

國立臺灣大學電機資訊學院電機工程學研究所

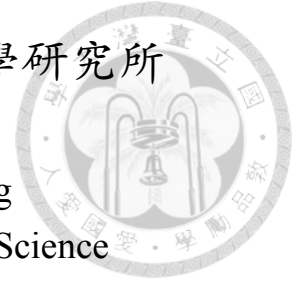
碩士論文

Graduate Institute of Electrical Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis



以全域性空間時間特徵輔以序列式生成機制完成多工之  
動作辨識及產生光流影像

Using Global Spatiotemporal Features with  
Sequentially Pooling Mechanism for Multi-tasking of  
Action Recognition and Optical Flow Estimation

葉佐新

Tso-Hsin, Yeh

指導教授：傅立成博士

Advisor: Li-Chen Fu, Ph.D.

中華民國 107 年 7 月

July, 2018



## 致謝

碩班的這兩年，我對學術研究的態度、分析和解決問題的能力有著十足的成長，首先要感謝的就是指導老師傅立成教授，老師對於學術研究的熱誠和追求卓越的堅持是我畢生仿效的對象，除此之外，老師在待人處事上也提點了我許多，這一些都是受用無窮的人生經驗與智慧，謝謝老師。

另外，我也感謝實驗室 208 的所有同學以及學弟們，謝謝你們跟我一起奮戰努力，同心協力一起度過碩二的艱苦生活，也感謝你們跟著我一起嬉鬧、打球以及從事娛樂活動，尤其是 vision 組的子俊、侑寰還有學弟宇閎，感謝你們陪著一起經歷 demo 這充實又不堪回首的時光。

最後感謝我的家人以及女朋友的支持與陪伴，謝謝你們！



## 中文摘要

在本篇中，我們提出了一個嶄新的深度學習架構來做動作辨識，近期的深度學習研究議題中，動作辨識是一個越來越重要的領域，深度學習方法已經被廣泛地運用且有能力產生泛化的模型，目前大部分存在的方法不是使用 Two-Stream，就是使用 3D ConvNet 的方法，前者使用了單張彩色影像與多張疊在一起的光流影像當作架構的輸入，而後者則是將輸入多張疊在一起的彩色影像，但會花較大時間及記憶體上的代價。本篇提出的 ResFlow 使用一個光流的數據庫以得到預處理的模型，並且在動作的數據庫上作微調處理，此模型可視為一個提取高維度特徵的模組來做動作辨識。在第一階段中，使用光流數據庫當作一個預先學習的基礎，整合空間時間的特徵透過自動編碼機得架構會從中間的高維度空間中被提取出來，在微調階段中，透過影片中分解出的影像可以得到一組區域性整合空間時間的特徵，並且利用設計的序列式機制，可以得到每一個區域性整合空間時間特徵的信心分數，而利用這個信心分數可以有效率地得到全域性整合空間時間的特徵，而這全域性整合空間時間的特徵可被拿來做動作辨識。

關鍵字：動作辨識、光流、序列式機制



# Abstract

In this thesis, we propose a brand-new architecture for action recognition. Recently, action recognition has been a rising issue in computer vision field. Deep learning method has been widely-used and is capable of generating generic model. Most of existing methods use either two-stream, RGB and stacking optical flow as inputs, or C3D, concatenating several RGB images as input, which cost much prices on time and memory. SSAnet, the proposed method, is pre-trained by optical flow dataset, FlyingChairs, and fine-tuned with action dataset, UCF101 and HMDB51, as a high level feature extractor. With optical flow pre-trained in first stage, spatiotemporal features are encoded in the latent high dimensional space in the middle of autoencoder architecture. In fine-tuning stage, the extracted spatiotemporal features from a set of frames from a video clip are given confidence scores by a designed Sequential Mechanism. This Sequential Mechanism takes ordered feature from the feature set as input and gives a confidence score to each feature to aggregate sequential features into a condensed feature which is leveraged for action recognition. This kind of design use only RGB images as input but with temporal information encoded, pre-trained by optical flow, and sequentially aggregate local spatiotemporal features into global spatiotemporal features in high efficiency for action recognition.

