, and could be seen as a semantics representation of an instance. In this case, we can calculate the distance between two corresponding prediction vectors to indirectly monitor changes. Note that although almost all methods output a prediction vector containing scores for each class, we should only compare the models trained with the same loss function such as methods all trained with CrossEntropy Loss after softmax activation. Otherwise the semantics between compared vectors are different, and are not directly comparable. Therefore models like IDT[5], which is trained with a Hinge Loss using SVM, should not be compared with deep methods. In this section, we introduce two measurements to evaluate models' temporal exploitation.

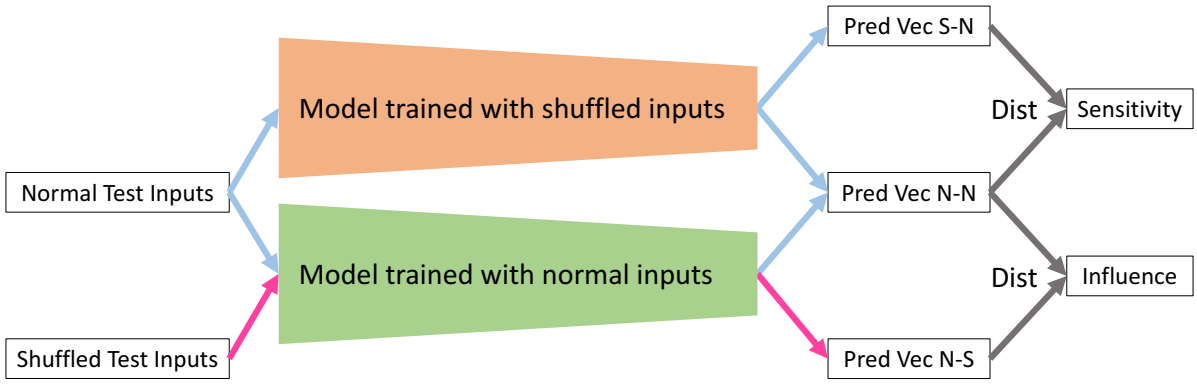

Figure 3–1 The illustration of the proposed measurement on the influence of time and models' sensitivity about time. This figure shows how the two scores are calculated. For sensitivity, we calculate the distance between prediction vectors of two models (shuffled-trained and normal-trained) on the same normal test inputs. For influence, we calculate the distance between prediction vectors on two test inputs (normal and shuffled) by a single normally trained model.

3.3.1 Influence of Time

First we try to give a quantitative measurement of the influence of temporal coherency on models' training (see Figure 3–1 lower part). Specifically, we train two versions of a model (with normal training frames and shuffled training frames as input respectively). When doing testing, we pass the normally-ordered test set through both of the two versions to compute two groups of prediction vectors. These two groups of vectors are both originated from the same normal test video set but processed by two differently trained models (whose architectures are also the same). This procedure can be interpreted as "two children are grown up in two environments with different temporal distributions (normal or shuffled training sets), and tested in the same normal environment to see how their understanding of the world differ". This measurement reflects how temporal information *influence* the formation of a model, and we name it as *Inf-Index*. Formally, we represent the two groups of prediction vectors as $\mathrm{Dom}(\boldsymbol{X}) = \{\boldsymbol{X}_1, \boldsymbol{X}_2, ..., \boldsymbol{X}_N\}$ and $\mathrm{Dom}(\boldsymbol{Y}) = \{\boldsymbol{Y}_1, \boldsymbol{Y}_2, ..., \boldsymbol{Y}_N\}$. N denotes the number of samples and $\boldsymbol{X}_i$ is the final prediction vector on probability of classes for sample i. These two groups could also be seen as two domains with paired instances whose temporal distributions are different. Note that the difference is not caused by the input data because for each pair of $(\boldsymbol{X}_i, \boldsymbol{Y}_i)$ the models are reading the identical input video with same temporal condition. It is the parameters learned with different training data (normal and shuffled) that cause the shift in the prediction vectors. Then the Inf-Index used to measure the distance between these two groups of predictions are defined as:

$$\text{Inf-Index} = Dis(\mathrm{Dom}(\boldsymbol{X}), \mathrm{Dom}(\boldsymbol{Y})) \tag{3–1}$$

The score of Inf-Index represents how the temporal information influences the formation (learning) of a model, or how likely a model picks up temporal cues during training. If this score is low, that means the perception of these two versions on the same videos are similar, thus the temporal information during training does not make the (normally trained) model feel much difference. For extreme cases where temporal

information is not modeled, e.g. average pooling among frames, training with normal and shuffle frames should make no difference therefore these two versions are just two times of training of a same model, which will result in low scores on Inf-Index, indicating no influence from time. Note that usually it is not 0 since this score is influenced by randomness such as stochastic gradient decent, random initialization, but should be low compared with scores by models with strong temporal modeling capability.

3.3.2 Sensitivity about Time

We are also interested in how sensitive a normally trained model is about the temporal coherency (see Figure 3–1 upper part), or if a model can feel the abnormality of the physical world like a human based on his experience (training). In this case, we simply train a model with normal training frames, and test it on normal and shuffled test set. We want to see the difference the model presents about different temporal information, just like a human if he can feel the disorder of frames in a video. We name this score as Sens-Index. The formula for this measurement is the same as the one in the last subsection 3.3.1:

$$\text{Sens-Index} = Dis(\text{Dom}(\boldsymbol{X}), \text{Dom}(\boldsymbol{X}')) \tag{3–2}$$

Here we are using the same calculation as Inf-Index. However, the semantics of these two scores are different. Sens-Index measures how sensitive a single trained model is about the temporal information, so the $\boldsymbol{X}$ is originated from normal test frames while $\boldsymbol{X}'$ is from shuffled test frames. But the two inputs are only different in temporal information thus spatially they contain the same contents. When the score is low, that means the model's perception on temporally-consistent videos is similar to the perception on abnormal videos (shuffled), e.g. for average pooling methods, this score should be 0 because all frames are average without further modeling. Otherwise, the trained model can feel the difference between normal videos and abnormal videos, just like a human can tell which sequence is ordered while the other is not.

A natural choice of the distance function is Root-Mean-Square deviation:

$$Dis(\boldsymbol{a}, \boldsymbol{b}) = \sqrt{\frac{\sum_{i=1}^{N} (\boldsymbol{a}_i - \boldsymbol{b}_i)^2}{N}} \tag{3–3}$$

where $(\boldsymbol{a}, \boldsymbol{b})$ can represent $(Dom(\boldsymbol{X}), Dom(\boldsymbol{Y}))$ or $(Dom(\boldsymbol{X}), Dom(\boldsymbol{X}'))$. There may be other more suitable distance functions but we leave this investigation as a future work.

Chapter 4 Evaluation on Representative Methods

4.1 Introduction

In this chapter, we conduct experiments on the selected representative methods (reviewed in Chapter 2) to give a quantitative evaluation on their performance of temporal modeling based on the protocols proposed in Chapter 3. We first introduce the datasets that will be used in our experiments, and we introduce the experimental details including training configurations for each method and the selected backbone architecture of convolution networks which should be shared across a large range of methods to keep a relatively fair comparison. Then we report our experimental results in section 4.4, 4.5, 4.6, and discuss valuable and interesting finds in section 4.7.

4.2 Datasets

Our research focuses on video action recognition using raw RGB frames as inputs without utilizing extra modalities such as skeleton data or depth information. There has been many related datasets introduced in the last few decades. Because of the limitation of time and resources, we focus our investigation on two action recognition datasets which possess different properties. The first is a very popular dataset, UCF101[32], which has been widely experimented and reported in the last five years. The second is Something-Something[22], a recently released dataset requiring more temporal modeling than usual datasets to reach high performance.

4.2.1 UCF101

UCF101[32] dataset consists of 13,320 realistic action videos clips collected from YouTube for 101 action classes, and there are at least 100 clips for each class. Action classification on this datasets is challenging owing to variations in pose, object appearance, camera motion and viewpoint. This dataset is a mid-scale dataset and widely used as a benchmark set in many previous literatures. The videos are grouped into 25 groups, where the same group can share some features such as similar background and viewpoint. There is an official division of the videos into five coarse categories (e.g. Human-Object Interaction, Playing Musical Instruments, etc.), but we did not adopt this feature in our experiments. There are three official train/test splits, but in this evaluation process we only report all results on the first split. Some examples from this dataset are shown in Figure 4–1. UCF101 dataset has been widely used to benchmark deep learning based methods in the last 5 years[7-10, 15, 17, 28, 31].

4.2.2 Something-Something

Something-Something[22] dataset consists of 108,499 videos across 174 classes, with videos' duration ranging from 2 to 6 seconds. The actions in video samples are performed by crowd workers who are asked to fill the label templates such as "Dropping [*something*] into [*something*]". Different from UCF101 in which scene

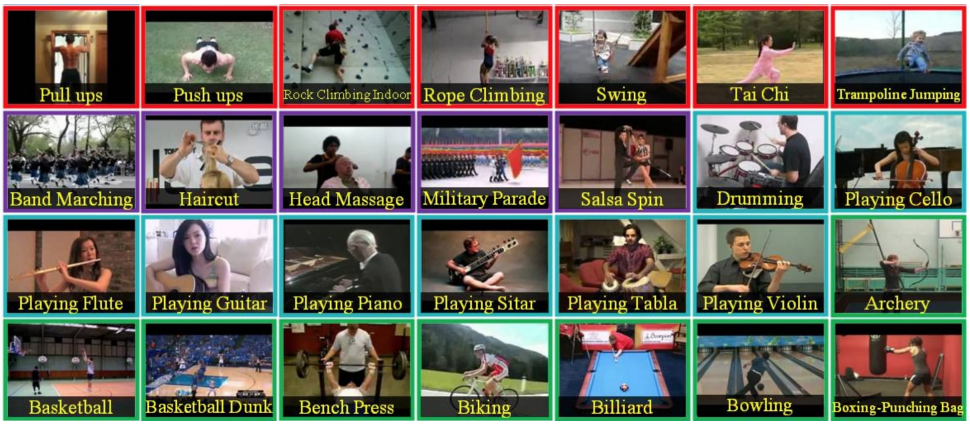

Figure 4–1 Examples taken from UCF101 dataset.

and appearance can greatly enhance recognition, this dataset consists of much more fine-grained physical causality and complex human-object interactions. Many videos require long-term observation to produce final correct predictions even for human. For example, "Pretending to put [*something*] onto [*something*]" may require checking last few frames to tell if an object is actually *on* something. To achieve high performance on this dataset, more delicate temporal modeling is required. This is a large scale dataset and contain around ten times more video samples than UCF101. Some examples from this dataset can be seen in Figure 4–2.

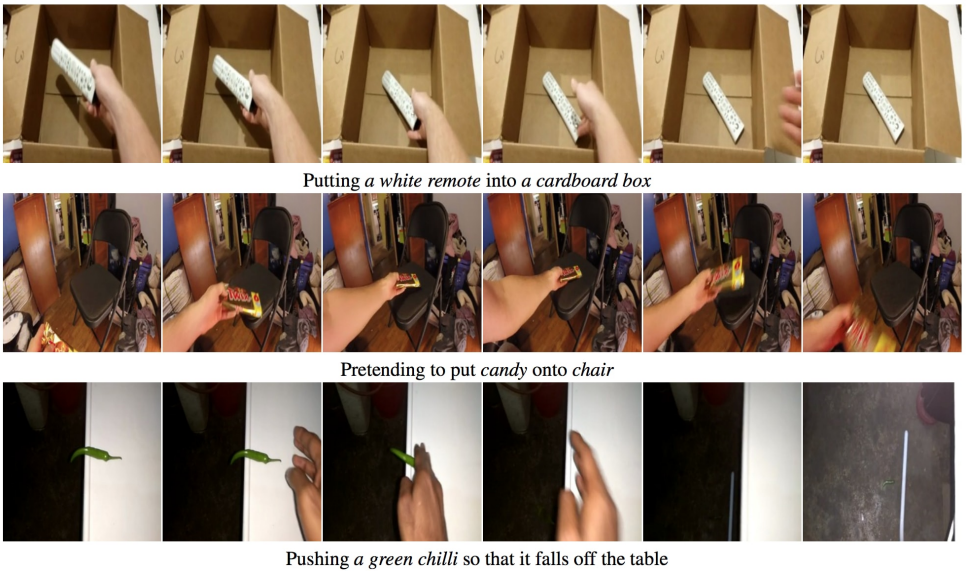

Figure 4–2 Examples taken from Something-Something dataset.

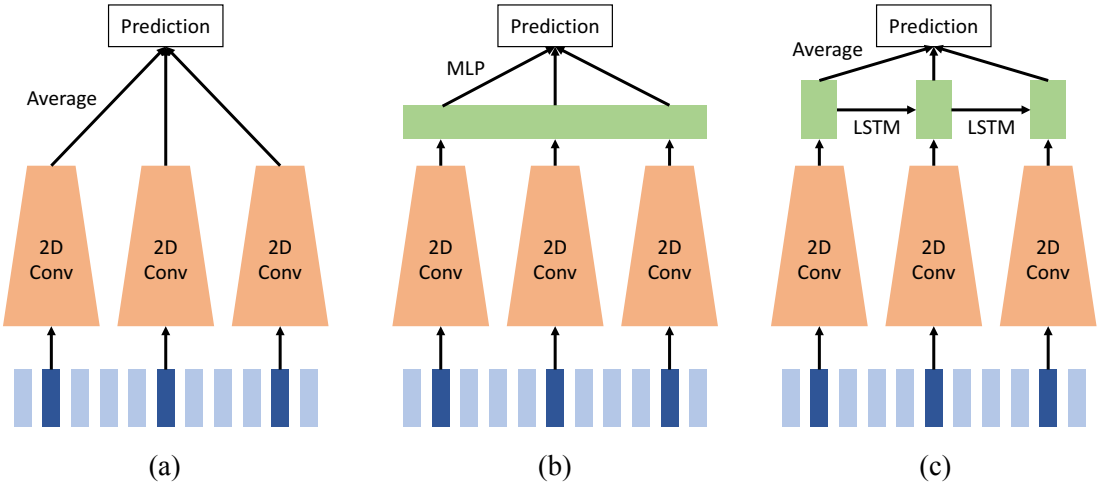

Figure 4–3 Illustrations for evaluated methods: (a) Average Fusion Pooling. (b) Temporal Relation model. (c) LSTM or Conv-LSTM model. Blue squares at the bottom are original video frames, and dark blue squares are selected frames for processing.

4.3 Experimental Details

4.3.1 General Configuration

We select the most representative models in action recognition to conduct our experiments. The selected methods include Average Pooling[10], Temporal Relation Network[34] (TRN), LSTM model[8], Conv-LSTM model[23], Optical Flow representation[7], Improved Dense Trajectory[5], and 3D Convolution model[12]. The illustration of these models are shown in Figure 4–3 4–4, while detailed introductions could be found in Chapter 2. In particular, LSTM based model, optical flow (two-stream type models), and 3D convolution models are regarded as the state-of-the-art deep learning methods in action recognition which exploit temporal information[34, 59].

Since our evaluation protocols haven't been used on any other existent methods, we need to conduct all the experiments on each model and dataset by ourselves, which dramatically increases our experimental period and implementation work. Therefore it is essential for us to properly choose the configurations of the tested models to make the experiments feasible while keeping relatively fair comparison and getting valuable results. Note that in this investigation we are not targeting at searching the optimal implementation of each method and we do not exploit any ensemble techniques which usually boost the performance of a method. Instead, we focus on evaluating if a model takes temporal information into account by the protocols we proposed in Chapter 3, so for all deep learning methods (all methods except IDT), we conduct testing with single center crop on the video frames instead of some widely used testing techniques like 10 cropping with flipped 4 corner and center cropping[10], thus our reproduction may result in decline in performance compared with those reported in the original papers.

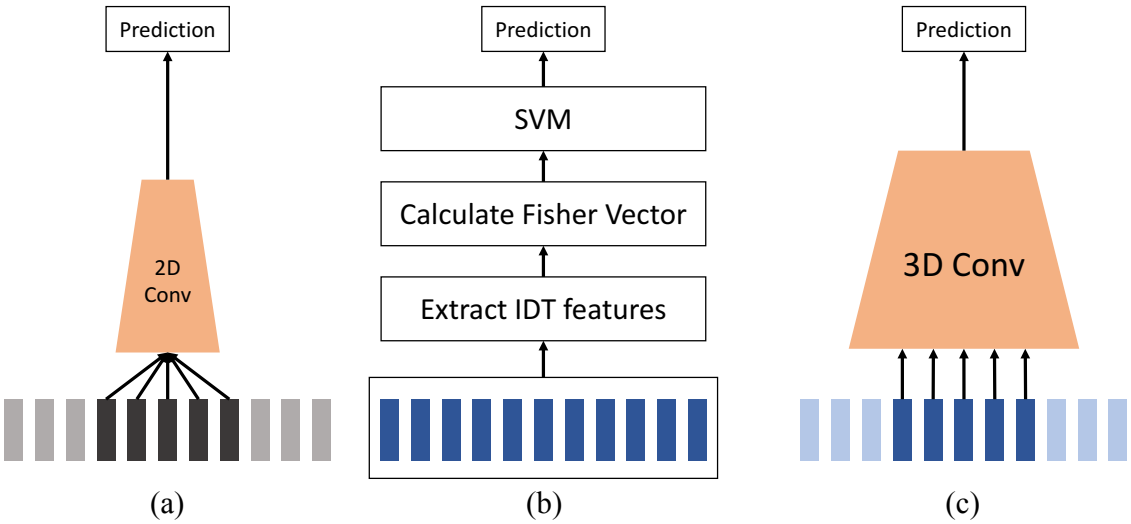

Figure 4–4 Illustrations for evaluated methods: (a) Optical Flow. (b) Improved Dense Trajectory. (c) 3D Convolution. Blue squares at the bottom are video frames, and dark blue squares are selected frames for processing. Grey squares represent optical flows.

Now we present the training and testing details for each method evaluated in our experiments. For Average Pooling, LSTM, Conv-LSTM, and optical flow, we spatially crop a 224×224 image as input on UCF101 dataset and 96×96 on Something-Something dataset, randomly sampled from four corner or center of a video during training, and we use the center cropping as the test input data. For TRN the input size on both datasets is 224×224. For the 3D convolution, the cropped sizes for UCF101 and Something-Something are reduced to 112×112 and 84×84 for more efficient computing. During training, we temporally sample 3 frames from a video (each randomly sampled from a evenly divided segment) for Average Pooling, following[10], 8 frames for TRN which is the best single scale configuration in[34], and 16 frames for LSTM and Conv-LSTM models following[8]; when testing, the frames are generated using the temporally center frame for each segment. For the 3D convolution, instead of sparsely sampling frames in each segment, we randomly sample a starting point and use the following consecutive 32 frames as the input data as in[12], and tested with temporally center cropped consecutive frames. For optical flow model, we train by sampling 10 consecutive flow fields each with 2 channels on UCF101 and Something-Something datasets. We use the TV-L1 algorithm to extract the flow fields same as in[10]. Note that these parameter selections roughly follow their original papers as we cannot search for an optimal solution for each single method, and we also know that using totally same sample configuration (e.g. all same frame size or length) is not a good choice because for some methods increasing sample length does not benefit performance such as Average Pooling, but some methods rely on longer temporal sampling to show their strength, e.g. TRN. For LSTM and Conv-LSTM models, we set the dimension of the hidden states and cells to be 512, and for TRN, two-layer MLPs with 256 units are used, same as the original paper[34]. All these models with RGB inputs are trained for 80 epochs

Table 4–1 ResNet-34 architecture. The input is 224×224.

Layers		Output Size
conv1	7×7, 64, stride 2	112×112
max pool	3×3 stride 2	56×56
conv2_x	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	56×56
conv3_x	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	28×28
conv4_x	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	14×14
conv5_x	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	7×7
average pool	7×7	1×1

while the model with optical flow is for 340 epochs as it takes much longer to converge. The best validation result for each model is reported. For the only handcrafted model, improved dense trajectory, we use the officially released public code to generate features for each video, and train a Gaussian Mixture Model (GMM) with $K = 256$ using 256000 randomly sampled IDT features, then calculate the Fisher Vectors to represent a video; we use PCA to reduce the final FV dimension to 6000 for faster training of the final one-vs-rest linear SVM. When the frames in training phase and testing phase are of different order, the GMM is trained using training features, and the GMM parameters are applied to testing set to produce fisher vectors. We leave the convolutional architecture selection to the next subsection to introduce.

4.3.2 Convolutional Neural Network Architecture

A shared property between all deep models is the backbone convolutional neural network (CNN) architecture. Since the boost of deep convolutional neural networks from 2012[60], there has been a great development of CNN architectures[51, 53, 61], and the action recognition performance is correlated with the strength of CNN architectures. Since the different methods for action recognition are usually proposed along with different convolutional architectures, it is quite obscure to determine whether the performance boost in a task is due to the special temporal modeling or the strengthened backbone architecture. Because our experiments focus on evaluating temporal modeling for each method, it is natural to fix their backbone convolutional architecture to give a relatively fair comparison between different methods only on their temporal modeling capacity.

As for the selection of the specific network, since we need to train many models, the CNN architecture should not be too deep as efficiency is very important. In our implementation, we adopt the ResNet-34[51] with zero-padding shortcuts (type A shortcut) architecture (see Table 4–1) as our backbone convolutional architecture due to its balance between good performance and relatively low complexity. We use it for both

Table 4–2 Backbone architecture performance for 3D ResNet methods. Results are taken from[12]. Accuracies are measured on Kinetics[50] validation set.

Method	Top-1	Top-5	Average
3D-Res-18	54.2	78.1	66.1
3D-Res-34	60.1	81.9	71.0
3D-Res-50	61.3	83.1	72.2
3D-Res-101	62.8	83.9	73.3

2D and 3D convolution architectures for simplicity. A performance investigation of different layers of 3D ResNets also proved the ResNet-34 is a balanced choice between complexity and performance[12] (see Table 4–2). For all 2D convolution based models, we initialize them with ImageNet dataset pretrained parameters, and for 3D convolution based models with pretained parameters on Kinetics[50] dataset. Note that 3D ResNet-34 contain more parameters than its 2D version, thus their accuracies are not directly comparable, but their own differences between normal and shuffled version do.

4.4 Results

In this section, we report the experimental results of the evaluated methods. The results on UCF101 and Something-Something are presented in section 4.4.1 and 4.4.2 respectively.

4.4.1 Results on UCF101

The accuracy results of each model on UCF101 are shown in Table 4–3. In order to specify the different temporal conditions in training and testing phases, we adopt the pattern "*-*", where "*" can be replaced by "n", "s", or "r" to denote "normal", "shuffled", and "reversed" conditions respectively. For instance, "n-s" means this model is trained with normal frames, and tested using shuffled frames. The results are all reproduced by ourselves. Since we do not adopt any fancy testing protocols such as "10-crop 25-segment", the accuracy results may be lower than those in the original papers, but could be potentially boosted using those techniques by a large margin. The results of Sens-Index and Inf-Index are shown in Table 4–4.

Average Pooling: The performance of Average Pooling method is very impressive as there is no explicit temporal modeling in it. As expected, the results in different testing conditions within the same trained version are identical and we just use the left arrow to indicate this case. Our proposed Sens-Index also agree with this (see Table 4–4). There is a slight accuracy difference between the two trained versions, and our Inf-Index also gives a non-zero influence result as well. In order to show more interpretability, we plot a figure to present the top 10 classes with accuracy increase or decrease under two training conditions in Figure 4–5. We can see clearly that frame order is definitely influencing the accuracies, but since Average Pooling does not take any temporal coherency information into account, we attribute this difference to the randomness during training (e.g. randomness in SGD, batch order, etc) instead of the presence of temporal information.

Table 4–3 Top-1 accuracy of different methods under different temporal conditions on UCF101 dataset. All results are reproduced by ourselves, therefore may not exactly match the values in the original papers. In the first row, "n", "r" and "s" denote input data with normal, reversed and shuffled frames, respectively. The letter on the left of "-" means training phase, and on the right means test phase. For instance, "n-r" means the model is trained with frames in normal order and tested in reversed order.

Method	UCF101					
	n-n	n-r	n-s	s-n	s-r	s-s
Avg Pooling[10]	80.78	←	←	80.39	←	←
Temporal Relation[34]	82.10	78.77	80.04	80.07	79.99	80.20
Regular LSTM[8]	80.33	79.17	79.75	79.83	80.33	80.28
Conv-LSTM[23]	79.88	79.33	80.33	79.17	79.46	79.80
Optical Flow[7]	72.09	70.71	70.42	71.19	70.10	70.66
IDT[5]	82.13	-	8.30	27.76	-	67.22
3D-Res-34[12]	80.94	80.94	72.01	70.47	70.21	71.79

Table 4–4 Inf-Index and Sens-Index measurements (see Chapter 3) among different methods on UCF101 dataset. Inf-Index reflects how the temporal information influences the formation (learning) of a model, and Sens-Index measures how sensitive a normally trained model is about the temporal coherency.

Method	UCF101	
	Inf-Index	Sens-Index
Avg Pooling[10]	0.01412	0.0
Temporal Relation[34]	0.01808	0.00496
Regular LSTM[8]	0.01686	0.00508
Conv-LSTM[23]	0.01437	0.00775
Optical Flow[7]	0.02161	0.01280
IDT[5]	0.00486	0.00519
3D-Res-34[12]	0.03098	0.02464

As we can expect, both the Sens-Index and Inf-Index of Average Pooling are the lowest among all methods (except for IDT, which is trained using Hing Loss with SVM and is not comparable using Inf-Index and Sens-Index), indicating this actually plays the role of baseline for temporal modeling.

Temporal Relation: This model was introduced by Zhou et.al. in paper[34] (see Figure 4–3 for an illustration). In the original paper, this model did not show outstanding performance on UCF101 dataset, and Zhou et.al. never highlighted its accuracy on UCF101. But in our experiments, when evaluated with

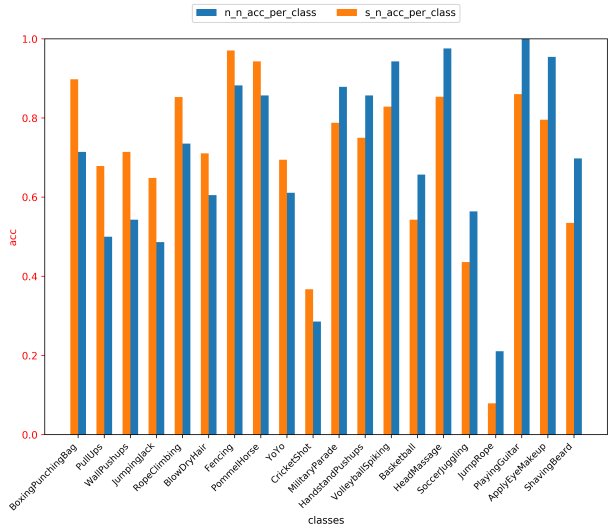

Figure 4–5 The top 10 classes with accuracy increase or decrease when trained with shuffled frames or not for Average Pooling. Left half axis shows the classes whose accuracies are most higher when trained with shuffled frames, and right half are for most lower.

only single center-crop testing scheme, Temporal Relation model achieves the highest accuracy among all deep methods. This may be caused by that the additional parameters brought by the MLPs in Temporal Relation model leads to a deeper CNN, which could be more discriminative than the usual ResNet-34 used in all networks. The relatively large accuracy difference between normal and shuffled testing data of normally trained version (n-n vs n-s) shows that this model can greatly exploit temporal information to achieve high accuracy. However, the low Sens-Index says this model adopts the lowest amount of temporal information compared with other approaches on UCF101. What causes this conflict? We plot the class-level top 10 increase and decrease between n-n and n-s, plus the corresponding class-level Sens-Index in Figure 4–6. This figure shows that the accuracy difference for all classes are low except for category *HighJump*, in which the gap is approximately 60%, and its Sens-Index is about 3 times higher than the other classes! This can explain why the Sens-Index of this model is low while an obvious gap between accuracy of n-n and n-s still exists. We believe this phenomenon indicates the Temporal Relation model's strong potential for elastically extracting temporal information for each class, e.g. all other classes on UCF101 require very weak temporal modeling while *HighJump* is an exception. This capability is caused by the additional cross frame MLPs brought by Temporal Relation Network which can capture cross frame patterns.