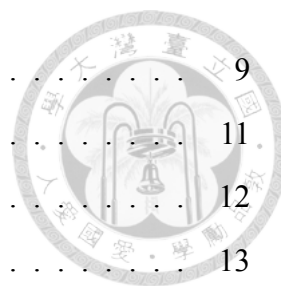
Keywords: action recognition, optical flow, sequential mechanism





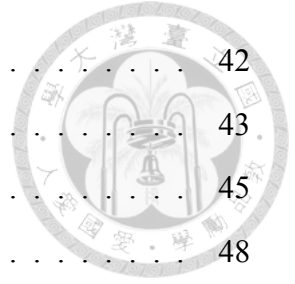
# Contents

<b>致謝</b>	<b>i</b>
<b>中文摘要</b>	<b>ii</b>
<b>Abstract</b>	<b>iii</b>
<b>Contents</b>	<b>iv</b>
<b>List of Figures</b>	<b>vii</b>
<b>List of Tables</b>	<b>x</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Literature Review . . . . .	2
1.2.1 Action Understanding . . . . .	2
1.2.2 Action Recognition . . . . .	3
1.2.3 Applications based on Convolution Neural Network . . . . .	4
1.3 Challenge . . . . .	4
1.4 Contribution . . . . .	5
1.5 Thesis Organization . . . . .	6
<b>2 Preliminary</b>	<b>7</b>
2.1 Optical Flow . . . . .	7
2.1.1 Traditional Methods . . . . .	8



2.1.2	Autoencoders . . . . .	9
2.2	Action Recognition . . . . .	11
2.2.1	Two-Stream . . . . .	12
2.2.2	3D ConvNet . . . . .	13
2.3	Joint Learning . . . . .	16
2.3.1	Object and Action . . . . .	16
2.3.2	Optical Flow and Action . . . . .	17
<b>3</b>	<b>Optical Flow Estimation</b> . . . . .	<b>19</b>
3.1	Design Concept . . . . .	19
3.2	Architecture . . . . .	21
3.2.1	Encoder . . . . .	21
3.2.2	Decoder . . . . .	22
3.3	Refinement Network . . . . .	23
3.4	Remarks . . . . .	25
<b>4</b>	<b>Action Recognition</b> . . . . .	<b>26</b>
4.1	Optical Flow to Action Recognition . . . . .	26
4.2	ResFlow . . . . .	27
4.3	Sequentially Pooling Mechanism . . . . .	29
4.4	Implementation Details. . . . .	32
<b>5</b>	<b>Experiment</b> . . . . .	<b>34</b>
5.1	Optical Flow Dataset . . . . .	34
5.1.1	FlyingChairs Dataset . . . . .	35
5.1.2	Sintel Dataset . . . . .	37
5.2	Action Recognition Dataset . . . . .	38
5.2.1	UCF101 Dataset . . . . .	38
5.2.2	HMDB51 Dataset . . . . .	41
5.2.3	Discussion . . . . .	41
5.3	Optical Flow Estimation . . . . .	41

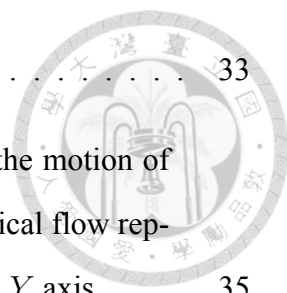
5.3.1	Comparison . . . . .	42
5.3.2	Refinement . . . . .	43
5.3.3	Sintel dataset . . . . .	45
5.3.4	Remark . . . . .	48
5.4	Action Recognition . . . . .	48
5.4.1	Spatiotemporal Feature . . . . .	49
5.4.2	Comparison . . . . .	50
5.4.3	Multitasking . . . . .	51
5.4.4	Remark . . . . .	52
<b>6</b>	<b>Conclusion</b>	<b>56</b>
	<b>Reference</b>	<b>57</b>