Regular LSTM and Conv-LSTM: Let's take a look at the LSTM based models, including regular LSTM and Conv-LSTM (see Figure 4–3 (d)). In fact, the results are really counter-intuitive because normally the LSTM models are good at dealing with sequential data, indicating their outstanding performance on temporal modeling. The accuracy results show that all trained versions perform almost identically on all types of testing sets, i.e. no matter trained or tested with temporal information preserved or not, all the same!

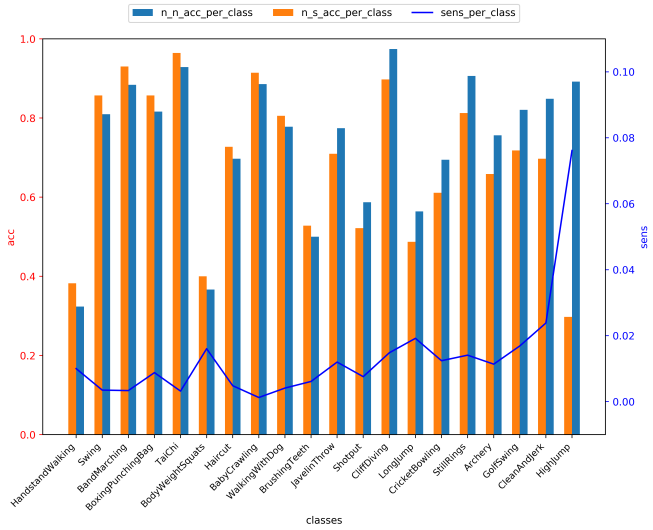

Figure 4–6 The top 10 classes with accuracy increase or decrease when trained with shuffled frames or not for Temporal Relation. Left half axis shows the classes whose accuracies are most higher when trained with shuffled frames, and right half for most lower. Blue line shows the class-level Sens-Index for each corresponding class to show their sensitivity to temporal coherency.

Moreover, not only the accuracies show such a phenomenon, our proposed Sens-Index and Inf-Index also agree with it, i.e. they are very low. The Inf-Index is very close to Average Pooling method which sees no temporal information, and Sens-Index indicates the model is not sensitive to temporal coherency. Here we also show the 10 classes with most accuracy changes between shuffled and normal test frames (n-n vs n-s) for regular LSTM and Conv-LSTM in Figure 4–7. We can see that for both two LSTM models, shuffling test frames doesn't dramatically change their accuracies for each class. Based on all these observations, we conclude that LSTM models cannot see the temporal dynamics on UCF101 datasets, or LSTMs adopt the strategy to ignore temporal structure for better performance on UCF101. This also agrees with the fact that nowadays LSTM models are not as promising as 3D convolutions or two-stream models using optical flow, when benchmarked on UCF101 or similar datasets.

Optical Flow: Then we evaluate optical flow representation to see if it can capture any temporal information (illustrated in Figure 4–4(a)). We use the TV-L1 algorithm to extract the flow between two consecutive frames using OpenCV, and the network architecture is similar to single frame method but with the first layer modified to handle 10 consecutive optical flow fields with 20 channels, including both x and y directions (could be seen as the temporal branch of two-stream model[7], but implemented with ResNet-34 backbone). But conceptually, optical flow itself is calculated by moving blobs of consecutive frames, therefore a single flow frame has already contain motion information. Shuffling the flow fields breaks the temporal coherency of motions, but cannot break the causality of instant motions captured by a single flow field. Thus the differences measured by shuffling in our protocol are about the temporal coherency of motions, but

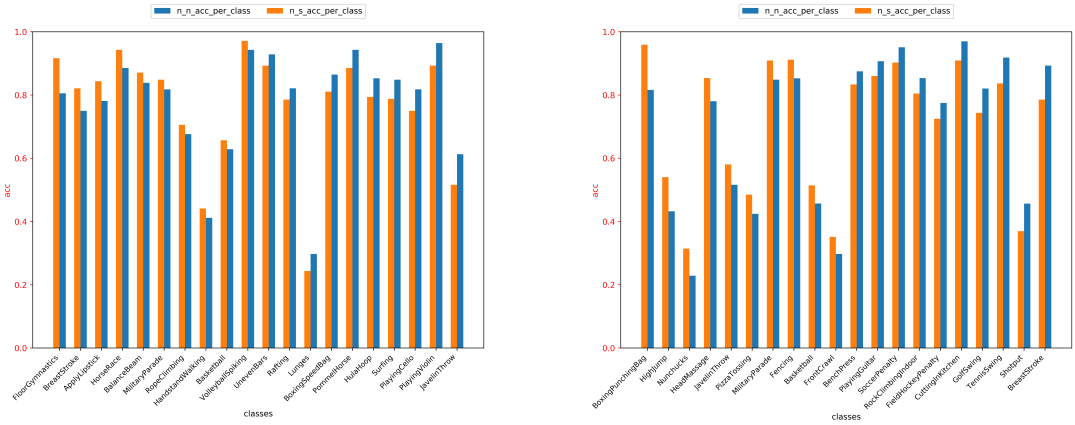

Figure 4–7 The 10 classes with the highest accuracy increase and 10 classes with the highest accuracy decrease for regular LSTM (left) and Conv-LSTM (right) between normal and shuffled test frames.

temporal information contained in a instant motion can still be adopted in shuffled version. The accuracies for normally trained version show that there is a slight better performance (about 1 to 2%) on normal test set than shuffled and reversed sets. But generally speaking, we cannot obtain much clue about it's awareness about temporal coherency through accuracy measurement. However, these results do not agree with the experiments in[33] in which the authors also shuffled the test set while showing obvious gap between shuffled and normal results. In addition, our proposed Sens-Index and Inf-Index also indicate this model *can* feel the difference caused by temporal coherency, since they are obviously higher than other evaluated methods. But what caused these unexpected results? Before we present our answers, let's consider another interesting phenomenon of optical flow. In most of the previous literatures[7, 14, 19], performance using optical flow can usually outperform RGB modality on UCF101 dataset, but in our experiments this is not the case. To answer these questions, we need to remind us that our experiments do not adopt any fancy testing schemes such as "10-crop 25-segment" testing, while others do. Thus in our case, the optical flow network only see a small range of consecutive fields, which may not lead to a strong prediction on each test example. Also the consecutive frames are relatively similar, leading to no much performance drop when testing on shuffled set. To confirm this hypothesis, we conduct the "10 crops and 25 segments" testing on the normally trained model, and the accuracy for normal test set increases to 83.05%, and for shuffled test set decreases to 68.18%. This is much more similar to the results in[33] with only small difference. Considering we are using a weaker backbone architecture, this is acceptable. The decrease on shuffled test set is caused by that when using more segments, frames from a larger time span are shuffled together, which is not what the model was trained to handle (trained with near frames). Then we conduct this "10-crop and 25-segment" testing scheme on Average Pooling method, and we get accuracy of 82.73%, which agrees with results in previous literatures that optical flow can beat RGB by a slight margin on UCF101 dataset. Now let's rethink about the shuffled flow fields. It becomes a little weird if we dig deeper because usually we think models would rely on appearance information to do recognition when frames are shuffled, but here the fact is the shuffled

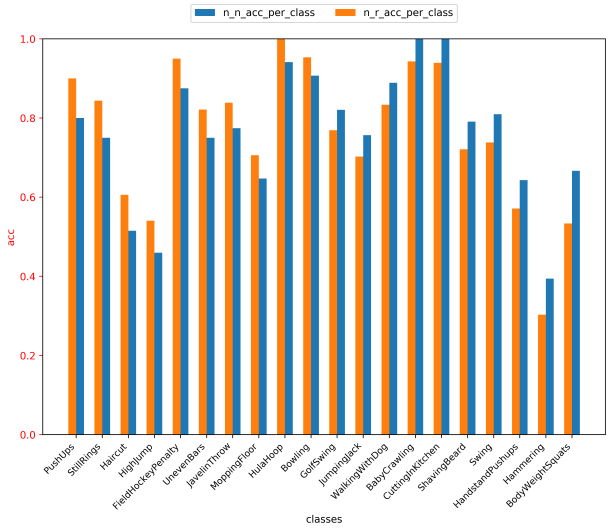

Figure 4–8 The 10 classes with the highest accuracy increase and 10 classes with the highest accuracy decrease for 3D Convolution between normal and reversed test sets.

flow fields do not contain appearance information! Instead it represents instant motion information that is invariant to appearance, but without temporal coherency. This indicates that temporal coherency is not the most value of small-scale consecutive optical flow fields, but the invariance to appearance (which agrees with the conclusion in[33]). To prove this, we reconduct the "10-crop 25-segment" experiment on n-s version to shuffle only flow fields within each segment (thus consecutive frames are still near each other), we get the accuracy of 82.57%, which is very close to n-n case and agree with the results in single crop testing. But we still need to note that when long-term video has been taken into account, the performance of optical flow still strongly rely on long-term consensus (25-segment augmentation) and the value of optical flow comes from both invariance to appearance, and its effectivity for long term consensus, i.e. optical flow can significantly benefit from 10-crop 25-segment consensus, which is usually not achievable by RGB modality. But the drawback of this method is obvious: because of the long time for preprocessing and postprocessing, it could not be used for real time application.

Improved Dense Trajectory: From the accuracy, we can see the IDT rely heavily on temporal coherency to get high accuracy. It is also very sensitive to the distribution shift between shuffled and normal data even it was trained with shuffled frames. Since it is a handcraft model and delicately designed to capture temporal patterns using consecutive frames, it is understandable that IDT seems to "hardwire" temporal information into its classification, and is not as adaptable as deep learning based models. However, the Sens-Index and Inf-Index show strangely low values and say IDT takes very little temporal information into account! This is due to the different training loss it uses compared with other deep learning based models. In fact, IDT uses Hinge Loss trained with an One-vs-Rest SVM, while other models use CrossEntropy Loss. It is optimized to maximize the margin between classes while CrossEntropy Loss is used to give a probability on the target

Table 4–5 Top-1 accuracy of different methods under different temporal conditions on Something-Something dataset. All results are reproduced by ourselves, therefore may not exactly match the values in the original papers. In the first row, "n", "r" and "s" denote input data with normal, reversed and shuffled frames, respectively. The letter on the left of "-" means training phase, and on the right means test phase. For instance, "n-r" means the model is trained with frames in normal order and tested in reversed order.

Method	Something-Something					
	n-n	n-r	n-s	s-n	s-r	s-s
Avg Pooling[10]	14.15	←	←	14.32	←	←
Temporal Relation[34]	29.58	9.81	15.01	16.61	16.69	16.73
Regular LSTM[8]	18.03	10.61	13.77	14.67	14.41	15.28
Conv LSTM[23]	19.87	10.04	12.86	15.64	14.73	16.81
Optical Flow[7]	19.85	17.18	17.65	17.99	17.96	18.13
3D-Res-34[12]	33.88	10.93	1.82	11.43	11.86	14.94

class close to 1 while other classes to 0. This makes the Sens-Index and Inf-Index on these two types of loss incomparable. From now on, we will not use Sens-Index and Inf-Index on IDT model in this thesis.

3D Convolution: The accuracies of normally trained 3D Convolution on normal and reversed testing set are the same, and is significantly worse on shuffled frames. This result is similar to the finds in paper[13] by Xie et.al., in which they thought it may be due to either the model or the dataset that does not allow or require inferring the arrow of time. However, if we plot the class-level accuracy difference between normal and reversed testing sets (see Figure 4–8), we can clearly see that the model performs differently between these two temporal conditions, although the gaps are not very large. At the same time, in the Table 4–5, the performance for normal testing set is definitely higher than reversed set, therefore we can safely conclude that it is the UCF101 dataset (and Kinetics dataset in[13]) that accidentally lead to the equality of these two accuracies. But we should also conclude that most of the instances of UCF101 do not require much fine-grained temporal modeling to tell if a sequence is reversed. There are two other interesting points for 3D Convolution. The significant drop on shuffled testing set for normally trained version (n-s), and the high value in Sens-Index both show that 3D Convolution exploits the most temporal information among all methods on UCF101 dataset. But considering this dataset should not require much temporal modeling to achieve high accuracy (as Average Pooling can reach high accuracy), 3D Convolution shows an unnecessary strong response to temporal dynamics as it doesn't help much for the absolute accuracy gain, and may has harmed the model's robustness for temporal coherency. Compared with Temporal Relation model, 3D Convolution shows less elasticity for temporal modeling. This may be due to that this model was pretrained on Kinetics video dataset instead of ImageNet dataset, leading to a higher sensitivity to temporal pattern and a weaker ability of 2D perception, which has been also indicated by its poor performance by shuffle-trained version (around 70% for s-n, s-r, and s-s).

Table 4–6 Inf-Index and Sens-Index measurements (see Chapter 3) among different methods on Something-Something dataset. Inf-Index reflects how the temporal information influences the formation (learning) of a model, and Sens-Index measures how sensitive a normally trained model is about the temporal coherency.

Method	Something-Something	
	Inf-Index	Sens-Index
Avg Pooling[10]	0.00640	0.0
Temporal Relation[34]	0.03847	0.03289
Regular LSTM[8]	0.01810	0.00967
Conv LSTM[23]	0.01904	0.01668
Optical Flow[7]	0.01179	0.01095
3D-Res-34[12]	0.04284	0.06863

4.4.2 Results on Something-Something

The accuracy results of each model on Something-Something dataset have been shown in Table 4–3, while Sens-Index and Inf-Index in Table 4–4. The notations are of the same meanings as in Section 4.4.1, where "n", "r", and "s" denote "normal", "reversed", and "shuffled" frames respectively. We still didn't use any testing schemes like "10-crop 25-segment" but show the "raw" performance of each model using center-crop testing, spatially and temporally.

Average Pooling: Similar to the case of UCF101, Average Pooling performs equally well on normal frames and abnormal fames. This model definitely plays the role of baseline, and we can clearly see when no temporal information is taken into account, the accuracy of a model on Something-Something should be around 14% to 15%. As expected, the Sens-Index is 0 because predictions are the same between n-n and n-s. The Inf-Index is 0.0064, which is the lowest one compared with all other models which more or less model some temporal dynamics, indicating the randomness during training (initialization, mini-batch order, etc) that accounts for this quantity.

Temporal Relation: The performance of TRN on Something-Something is similar to it on UCF101 dataset, but with some values significantly exaggerated. It achieves 29.58% accuracy on Something-Something dataset, and is evidently higher than all other methods except 3D Convolution. We can see an obvious accuracy gap between n-n and n-s, and even larger gap between n-n and n-r, indicating the high performance of TRN really comes from its outstanding temporal modeling. The performance of the shuffled trained version on all testing sets are almost identical (slightly higher on s-s than s-n and s-r), showing this model is utilizing the co-occurrence of frames to inference the class label. This is unlike Average Pooling model which does not adopting any co-occurrence information but just an averaging consensus scheme to produce a prediction. For Sens-Index, unlike in UCF101 case where only one class requires temporal modeling, it now indicates an overall high temporal sensitivity, showing this model can actually capture quite a lot

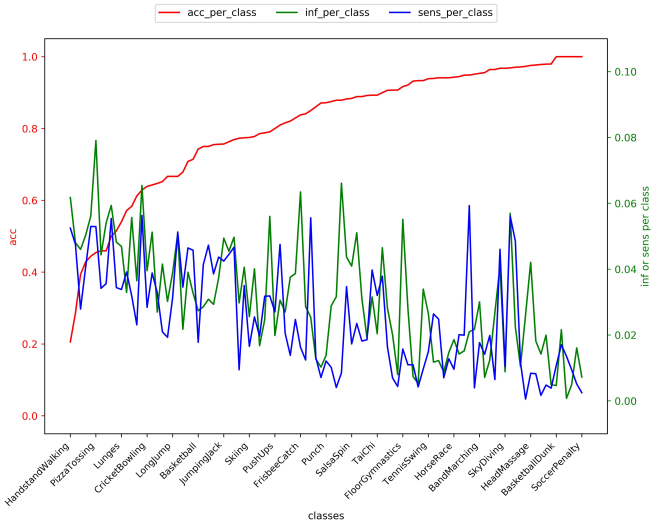

Figure 4–9 Class-level accuracy n-n, Inf-Index, and Sens-Index of 3D Convolution model on UCF101 dataset. Sorted by accuracy n-n. We can see classes with high accuracy tend to be less sensitive to, and less influenced by temporal information.

temporal dynamics. The Inf-Index also agrees with this situation. Thus we conclude that Temporal Relation Network can actually make use of temporal information to do action recognition on Something-Something dataset.

Regular LSTM and Conv-LSTM: Different from the case on UCF101 dataset, LSTM models on Something-Something show obviously more positive performance for temporal modeling, e.g. now they are both evidently higher than Average Pooling method. The accuracy gaps between normal, shuffled, and reversed testing frames are also clear and consistent. The performance of Conv-LSTM is 2 percent higher than regular LSTM, and we think it is due to the additional parameters in Conv-LSTMs used to capture more spatial information in LSTM transition operations, which makes the model more discriminative. However, compared with Temporal Relation model and 3D Convolution, LSTM models seem to be less efficient and effective to model videos. We think this is because LSTMs try to use shared weights to model all types of transitions which is not suitable for videos with high temporal dynamics, while Temporal Relation model and 3D Convolution allows more capability by non-shared parameters on time axis. Generally speaking, LSTMs can exploit a level of temporal information on Something-Something, but is not very efficient and the specific reason for it requires more future research.

Optical Flow: The absolute accuracy of Optical Flow method on Something-Something (n-n) is 19.85% and is very similar to the performance of LSTM based models. However, this does not hold for reversed or shuffled testing sets (n-r and n-s) where its performance is obviously higher than those of LSTMs. As we have analyzed in UCF101's subsection, the Optical Flow modality contains some instant motion information even when the flow fields are shuffled or reversed, and also because the input frames of Optical Flow are in short range and flow fields are similar to each other, temporal coherency is not so influential to the performance.

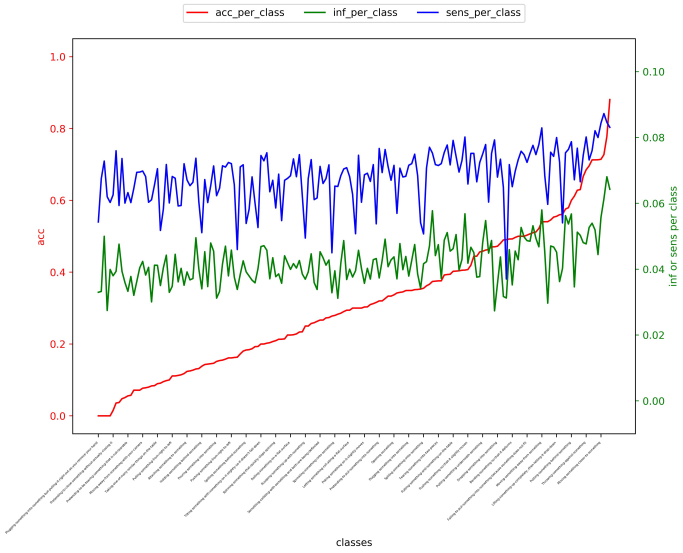

Figure 4–10 Class-level accuracy n-n, Inf-Index, and Sens-Index of 3D Convolution model on Something-Something dataset. Sorted by accuracy n-n. Different from UCF101 (Figure 4–9), here we can see a slight trend that classes with high accuracy tend to be more sensitive and more influenced by temporal information.

Therefore it achieves the highest performance among all approaches on shuffled and reversed testing sets (n-r and n-s), and also all the testing sets by shuffled trained version (s-n, s-r, and s-s). However, a strange thing comes from Inf-Index and Sens-Index. In UCF101 case, Inf-Index and Sens-Index are both high for Optical Flow method, but on Something-Something dataset, they are all very low compared with other approaches. Actually, this can be inferred from accuracy (Table 4–5) since its performance gap between n-n and n-s (and between n-n and s-n) is the lowest compared with other methods. The reason still comes from the modality itself: optical flow contains instant motion information which already consists of some temporal information to help do classification, while RGB frames can not offer this benefit. The model can exploit the co-occurrence of motion pieces for prediction while other approaches can only rely on appearance without temporal information. This model shows us that even when temporal coherency is not maintained, temporal information hidden in flow fields can also help do classification. Therefore on Something-Something, the low quantity of Inf-Index and Sens-Index tell us that Optical Flow method's temporal capability comes from its instant motion information within individual flow fields rather than the coherency of multiple frames. But why Inf-Index and Sens-Index are high on UCF101 dataset? This is because UCF101 dataset requires very little temporal modeling, and thus the network chooses to exploit flow modality largely by its flow pattern in multiple fames (which is invariant to appearance), rather than its motion information in individual flow fields. Now we can conclude that, Optical Flow is valued in both invariance to appearance, and the encoded temporal information by motion, but how much of either two are used depends on the specific dataset.

3D Convolution: Different from the UCF101 case where 3D Convolution didn't show advanced absolute performance among all methods, it achieves the highest performance on Something-Something dataset. On

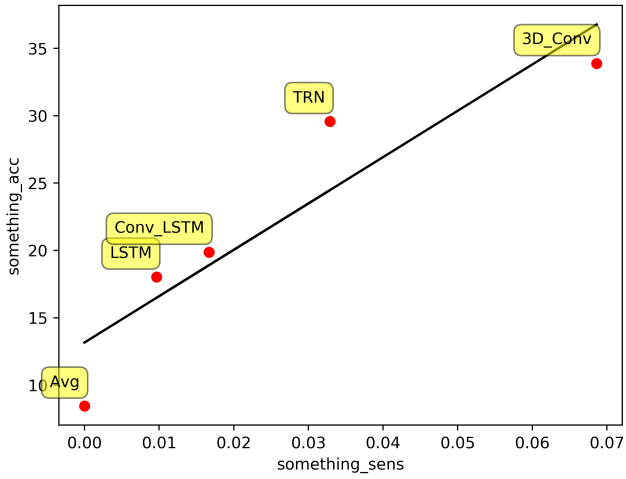

Figure 4–11 The correlation between Sens-Index and Accuracy on Something-Something dataset. We can see the correlation is obvious, indicating that a model's sensitivity to temporal coherency contributes to its performance on this dataset.

the other hand, the normally trained version's performance on shuffled testing set (n-s) does not work at all. The Sens-Index also says it is the model that use the most temporal information among all evaluated methods as it is most sensitive to temporal coherency. We compare the class-level Inf-Index and Sens-Index of 3D Convolution on UCF101 and Something-Something datasets in Figure 4–9 and 4–10. In the two figures, we sort the classes by their absolute accuracy (n-n), and plot the Inf-Index and Sens-Index as well. We can see, on UCF101 dataset (see Figure 4–9) high accuracy classes tend to be less sensitive to and less influential by temporal information, while on Something-Something (see Figure 4–10) there is a trend that high accuracy classes rely on more temporal modeling. These are strong hints showing that when temporal information on a dataset is important for classification, 3D Convolution tend to strongly rely on the temporal pattern, and it is obvious that UCF101 and Something-Something are two datasets with very different temporal modeling requirements. An interesting phenomenon is that on Something-Something, 3D Convolution seems to over rely on temporal patterns though sometimes it seems not very necessary, especially compared with the higher performance from other methods in n-s case. It is still not clear whether this tradeoff is necessary for temporal modeling (high n-n vs low n-s), but we will conduct more experiments to explore their relations in future works. When we look at the trained version with shuffled training set, there is a clear accuracy drop when the testing temporal distribution is shifted (compare s-s with s-n and s-r). This accuracy drop is more obvious than other methods like LSTMs, indicating 3D Convolution is more sensitive to temporal distribution changes. Besides, we can also see its performance of the shuffled version (s-s) is poor than that of LSTM models. We believe this is caused by the 2D classification capacity which is strongly influenced by the pretraining dataset. In our case, 3D Convolution model is pretrained on Kinetics dataset while others are pretrained on ImageNet dataset, indicating Kinetics is more suitable for pretraining to exploit temporal modeling, while for problems requiring accurate 2D classification, ImageNet should be a better choice.

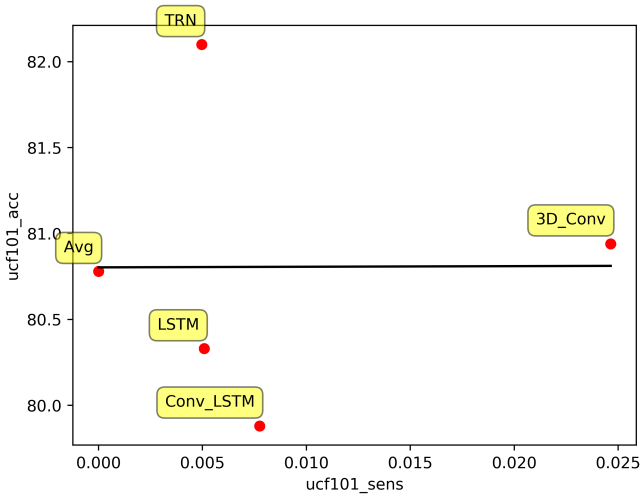

Figure 4–12 The correlation between Sens-Index and Accuracy on UCF101 dataset. We can see the there is no obvious correlation, indicating that on UCF101, temporal sensitivity is not very useful for high performance. This is also why most of the evaluated models rely on appearance for action recognition on UCF101.

4.5 Relation Between Accuracy and Temporal Sensitivity

Besides only measuring a model's sensitivity to temporal coherency, the Sens-Index is also supposed to provide a rough performance estimation of a model on a dataset that requires temporal modeling. This is hinted by the positive correlation between the Sens-Index and Accuracy on Something-Something dataset. Here we plot the regression in Figure 4–11 of Something-Something. The correlation is obvious, indicating that a model's sensitivity to temporal coherency contributes well to its performance on this dataset.

Note that, this conclusion holds for datasets requiring temporal modeling only, e.g. Something-Something. As a comparison, we plot the same regression for UCF101 dataset in Figure 4–12. Now we can easily see that temporal modeling is not correlated with the overall performance on UCF101 dataset. This why most of the evaluated methods performs equally well on this dataset and almost all of them try to rely on appearance for action recognition.

One may ask why Sens-Index is useful if it is correlated with accuracy. There are two main reasons for it.

- First is that, Sens-Index does not rely on ground truth labels. This is a better property than accuracy measurement because we can augment the training set by semi-supervised learning or unsupervised learning using unlabeled videos, considering video labeling is very expensive especially for videos requiring long-term reasoning. We may optimize the Sens-Index by considering it as another task paralleled with accuracy, or we may use more unlabeled data to pretrain the model. Incorporating the semi-supervised learning is a potential future work of this thesis.
- Second is that, even appearance-oriented datasets like UCF101 can activate the Sens-Index of a model which can help to estimate a model's performance on other datasets, e.g. Something-Something. To

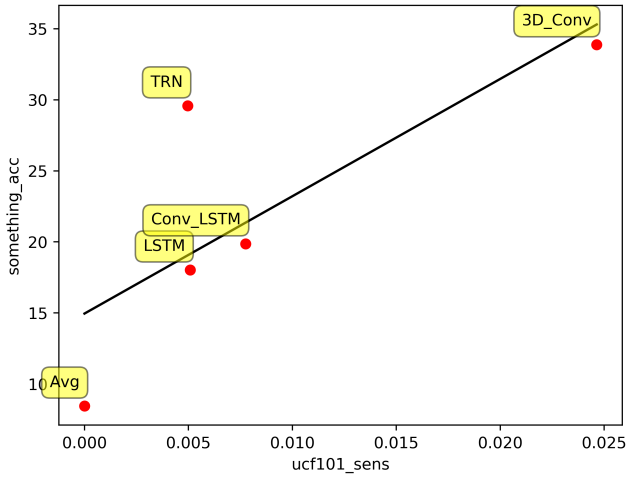

Figure 4–13 The correlation between Sens-Index on UCF101 and Accuracy on Something dataset. Surprisingly, an obvious correlation could be observed, though these two parameters are only connected by models rather than datasets.

show this, we plot the regression between Sens-Index on UCF101 and the Accuracy on Something-Something in Figure 4–13. We can see, though these two parameters are describing different datasets, they are correlated with each other and the only thing connects each other is the model. This indicates that even temporal sensitivity is not accounts much to the absolute performance on UCF101 dataset, its Sens-Index score is still meaningful and could be used to estimate the performance of the model on a dataset that requires temporal modeling.

4.6 Relation Between Two Datasets

In this section, we ask the question that if there is a way to quantitatively describe a dataset's requirement for temporal modeling. Normally there is no such an intuitive measurement to give us the answer because we have seen that when datasets require very little temporal modeling, the accuracy will conceal a model's capability for capturing temporal dynamics. But since we have discovered that Sens-Index is still useful on appearance-oriented datasets, we may use this measurement to roughly describe a dataset's requirement for temporal modeling. Here we plot the regression between the Sens-Index on UCF101 and Sens-Index on Something-Something in Figure 4–14. We use the regression coefficient to quantitatively describe the relative requirement for temporal modeling of the two datasets. In this case, we regard the UCF101 as the baseline whose requirement for temporal modeling is 1, and Something-Something is 2.650. This means the requirement for temporal modeling on Something-Something is 2.650 times higher than UCF101. We do not use the bias term of the regression expression, since the line usually stays close to (0, 0) because that's where Average Pooling will locate (whose Sens-Index is always 0). Using only the regression coefficient is also more interpretable (as n times higher).

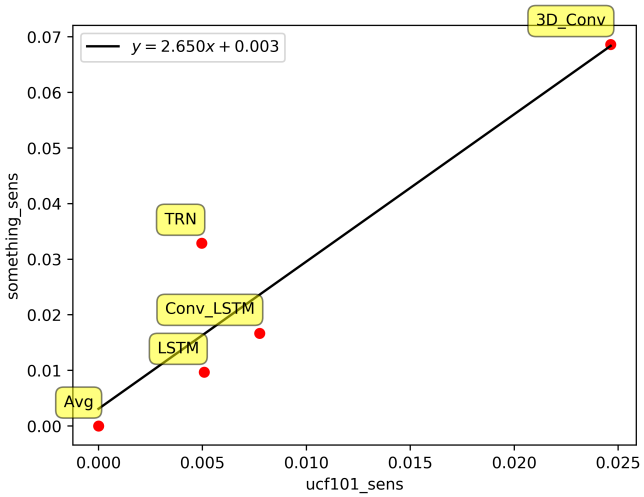

Figure 4–14 We conduct linear regression on Sens-Index on UCF101 and Something-Something to quantitatively describe their relative requirement for temporal modeling. In this thesis, we use the regression coefficient to relatively and quantitatively describe datasets, e.g. 1 for UCF101, and 2.650 for Something-Something.

4.7 Discussion and Conclusion

In this chapter, we evaluated several representative models that capture temporal dynamics in different ways, from no temporal modeling methods like Average Pooling to strong models like 3D Convolution. We conducted experiments on UCF101 dataset which does not require much temporal modeling to achieve high performance, and Something-Something dataset which on the contrary need very careful temporal modeling to get high accuracy. The methods were evaluated based on the protocols we proposed in the last chapter.

We conclude that, for UCF101 dataset, almost all deep learning based models tend to rely heavily on appearance to do the action recognition, leading to no obvious accuracy gap among different training versions or testing sets. On the contrary, the Improved Dense Trajectory model, which is a handcrafted approach, relies heavily on explicit picked trajectories in videos to do classification. Another exception is 3D Convolution model, which has very high Sens-Index and Inf-Index. However, its high sensitivity about time does not result in advanced absolute accuracy. We also found that though Temporal Relation network show weak temporal modeling on this dataset, the class-level statistics indicates its power for elastic temporal modeling, i.e. it can capture strong temporal dynamics when needed. We found that on UCF101 dataset, Optical Flow works because of its large gain from multi-segment consensus. The experimental results say the LSTM models cannot see the temporal dynamics on UCF101, or they choose to ignore temporal information for better performance.

For Something-Something dataset, all models except Average Pooling benefit significantly from their technique of temporal modeling. Among all of them, the 3D Convolution takes most temporal information into account. But from its poor performance on shuffled test set we can see it rely “too” heavily on temporal modeling, which may lead to weak robustness about temporal coherency. The general poor performance of

shuffled trained 3D Convolution indicates that Kinetics dataset is less powerful than ImageNet for 2D feature learning. The advanced results of shuffledly trained Optical Flow indicate its value actually comes from the motion information it contains, which is not shown in the UCF101 case. LSTM models show their power for sequence modeling on Something-Something, but compared with Temporal Modeling and 3D Convolution they are less efficient and effective. From the overall better performance of TRN and 3D Convolution, we believe when dealing with datasets containing complex temporal patterns, non-shared parameters on time dimension (unlike LSTMs) is very important for increasing the capability to capture temporal dynamics.

We found that Sens-Index is correlated to the performance on Something-Something dataset (that requires temporal modeling), but not on UCF101 (dataset that does not require temporal modeling). However, on UCF101, the Sens-Index is still meaningful for describing temporal modeling, and it is also correlated with Accuracy on Something-Something, even these two results are for different datasets. We also found that we can roughly describe a dataset's requirement for temporal modeling quantitatively by Sens-Index, using the regression coefficient calculated through Sens-Indeices on two datasets.

Chapter 5 Approximated Bilinear Fusion

5.1 Introduction

In this chapter, we introduce a method extended from bilinear models[55, 62-64] to enhance temporal modeling. The reason we think bilinear pooling would help temporal modeling is that bilinear pooling has been shown to be good at capturing fine-grained features in images[63], and we believe if we can discover finer relations on time dimension, the action recognition system could be improved. We first briefly introduce what is bilinear pooling in 5.2, and derive a general form of low-rank approximation of bilinear pooling in section 5.3. In section 5.4 we introduce a bilinear pooling method for action recognition using only RGB modality.

5.2 Bilinear Pooling

Bilinear pooling[62, 63] or second order pooling[65] is to conduct a pooling operation to produce a global description that summarizes local features in images, just like linear pooling operations such as average pooling and max pooling[65]. But different from linear pooling operations which employ first-order information of features, bilinear pooling takes the outer product of two features to exploit second-order information[66], which has been shown powerful in semantic segmentation[65], fine-grained image classification[63] and image action recognition[55]. For action recognition in videos, Diba et.al.[40] employed the idea of bilinear pooling to pool the fused feature maps of a video, but they didn't use it to capture temporal information. Wang et.al.[17] proposed Spatiotemporal compact bilinear (STCB) to fuse optical flow features and produce attention maps, which gives a good exploration of bilinear pooling on video modeling, but their method still requires optical flow to model temporal dynamics which is too expensive. In this section we introduce the basics of bilinear pooling, and present our methods in the following sections.

Bilinear pooling could be formulated as:

$$\boldsymbol{z} = \boldsymbol{w} Vec(\sum_{s \in S} \boldsymbol{\alpha}_s \boldsymbol{\beta}_s^T) \tag{5–1}$$

where $\boldsymbol{z} \in \mathbb{R}^K$ is the pooled output vector with dimension K. $Vec(\ )$ denotes the vectorization operator which reshape a matrix into a vector. $\boldsymbol{X} = \{\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, ..., \boldsymbol{\alpha}_S; \boldsymbol{\alpha}_s \in \mathbb{R}^C\}$ is a set of local descriptors with each containing C elements (channels), and could be a layer of feature maps produced by a CNN, in which S represents the spatial locations. $\boldsymbol{Y} = \{\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, ..., \boldsymbol{\beta}_S; \boldsymbol{\beta}_s \in \mathbb{R}^{C'}\}$ contains the same meaning as $\boldsymbol{X}$ but refers to a different tensor. $\boldsymbol{w} \in \mathbb{R}^{K \times CC'}$ represents the parameters to be learned. The bilinear pooling leverages outer product to model the interactions between two features thus can capture finer relation[63] which could be more discriminative than first-order pooling methods like average-pooling or max-pooling.

A problem of the bilinear pooling which prevents it from being widely used is the high dimensionality of the output representation, leading to a very large number of parameters to be learned, e.g. there are $K \times C \times C'$ parameters of $\boldsymbol{w}$ for bilinear pooling method compared with only $K \times C$ for average- or max-pooling with a

fully-connected layer. There are two popular ways to alleviate this problem. First is to employ the compact bilinear pooling[66] proposed by Gao et.al. which approximates the polynomial kernel of bilinear pooling with a low-dimensional projection algorithm. This method has been adopted in[17] and[40]. Another way is to approximate the bilinear pooling using low-rank approximation by factorizing the weights to be learned, which has been adopted in[55] and[54]. In the next section, we introduce the low-rank approximation we adopted in detail.

5.3 Low-Rank Approximation of Bilinear Pooling

In order to ease the high-dimension problem, we start by rewriting the definition of bilinear pooling (equation 5–1) formulation into a matrix multiplication with trace operations as:

$$z_k = tr(\boldsymbol{X}^T \boldsymbol{Y} \boldsymbol{W}_k^T) \tag{5–2}$$

where $\boldsymbol{W} \in \mathbb{R}^{K \times C \times C'}$ (here $\boldsymbol{W}_k \in \mathbb{R}^{C \times C'}$), $\boldsymbol{X} \in \mathbb{R}^{S \times C}$ and $\boldsymbol{Y} \in \mathbb{R}^{S \times C'}$. The $tr(\ )$ means the trace operation, which is the sum of the elements on the main diagonal (the diagonal from the upper left to the lower right). When using a matrix to represent parameters, it is natural to restrict the freedom of $\boldsymbol{W}$ by approximating it with low-rank approximation[55, 64]: $\boldsymbol{W}_{ijk} = \sum_{r=1}^{R} \boldsymbol{u}_{kr} \boldsymbol{a}_{ir} \boldsymbol{b}_{jr}$, where $\boldsymbol{u}_r \in \mathbb{R}^{K \times 1} \boldsymbol{a}_r \in \mathbb{R}^{C \times 1}, \boldsymbol{b}_r \in \mathbb{R}^{C' \times 1}$, and R is the number of ranks used for approximation. Then we have:

$$z_k = tr(\boldsymbol{X}^T \boldsymbol{Y} (\sum_{r=1}^{R} \boldsymbol{u}_{rk} \boldsymbol{a}_r \boldsymbol{b}_r^T)^T) \tag{5–3}$$

$$= tr(\sum_{r=1}^{R} \boldsymbol{u}_{rk} \boldsymbol{X}^T \boldsymbol{Y} \boldsymbol{b}_r \boldsymbol{a}_r^T) \tag{5–4}$$

$$= \sum_{r=1}^{R} \boldsymbol{u}_{rk} tr(\boldsymbol{X}^T \boldsymbol{Y} \boldsymbol{b}_r \boldsymbol{a}_r^T) \tag{5–5}$$

$$= \sum_{r=1}^{R} \boldsymbol{u}_{rk} tr(\boldsymbol{a}_r^T \boldsymbol{X}^T \boldsymbol{Y} \boldsymbol{b}_r) \tag{5–6}$$

$$= \sum_{r=1}^{R} \boldsymbol{u}_{rk} \boldsymbol{a}_r^T \boldsymbol{X}^T \boldsymbol{Y} \boldsymbol{b}_r \tag{5–7}$$

$$= \sum_{r=1}^{R} \boldsymbol{u}_{rk} (\boldsymbol{X} \boldsymbol{a}_r)^T (\boldsymbol{Y} \boldsymbol{b}_r) \tag{5–8}$$

Note that because indexed by r and k, $\boldsymbol{u}_{rk}$ is a scalar in this derivation. The equation 5–6 uses the property that the trace of multiplication of matrices is invariant under cyclic permutations. The equation 5–7 uses the fact that the variable in the trace operator is a scalar. If we drop the $\boldsymbol{u}_{rk}$ term, leaving parameter $\boldsymbol{a}$ to be class specific (indexed with k), and replace $\boldsymbol{Y}$ by $\boldsymbol{X}$, then this function is called on feature maps of a single image, and it converges to the implementation of image action recognition in[55]. As has been pointed out in[55], $\boldsymbol{X}\boldsymbol{a}_r$ and $\boldsymbol{Y}\boldsymbol{b}_r$ could be seen as two attention maps, and the output is the sum of the inner product of each pair of heatmaps for all ranks, and generating these attention maps could be efficiently implemented with convolution on GPUs.

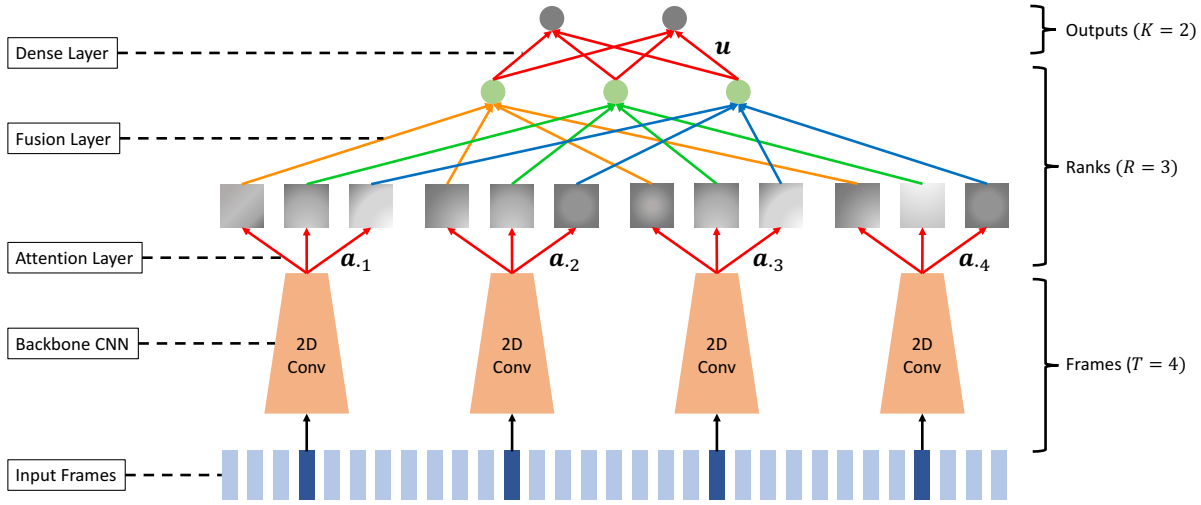

Figure 5–1 The illustration of proposed attention fusion method. Left annotations highlight the different components of our model. In this illustration, we sample 4 frames for processing. Parameters in Conv CNN are shared across all frames, while the ones in the attention layer are not, e.g. $\boldsymbol{a}_{\cdot t}$ represents parameters for each frame. Above the attention layer is the fusion layer, where the results are grouped by ranks, e.g. here 3 ranks are used for approximation. Above the fusion layer is the dense layer which could be used as the inputs to another layer or to output predictions. This layer has parameters u. This figure corresponds to the equation 5–10 or 5–11 or 5–12, where $\boldsymbol{X}_t$ represents the output feature maps of each backbone CNN. Note that besides backbone CNNs, only attention layer and dense layer contain learnable parameters (links colored in red).

5.4 Approximated Bilinear Fusion on 2D Feature Maps

We seek the possibility to exploit bilinear pooling to fuse 2D convolutional features on time dimension, in order to capture temporal patterns. However, it is not clear how to apply the bilinear pooling technique on multiple frames since it takes only two bunches of feature maps as input. In this thesis, we investigate three ways to fuse multiple frames.

- The first way is to extend the multiplication in bilinear pooling to multiple elements. Specifically, the eq. 5–8 can be rewritten as:

$$z_k = \sum_{r=1}^{R} u_{rk} \mathbf{1}^T ((\boldsymbol{X}\boldsymbol{a}_r) \odot (\boldsymbol{Y}\boldsymbol{b}_r)) \tag{5–9}$$

  where $\mathbf{1}$ is a vector of ones, and $\odot$ denotes element-wise product. Since both the input attention maps ($\boldsymbol{X}$ and $\boldsymbol{Y}$) now are computationally equal, the element-wise product of two attention heatmaps can be extended to maps from multiple frames, and then this could be seen as a multi-linear pooling along time axis. Assuming the 2D convolutions are operated individually on T video feature maps (extracted from T frames) and the feature maps are denoted by $\{\boldsymbol{X}_1, \boldsymbol{X}_2, ..., \boldsymbol{X}_T\}$, the output vector $\boldsymbol{z}$

could be written as:

$$z_k = \sum_{r=1}^{R} u_{rk} \mathbf{1}^T (\odot_{t=1}^{T} (X_t a_{rt})) \tag{5–10}$$

where $X_t a_{rt}$ denotes an attention map generated at time point t of rank r. Attention maps at all time steps of the same rank are element-wisely multiplied together.

- The second way is to get the bilinear pooling fusion of each neighbored pairs, and add them up to fuse multiple frames. This way keeps the original meaning of bilinear pooling, and could be formulated as:

$$z_k = \sum_{r=1}^{R} u_{rk} (\sum_{t=1}^{T-1} (X_t a_{rt})^T (X_{t+1} a_{r(t+1)})) \tag{5–11}$$

where all notations are of the same meaning as in eq. 5–10, but here we only do bilinear pooling on neighbored frame pairs.

- The third way is a little different from the second one. We consider the multiplication between neighbored pairs may result too much local temporal information which may be not very informative. We instead design another fusion method which multiply all other frames with the first frame rather than multiplying every neighbored pairs (see Figure 5–2(c) for an illustration). This implementation could be formulated as:

$$z_k = \sum_{r=1}^{R} u_{rk} (\sum_{t=2}^{T} (X_1 a_{r1})^T (X_t a_{rt})) \tag{5–12}$$

where all other elements are multiplied with the first one.

Note that for all three fusion approaches, the generated heatmaps are 2D maps and contain no temporal information, thus the temporal structure is instead captured by parameters a_{rt}, which should not be shared across T frames. Up to now, the overall architecture of our bilinear fusion model is clear: first we generate R

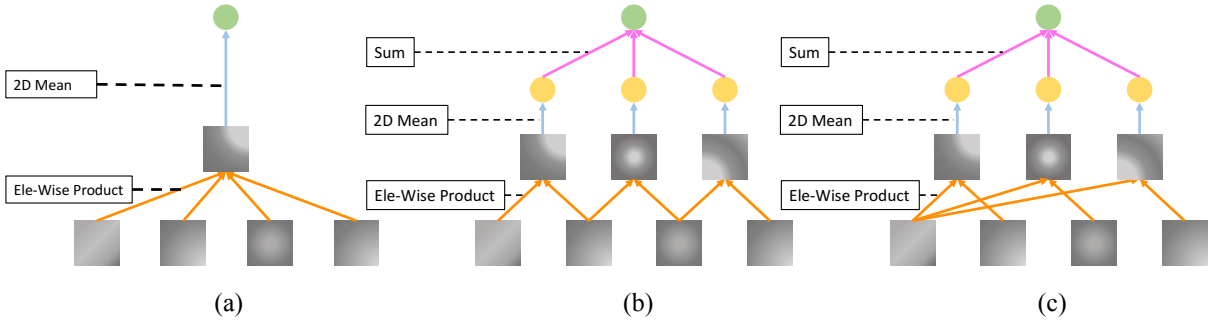

Figure 5–2 The illustration of three fusion approaches. (a) Using Element-Wise Product to fuse multiple frames. The corresponding equation is eq. 5–10. (b) Summing bilinear pooling of multiple neighbored pairs to fuse frames. The corresponding equation is eq. 5–11. (c) Same as (b) but the multiplications happen between every other frame and the first frame rather than every neighbored pairs.

(ranks of) attention maps using feature maps of each frame where the corresponding weights for each time step are not shared; second we fuse the maps at each time step using either the approaches we introduced earlier in which no parameters are learned; finally we add a multi-layer perceptron to output the activation scores, which could be used as the input of another layer or the final predictions. The whole model is trained end-to-end using standard backpropagation. An illustration of this model is in Figure 5–1 with Attention Layer, Fusion Layer, and Dense Layer annotated. The three fusion approaches are illustrated in Figure 5–2. This model conceptually simulates the temporal relation network[34] which does not share parameters on time dimension as LSTMs, increasing the capability to capture more usable temporal patterns in videos.

Although the architecture of our proposed model has been introduced, some details are still not determined and should be selected by experiments. We need to choose which fusion method works better. And there are some hyperparameters, such as the number of ranks used to approximate the bilinear pooling, the filter size used when generating attention maps. These should be searched to make this model work well. We also need to know if another classification layer should be used at the top to produce predictions, and its corresponding dropout rate. The correct activation function used after the attention maps, and the right number of frames to be sampled also remain unknown. These uncertainties will be solved in the next chapter.

Chapter 6 Experiments on Proposed Model

6.1 Introduction

In this chapter, we conduct experiments on our proposed bilinear fusion model introduced in the Chapter 5. We first introduce the configurations adopted in our experiments in Section 6.2, and we conduct parameter search in Section 6.3. Then we evaluate our proposed model with our protocols introduced in Section 6.4, in which we also compare ours with other methods evaluated. Finally our discussions and conclusions are presented in 6.5.

6.2 Configurations

As our model is proposed to exploit temporal information, we choose to do parameter search on Something-Something dataset, since the experiments have shown that this dataset requires temporal modeling to achieve high accuracy (see Chapter 4). Something-Something dataset consists of 108,499 videos across 174 classes, with videos' duration ranging from 2 to 6 seconds, and more detailed introduction of this dataset could be seen in Section 4.2.

Our backbone architecture is ResNet34 (same as the architecture used in Chapter 4) pretrained on ImageNet, with 96×96 as input size for faster convergence. New weights introduced by our fusion modules are initialized by uniform distribution, and biases are initialized as 0. Then the whole model is fine-tuned by backpropagation using mini-batch Stochastic Gradient Descent optimization, with Momentum set to be 0.9. The learning rate is initialized to be 0.001 and shrunk by 10 at epoch 15 and 25, and the model converges at epoch 30 (which is significantly faster than the models we compared earlier).

6.3 Parameter Search

6.3.1 Fusion Choice

First we need to determine which fusion approach (introduced in Section 5.4) to use in our model. Here we compare the accuracy results on Something-Something of the three fusion methods (namely *Product*, *Sum*

Table 6–1 Fusion comparison on Something-Something dataset.

Fusion Type	Top-1	Top-5
Product	0.83	4.44
Sum	23.97	50.08
Jump	24.72	50.98

Table 6–2 Find the number of working frames for *Product* on Something-Something dataset.

Number of Frames using *Product*	Top-1	Top-5
4	20.41	44.25
5	1.01	4.89
6	0.81	4.12

Table 6–3 Video Length search on Something-Something dataset.

Video Length	Top-1	Top-5
4	23.21	49.77
6	23.56	49.92
8	23.97	50.08
10	23.99	50.15

and *Jump*), while keeping other configurations fixed. The number of ranks is fixed to be 64, and we sample 8 frames as the input. The results are shown in Table 6–1.

From the results, we can see the *Jump* fusion method works best, and *Sum* performs a little bit poorer. We believe the skip connection makes the *Jump* more stable than the *Sum* way. Interestingly, the *Product* fusion approach does not work at all. We think this is caused by the cumulative product on too many frames, which makes the values explode or vanish (similar to vanilla recurrent neural networks). To validate this hypothesis, we apply *Product* to fewer frames in Table 6–2.

We can see the model does not work when the number of sampled frames exceeds 4. Therefore we can conclude that the *Product* fusion approach is not suitable for long-term action recognition.

6.3.2 Video Length

We then need to determine the video length used for inputs, i.e. how many frames should be sampled as input for a video instance. Too few frames may not be enough for correct classification since some classes on Something-Something are very challenging and require long term observation, while too many sampled frames could result in redundancy and make our model less efficient. The investigation results are shown in Table 6–3. For a test video to sample T frames, we evenly divide the whole video into T segments (same as in[10]), and pick the middle frame from each segment to form an input of our model (*Sum* fusion is used here).

As we can see, the performance increase is marginal when the number of frames exceed 6. Considering we are only using 64 ranks in this search experiment which many restrict the influence of video length, we use 8 frames as input in the following experiments.

Table 6–4 Number of ranks search on Something-Something dataset.

Number of Ranks	Top-1	Top-5
64	24.72	50.98
128	25.20	52.45
256	25.51	51.42

Table 6–5 Filter Size search on Something-Something dataset.

Filter Size	Top-1	Top-5
1	25.20	52.45
2	24.30	50.61
3	24.41	50.77

6.3.3 Number of Ranks

As we use the low-rank approximation to approximate the bilinear pooling (see Chapter 5 for details), we need to determine how many ranks should be used in our setup. Too many ranks will lead to tremendous parameters to learn which loses the efficiency of this model, and too few ranks may prevent our model from working effectively. The search results are shown in Table 6–4.

The results indicate the more ranks we use, the higher performance we reach. To keep the efficiency, in this thesis we do not use more ranks and keep 256 ranks as default.

6.3.4 Generating Attention Maps

As we have pointed out in Section 5.3, the term $\boldsymbol{X}\boldsymbol{a}$ could be seen as generating an attention map on $\boldsymbol{X}$ by convolution with kernel size 1 (parameter $\boldsymbol{a}$). But one may ask if this could be extended by modifying the filter size to consider more spatial information, though it is not suggested through the derivation. We conduct experiments to see if this modification can improve the performance in Table 6–5.

We now can see increasing the filter size could not boost the performance in our experiments. We believe this is because this modification breaks the mathematical soundness of bilinear pooling, thus this operation should be better seen as a dot product instead of convolution.

6.4 Evaluation

6.4.1 Accuracy

We now evaluate our model on UCF101 and Something-Something datasets. In this section, we fix the input size of our model to be 224×224, sample 8 frames for each video, and set the number of ranks to be 256.

Table 6–6 Comparison of top-1 accuracy between our model and other methods under different temporal conditions on Something-Something dataset. All results are reproduced by ourselves, therefore may not exactly match the values in the original papers. In the first row, "n", "r" and "s" denote input data with normal, reversed and shuffled frames, respectively. The letter on the left of "-" means training phase, and on the right means test phase. For instance, "n-r" means the model is trained with frames in normal order and tested in reversed order.

Method	Something-Something					
	n-n	n-r	n-s	s-n	s-r	s-s
Avg Pooling[10]	14.15	←	←	14.32	←	←
Temporal Relation[34]	29.58	9.81	15.01	16.61	16.69	16.73
Regular LSTM[8]	18.03	10.61	13.77	14.67	14.41	15.28
Conv LSTM[23]	19.87	10.04	12.86	15.64	14.73	16.81
Optical Flow[7]	19.85	17.18	17.65	17.99	17.96	18.13
3D-Res-34[12]	33.88	10.93	1.82	11.43	11.86	14.94
Ours	31.72	9.77	14.27	14.92	14.85	15.11

We use another MLP classifier with 256 hidden units and 0.5 dropout rate to prevent overfitting before the final prediction. We present the comparison tables on Something-Something (6–6) and UCF101 (Table 6–7) here again to compare our model with others.

Our model achieves the best accuracy among 2D based models on Something-Something. Honestly this is under our expectation since we are adjusting the parameters on this dataset and we tried to make our model perform as high as possible. But it is still amazing since we outperform the state-of-the-art model TRN (at the end of 2017) on Something-Something, in which the configurations are kept the same as in the original paper[34]. Note that TRN is still the best baseline on Moments in Time dataset[39], and we believe after more improvements we can also outperform it on other (larger) datasets. Note again that the accuracies shown in our tables are not state-of-the-art since we are using relatively shallow backbone CNNs and no cropping strategies are used for testing. We believe our model can boost quite a lot if all these strategies are taken into consideration. The evident gap between the performance under normal condition and abnormal conditions (trained or tested with reversed or shuffled frames) indicates the value of the temporal modeling in our model.

Surprisingly, the results on UCF101 also say our model achieves the best performance compared to all other deep methods under normal condition. Though this is far from state-of-the-art, considering we are not carefully adjusting parameters on UCF101, not using a strong backbone architecture, not using optical flow modality, not using any augmentation like 10-crop, and not pretraining our model on larger video datasets like Kinetics, this result is good and we believe it can achieve much higher or even state-of-the-art accuracy if we take some of the above modifications into account. Like most other methods, the performances of our model under any abnormal conditions (trained or tested with reversed or shuffled frames) are very close, indicating on UCF101, spatial information is still dominating temporal information, however the little improvement on

Table 6–7 Comparison of top-1 accuracy between our method and other methods under different temporal conditions on UCF101 dataset. All results are reproduced by ourselves, therefore may not exactly match the values in the original papers. In the first row, "n", "r" and "s" denote input data with normal, reversed and shuffled frames, respectively. The letter on the left of "-" means training phase, and on the right means test phase. For instance, "n-r" means the model is trained with frames in normal order and tested in reversed order.

Method	UCF101					
	n-n	n-r	n-s	s-n	s-r	s-s
Avg Pooling[10]	80.78	←	←	80.39	←	←
Temporal Relation[34]	82.10	78.77	80.04	80.07	79.99	80.20
Regular LSTM[8]	80.33	79.17	79.75	79.83	80.33	80.28
Conv-LSTM[23]	79.88	79.33	80.33	79.17	79.46	79.80
Optical Flow[7]	72.09	70.71	70.42	71.19	70.10	70.66
IDT[5]	82.13	-	8.30	27.76	-	67.22
3D-Res-34[12]	80.94	80.94	72.01	70.47	70.21	71.79
Ours	82.87	77.53	79.78	80.81	80.70	80.47

Table 6–8 Comparison of Inf-Index and Sens-Index between our model and other methods on UCF101 and Something-Something datasets. Inf-Index reflects how the temporal information influences the formation (learning) of a model, and Sens-Index measures how sensitive a normally trained model is about the temporal coherency.

Method	UCF101		Something-Something	
	Inf-Index	Sens-Index	Inf-Index	Sens-Index
Avg Pooling[10]	0.01412	0.0	0.00640	0.0
Temporal Relation[34]	0.01808	0.00496	0.03847	0.03289
Regular LSTM[8]	0.01686	0.00508	0.01810	0.00967
Conv LSTM[23]	0.01437	0.00775	0.01904	0.01668
3D-Res-34[12]	0.03098	0.02464	0.04284	0.06863
Ours	0.01956	0.01191	0.03000	0.02833

n-n still indicates the value of temporal modeling in our model.

6.4.2 Sens-Index and Inf-Index

The comparison of Sens-Index and Inf-Index between our model and others on both datasets are in Table 6–8. We can see the our model's Inf-Index is very high on UCF101, which is only lower than 3D Convolution, showing that temporal coherency is very important for training a good version of our model. Similarly,

Table 6–9 Complexity comparison among models. We show the number of parameters learned in each model (not considering backbone CNN since its shared by all methods), and the accuracies on both datasets in the last two columns. Note that the accuracy results are reproduced by ourselves thus may not be optimal, which may also be unfair to directly compare them together. But we can roughly say that our model is of the lowest complexity and are at least comparable to all of other models.

Models	Parameters	Top-1 Acc Something	Top-1 Acc UCF101
Regular LSTM	2.1M	18.03	80.33
Conv-LSTM	18.9M	19.87	79.88
TRN	2.2M	29.58	82.10
Raw Bilinear Model	469.9M	-	-
Ours	1.1M	31.72	82.87

Sens-Index is also high on UCF101, indicating even on dataset like UCF101 where temporal modeling is not required, our model is still very sensitive to temporal coherency. On Something-Something, our Inf-Index and Sens-Index are lower than 3D Convolution and Temporal Relation Network, indicating TRN is still better at exploiting temporal information, and there are still potential for temporal modeling in our model. But generally speaking, our model is novel model which is good at temporal modeling.

6.4.3 Complexity Analysis

In this section, we compare the number of parameters learned in evaluated models. Specifically, we compare our model with Regular LSTM, Conv-LSTM, TRN, and raw bilinear model without low-rank approximation. For simplicity, we omit the parameters in the backbone CNN since all models share the same architecture with the same number of parameters. The results are shown in Table 6–9.

The table says that our model is of the lowest complexity among all compared methods (2D-Conv based models), while achieving the highest performance. Note that all models are reproduced by ourselves therefore may not be optimal and may be unfair to directly compare together. But at least we can say our model is comparable to all other ones while keeping beautiful simplicity. We can also see the effectivity of low-rank approximation which reduces the parameters from 469.9M to 1.1M. Unfortunately in this thesis we do not train the raw bilinear model since it could be too expensive.

We also want to mention that our Approximated Bilinear Fusion model can be trained quickly which converges within 30 epochs while all other compared models need over 50 epochs.

6.5 Discussions and Conclusions

In this chapter, we first conduct exploration experiments on our proposed Approximated Bilinear Fusion model. We show that among the proposed fusion approaches (*Product*, *Sum*, and *Jump*), *Product* does not work when sampled frames exceed 4, and *Jump* is the best choice. We choose to use 8 frames for each video

as input to our model based on the search results. After comparison, we use 256 ranks to approximate the bilinear model because of the balance between complexity and effectivity. We also show that increasing the filter size to generate attention maps is not a good idea. We believe it is because increasing the filter size breaks the mathematical soundness of our method, and seeing it as 1×1 convolution is not appropriate (seeing as matrix multiplication may be better).

Then we evaluate our model on UCF101 and Something-Something datasets. We compare the accuracy, Sens-Index, and Inf-Index of our model with other evaluated methods in this thesis. The results show that our model performs best among all methods except 3D Convolution. Surprisingly, using only half of learnable parameters, our model can outperform the TRN model, which was a state-of-the-art on Something-Something at the end of the 2017, indicating our model can successfully exploit temporal information at least as well as TRN. Scores of Sens-Index and Inf-Index also say our can definitely benefit from temporal modeling, especially having higher temporal sensitivity on UCF101 dataset. The complexity comparison with other models shows that our model is of the lowest complexity and can converge very fast, while achieving very good performance on both datasets.

In the future, we plan to do more detailed exploration on more hyperparameter selection, and apply this approximated bilinear fusion to 3D convolution based models. We also want to train larger models on Kinetics or Moments in Time datasets since smaller datasets can benefit from pretraining on them. Additionally, we are going to increase interpretability of this model, since the inner representations like the generated attention maps could be used to visualize the extracted relations across multiple frames.

Chapter 7 Conclusions and Future Directions

7.1 Conclusions

In this thesis, we have focused on two main topics to present our research on temporal modeling in action recognition. First we asked the question how well previous methods work on temporal modeling, and second is how to design a novel method that can exploit temporal information well in action recognition. For the first question, we did a literature review on recently proposed models in Chapter 2, proposed an evaluation method to quantitatively describe a model's capability in temporal modeling in Chapter 3, and we evaluated a bunch of representative approaches in Chapter 4. For the second question, we consider to exploit finer temporal information by introducing our approximated bilinear fusion model in Chapter 5, and we presented the exploration study and evaluation experiments of our model in Chapter 6. Our work is going to be submitted to CVPR2019, when more works are done in the following months.

More detailed conclusions are presented as follows:

- **Chapter 3** proposes our temporal evaluation methods to quantitatively measure a model's sensitivity about temporal coherency, and how much the temporal coherency can influence a model's training, e.g. Accuracy, Sens-Index, and Inf-Index have been used. To calculate Sens-Index and Inf-Index, Root-Mean-Square deviation between prediction vectors has been used in our definition. The Sens-Index and Inf-Index do not require ground-truth labels to be revealed therefore may help enhance training in future works.
- **Chapter 4** evaluates representative methods reviewed in Chapter 2. From the experiments, we can conclude that for UCF101 dataset which does not require much temporal modeling to get high accuracy, many models tend to ignore temporal information during training to do classification. For Something-Something, most methods benefit from their capability in temporal modeling. Though achieving the highest performance, 3D Convolution tend to rely too much on temporal patterns, and seems to be less robust to temporal coherency than other models. Unlike in[33] where Optical Flow was thought to be more important for appearance invariance than motion capturing, we show that Optical Flow *can* definitely exploit motion information even when flow fields are not ordered. We also show that a model's accuracy on Something-Something is correlated with its sensitivity about time, while on UCF101 it does not hold. Besides, we show that a dataset's requirement for temporal modeling can be quantitatively roughly described using the Sens-Index of multiple models.
- **Chapter 5** introduces our proposed Approximated Bilinear Fusion model, which is a variant of bilinear pooling but with low-rank approximation. We show that this procedure could be achieved by convolution using 1×1 kernels, and the inner product of attention maps. Long term temporal patterns can be encoded using non-shared parameters across multiple frames, which can provide more capacity to capture temporal dynamics. Three fusion methods are proposed, though we show only the

Sum and *Jump* versions work well in the following experiments.

- **Chapter 6** presents our exploration study on the proposed model. We show that *Product* fusion method does not work when more than 4 frames are sampled for a video instance. Larger filter size could not enhance our model when producing the attention maps, and we believe it is due to that increasing filter size breaks the mathematical soundness. When evaluated on Something-Something and UCF101, our model can outperform or compete with all models we evaluated earlier, while containing significantly fewer parameters and converging much faster than them, showing the effectivity and efficiency of our model.

7.2 Future Directions

We present several directions of future works that can be extended from this thesis in this section. These potential directions are based on the conclusions we draw from the evaluation experiments about temporal modeling, and the potential improvements on our proposed model. We believe these directions can benefit from our contributions, and further contribute to the area of action recognition.

- **Boosting Performance by Considering Temporal Sensitivity:** We have shown in Chapter 4 that on datasets requiring temporal modeling, a model's accuracy is correlated with its sensitivity about time. Thus we can consider not only to do optimization based on accuracy, but also on temporal sensitivity, which can be implemented by multi-tasking. As the calculation of our proposed Sens-Index does not require revealing of labels, we can further employ more unlabeled data in the training, benefiting from semi-supervised learning. This setup encourages the model to take more temporal information into consideration when doing prediction, and can prevents the model depending too much on appearance classification. We believe some models like LSTMs can benefit greatly from it as they haven been shown less efficient in exploiting temporal information in our experiments.
- **Approximated Bilinear Fusion on 3D Feature Maps:** In this thesis, we introduce Approximated Bilinear Fusion on 2D Feature Maps, whose effectivity and efficiency have been validated. We may consider if this technique can be employed on 3D Convolutions. The value of this future direction comes from that unlike 2D feature maps, the 3D feature maps already contain temporal information, thus the attention maps could span more than 1 frame, capturing salience information of reasoning across time. This can increase the interpretability of 3D Convolutions, and also boost its performance since bilinear pooling is supposed to offer finer modeling on feature relations.
- **Interpreting Relations:** Another work we are going to do is to enhance our Approximated Bilinear Fusion model by revealing more interpretability. We plan to show what relations are important for identifying an example, what relations are corresponded to which activation maps, and how the interpretability can help debug or improve our models. We believe these properties are related to other tasks like video retrieval, video generation, and video based Visual Question Answering (VQA).
- **Temporal Action Detection:** As our work focuses on temporal modeling, we believe a future direction is to apply some of our conclusions to temporal action detection, since this task requires finer temporal exploitation to determine the location of actions in an untrimmed video. Our investigation on previous

methods can offer a hint of which techniques is more sensitive to temporal information, and our proposed model offer a potential solution to capture finer temporal dynamics.

Bibliography

[1] YouTube Statistics –2018. [J]. Https://merchdope.com/youtube-statistics/, 2018.

[2] UMAKANTHAN S. Human action recognition from video sequences[D/OL]. Queensland University of Technology, 2016. https://eprints.qut.edu.au/93749/.

[3] LAPTEV I. On Space-Time Interest Points[J/OL]. Int. J. Comput. Vision, 2005, 64(2-3): 107-123. http://dx.doi.org/10.1007/s11263-005-1838-7. DOI: 10.1007/s11263-005-1838-7. ISSN: 0920-5691.

[4] WANG H, KLÄSER A, SCHMID C, et al. Action recognition by dense trajectories[J]. CVPR 2011, 2011: 3169-3176.

[5] WANG H, SCHMID C. Action Recognition with Improved Trajectories[J]. 2013 IEEE International Conference on Computer Vision, 2013: 3551-3558.

[6] KARPATHY A, TODERICI G, SHETTY S, et al. Large-Scale Video Classification with Convolutional Neural Networks[J]. 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014: 1725-1732.

[7] SIMONYAN K, ZISSERMAN A. Two-Stream Convolutional Networks for Action Recognition in Videos[C]// NIPS. [S.l.]: [s.n.], 2014.

[8] DONAHUE J, HENDRICKS L A, GUADARRAMA S, et al. Long-Term Recurrent Convolutional Networks for Visual Recognition and Description[J]. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015: 2625-2634.

[9] TRAN D, BOURDEV L, FERGUS R, et al. Learning Spatiotemporal Features with 3D Convolutional Networks[C/OL]// Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV). ICCV '15. Washington, DC, USA: IEEE Computer Society, 2015: 4489-4497. ISBN: 978-1-4673-8391-2. http://dx.doi.org/10.1109/ICCV.2015.510. DOI: 10.1109/ICCV.2015.510.

[10] WANG L, XIONG Y, WANG Z, et al. Temporal Segment Networks: Towards Good Practices for Deep Action Recognition[C]// ECCV. [S.l.]: [s.n.], 2016.

[11] HERATH S, HARANDI M T, PORIKLI F M. Going deeper into action recognition: A survey[J]. Image Vision Comput., 2017, 60: 4-21.

[12] HARA K, KATAOKA H, SATOH Y. Can Spatiotemporal 3D CNNs Retrace the History of 2D CNNs and ImageNet?[J]. CoRR, 2017, abs/1711.09577.

[13] XIE S, SUN C, HUANG J, et al. Rethinking Spatiotemporal Feature Learning For Video Understanding[J]. CoRR, 2017, abs/1712.04851.

[14] FEICHTENHOFER C, PINZ A, ZISSERMAN A. Convolutional Two-Stream Network Fusion for Video Action Recognition[J]. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016: 1933-1941.

[15] FEICHTENHOFER C, PINZ A, WILDES R P. Spatiotemporal Residual Networks for Video Action Recognition[C]// NIPS. [S.l.]: [s.n.], 2016.

[16] FEICHTENHOFER C, PINZ A, WILDES R P. Spatiotemporal Multiplier Networks for Video Action Recognition[J]. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017: 7445-7454.

[17] WANG Y, LONG M, WANG J, et al. Spatiotemporal Pyramid Network for Video Action Recognition[J]. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017: 2097-2106.

[18] MA C Y, CHEN M H, KIRA Z, et al. TS-LSTM and Temporal-Inception: Exploiting Spatiotemporal Dynamics for Activity Recognition[J]. CoRR, 2017, abs/1703.10667.

[19] WANG L, XIONG Y, WANG Z, et al. Towards Good Practices for Very Deep Two-Stream ConvNets[J]. CoRR, 2015, abs/1507.02159.

[20] NG J Y H, HAUSKNECHT M J, VIJAYANARASIMHAN S, et al. Beyond short snippets: Deep networks for video classification[J]. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015: 4694-4702.

[21] SHARMA S, KIROS R, SALAKHUTDINOV R. Action Recognition using Visual Attention[J]. CoRR, 2015, abs/1511.04119.

[22] GOYAL R, KAHOU S E, MICHALSKI V, et al. The "Something Something" Video Database for Learning and Evaluating Visual Common Sense[C]// ICCV. [S.l.]: [s.n.], 2017.

[23] SHI X, CHEN Z, WANG H, et al. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting[C]// NIPS. [S.l.]: [s.n.], 2015.

[24] BALLAS N, YAO L, PAL C, et al. DELVING DEEPER INTO CONVOLUTIONAL NETWORKS FOR LEARNING VIDEO REPRESENTATIONS[G]// International Conference on Learning Representations (ICLR). [S.l.]: [s.n.], 2016.

[25] SUN L, JIA K, CHEN K, et al. Lattice Long Short-Term Memory for Human Action Recognition[J]. 2017 IEEE International Conference on Computer Vision (ICCV), 2017: 2166-2175.

[26] JI S, XU W, YANG M, et al. 3D Convolutional Neural Networks for Human Action Recognition[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2013, 35(1): 221-231. DOI: 10.1109/TPAMI.2012.59. ISSN: 0162-8828.

[27] SUN L, JIA K, YEUNG D Y, et al. Human Action Recognition Using Factorized Spatio-Temporal Convolutional Networks[J]. 2015 IEEE International Conference on Computer Vision (ICCV), 2015: 4597-4605.

[28] TRAN D, RAY J, SHOU Z, et al. ConvNet Architecture Search for Spatiotemporal Feature Learning[J]. CoRR, 2017, abs/1708.05038.

[29] TRAN D, WANG H, TORRESANI L, et al. A Closer Look at Spatiotemporal Convolutions for Action Recognition[J]. CoRR, 2017, abs/1711.11248.

[30] VAROL G, LAPTEV I, SCHMID C. Long-term Temporal Convolutions for Action Recognition[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017.

[31] CARREIRA J, ZISSERMAN A. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset[J]. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017: 4724-4733.

[32] SOOMRO K, ZAMIR A R, SHAH M. UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild[J]. CoRR, 2012, abs/1212.0402.

[33] SEVILLA-LARA L, LIAO Y, GÜNEY F, et al. On the Integration of Optical Flow and Action Recognition[J]. CoRR, 2017, abs/1712.08416.

[34] ZHOU B, ANDONIAN A, TORRALBA A. Temporal Relational Reasoning in Videos[J]. CoRR, 2017, abs/1711.08496.

[35] DUTA I C, IONESCU B, AIZAWA K, et al. Spatio-Temporal Vector of Locally Max Pooled Features for Action Recognition in Videos[J]. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017: 3205-3214.

[36] DU W, WANG Y, QIAO Y. Recurrent Spatial-Temporal Attention Network for Action Recognition in Videos[J]. IEEE Transactions on Image Processing, 2018, 27: 1347-1360.

[37] LI Z, GAVRILYUK K, GAVVES E, et al. VideoLSTM convolves, attends and flows for action recognition[J]. Computer Vision and Image Understanding, 2018, 166: 41-50.

[38] KUEHNE H, JHUANG H, GARROTE E, et al. HMDB: A large video database for human motion recognition[J]. 2011 International Conference on Computer Vision, 2011: 2556-2563.

[39] MONFORT M, ZHOU B, BARGAL S A, et al. Moments in Time Dataset: one million videos for event understanding[J]. CoRR, 2017, abs/1801.03150.

[40] DIBA A, SHARMA V, GOOL L V. Deep Temporal Linear Encoding Networks[J]. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017: 1541-1550.

[41] CHARIKAR M, CHEN K, FARACH-COLTON M. Finding Frequent Items in Data Streams[C/OL]// Proceedings of the 29th International Colloquium on Automata, Languages and Programming. ICALP '02. London, UK, UK: Springer-Verlag, 2002: 693-703. ISBN: 3-540-43864-5. http://dl.acm.org/citation.cfm?id=646255.684566.

[42] Twentybn jester dataset: a hand gesture dataset. [J]. Https://www.twentybn.com/datasets/jester, 2017.

[43] SIGURDSSON G A, VAROL G, WANG X, et al. Hollywood in Homes: Crowdsourcing Data Collection for Activity Understanding[C]// European Conference on Computer Vision. [S.l.]: [s.n.], 2016.

[44] SRIVASTAVA N, MANSIMOV E, SALAKHUTDINOV R. Unsupervised Learning of Video Representations using LSTMs[C]// ICML. [S.l.]: [s.n.], 2015.

[45] ZHENG Z, AN G, RUAN Q. Multi-Level Recurrent Residual Networks for Action Recognition[J]. ArXiv e-prints, 2017arXiv: 1711.08238 [cs.CV].

[46] LEI T, ZHANG Y, ARTZI Y. Training RNNs as Fast as CNNs[J]. CoRR, 2017, abs/1709.02755.

[47] CHO K, van MERRIËNBOER B, GÜLÇEHRE Ç, et al. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation[C/OL]// Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Doha, Qatar: Association for Computational Linguistics, 2014: 1724-1734. http://www.aclweb.org/anthology/D14-1179.

[48] NG J Y H, CHOI J, NEUMANN J, et al. ActionFlowNet: Learning Motion Representation for Action Recognition[J]. CoRR, 2016, abs/1612.03052.

[49] SUN S, KUANG Z, OUYANG W, et al. Optical Flow Guided Feature: A Fast and Robust Motion Representation for Video Action Recognition[J]. CoRR, 2017, abs/1711.11152.

[50] KAY W, CARREIRA J, SIMONYAN K, et al. The Kinetics Human Action Video Dataset[J]. CoRR, 2017, abs/1705.06950.

[51] HE K, ZHANG X, REN S, et al. Deep Residual Learning for Image Recognition[J]. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016: 770-778.

[52] DIBA A, FAYYAZ M, SHARMA V, et al. Temporal 3D ConvNets: New Architecture and Transfer Learning for Video Classification[J]. CoRR, 2017, abs/1711.08200.

[53] HUANG G, LIU Z, van der MAATEN L, et al. Densely Connected Convolutional Networks[J]. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017: 2261-2269.

[54] WANG L, LI W, LI W, et al. Appearance-and-Relation Networks for Video Classification[J]. 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[55] GIRDHAR R, RAMANAN D. Attentional Pooling for Action Recognition[C]// NIPS. [S.l.]: [s.n.], 2017.

[56] MISRA I, ZITNICK C L, HEBERT M. Shuffle and Learn: Unsupervised Learning Using Temporal Order Verification[C]// ECCV. [S.l.]: [s.n.], 2016.

[57] WANG X, FARHADI A, GUPTA A. Actions Transformations[J]. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016: 2658-2667.

[58] PICKUP L C, PAN Z, WEI D, et al. Seeing the Arrow of Time[J]. 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014: 2043-2050.

[59] ZOLFAGHARI M, SINGH K, BROX T. ECO: Efficient Convolutional Network for Online Video Understanding[J]. ArXiv e-prints, 2018arXiv: 1804.09066 [cs.CV].

[60] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. ImageNet Classification with Deep Convolutional Neural Networks[G/OL]// Advances in Neural Information Processing Systems 25. Ed. by PEREIRA F, BURGES C J C, BOTTOU L, et al. [S.l.]: Curran Associates, Inc., 2012: 1097-1105. http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[61] SIMONYAN K, ZISSERMAN A. Very Deep Convolutional Networks for Large-Scale Image Recognition[C]// International Conference on Learning Representations. [S.l.]: [s.n.], 2015.

[62] TENENBAUM J B, FREEMAN W T. Separating Style and Content with Bilinear Models[J]. Neural computation, 2000, 12 6: 1247-83.

[63] LIN T Y, ROYCHOWDHURY A, MAJI S. Bilinear CNNs for Fine-grained Visual Recognition[C]//. [S.l.]: [s.n.], 2017.

[64] PIRSIAVASH H, RAMANAN D, FOWLKES C C. Bilinear classifiers for visual recognition[C]// NIPS. [S.l.]: [s.n.], 2009.

[65] CARREIRA J, CASEIRO R, BATISTA J P, et al. Semantic Segmentation with Second-Order Pooling[C]// ECCV. [S.l.]: [s.n.], 2012.

[66] GAO Y, BEIJBOM O, ZHANG N, et al. Compact Bilinear Pooling[J]. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016: 317-326.

Acknowledgements

I would like to thank Professor Dacheng Tao for providing the project of action recognition, which is such an interesting and promising research direction that investigating the state-of-the-art algorithms and exploring new ideas are really interesting.

I would like to thank Professor Cewu Lu for his patient instructions, and encouraging me to do detailed investigation and evaluation about temporal modeling in action recognition. Without these efforts, I would never really know detailed weaknesses of the existing approaches, and I would never gain so much experience in implementation.

I would like to thank Professor Chang Xu for his highly constructive and inspiring discussions, as well as the support of providing powerful GPUs and a remote server which are essential for the experiments in this thesis.

I would like to thank Xiyu Yu and Rong Li for providing maintenance of the hardwares. Without their support, the progress in this thesis would be significantly delayed and no results can be obtained.

I would like to thank my friends, roommates, and my family for their encouragement and help in this semester.