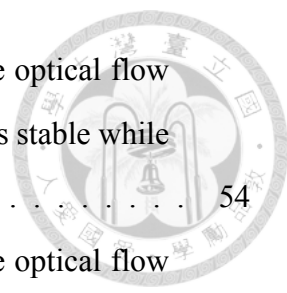


# List of Figures

2.1	<b>Optical Flow.</b> From left to right, they are the first image, the second image, and the optical flow estimation. . . . .	8
2.2	<b>Optical Flow Autoencoder.</b> Traditional autoencoder using deep learning architecture. . . . .	9
2.3	<b>Two Stream.</b> Two stream architecture include two stream, spatial stream and temporal stream. . . . .	11
2.4	<b>3D ConvNet.</b> 3D ConvNets architecture. . . . .	14
2.5	<b>Joint Learning of Object Detection and Action Recognition.</b> Joint learning of object detection and action recognition is described in the figure.	15
2.6	<b>Joint Learning of Optical Flow and Action.</b> Joint learning of optical flow estimation and action recognition. . . . .	17
3.1	<b>Autoencoder Architecture</b> Architecture of our designed autoencoder. . .	20
3.2	<b>Encoder (a)</b> Architecture of encoder. . . . .	21
3.3	<b>Decoder (a)</b> Architecture of Decoder. . . . .	22
3.4	<b>Optical Flow Refinement</b> Process flow of optical flow refinement. . . .	23
4.1	<b>Overall Architecture.</b> The overall Architecture which contains both optical flow estimation task and action recognition task. . . . .	27
4.2	<b>Aggregation of Spatiotemporal Features.</b> Process of aggregating spatiotemporal features for action recognition. . . . .	28
4.3	<b>Sequentially Pooling Mechanism.</b> Process flow of Sequentially Pooling Mechanism at time $t$ . . . . .	30



4.4	<b>Details of SPM.</b> This is a simplified visualization of SPM. . . . .	33
5.1	<b>Optical Flow.</b> Optical flow is calculated from evaluating the motion of first image and second image. Based on the first image, optical flow represents the moving displacement of the pixel along $X$ - and $Y$ axis. . . . .	35
5.2	<b>FlyingChairs Dataset Visualization.</b> We visualize the FlyingChairs dataset which is an optical flow dataset. From left to right, there are the first image, second image, and corresponding optical flow ground-truth in each row. . . . .	36
5.3	<b>Sintel dataset.</b> We visualize Sintel dataset which is a optical flow dataset. Each row represents a sequence of first image, second image, and corresponding optical flow from left to right. . . . .	37
5.4	<b>UCF101.</b> UCF101 dataset is an action dataset which contains 101 action categories with 13320 total action video clips. . . . .	39
5.5	<b>HMDB51.</b> HMDB51 dataset is an action dataset which contains 51 action categories with 6766 total action video clips. . . . .	40
5.6	<b>Optical Flow Refinement Visualization.</b> The progress of optical flow refinement are shown as the figure. . . . .	44
5.7	<b>FlyingChairs Visualization.</b> We visualize the optical flow estimation of our proposed ResFlow v2 on FlyingChairs dataset. . . . .	46
5.8	<b>Visualization of Sintel Optical Flow Estimation.</b> The images are the optical flow estimation on Sintel dataset without finetuning but pre-train on FlyingChairs. . . . .	47
5.9	<b>Visualization of Sintel Optical Flow Estimation.</b> The images are the optical flow estimation on Sintel dataset and finetuning on Sintel dataset. . . . .	47
5.10	<b>Optical Flow Refinement Visualization.</b> We visualize the optical flow estimation of an action, <i>playingviolin</i> , in the figure. The camera is stable while only the human and the violin is moving. . . . .	53



5.11 **Optical Flow Refinement Visualization.** We visualize the optical flow estimation of an action, *battling*, in the figure. The camera is stable while only the human is moving. . . . . 54

5.12 **Optical Flow Refinement Visualization.** We visualize the optical flow estimation of an action, *throwingball*, in the figure. The camera is moving which cause camera motions while the human is moving as well. . . . 55



# List of Tables

5.1	<b>Optical Flow Estimation Comparison.</b> We evaluate the performance of optical flow estimation of our proposed ResFlow v1 and ResFlow v2 on FlyingChairs dataset and compare them with the state-of-the-art FlownetS, FlownetC, and SpyNet. . . . .	43
5.2	<b>Optical Flow Refinement Evaluation.</b> We evaluate the performance of optical flow estimation of our proposed ResFlow v1 and ResFlow v2 on FlyingChairs dataset at all stages. ResFlow v2 obviously outperforms ResFlow v1 due to the <i>EPE</i> is smaller. . . . .	45
5.3	<b>Spatiotemporal Features Impact.</b> We directly use ResFlow which is pretrained on optical flow dataset to predict action recognition in condition of fixing the convolution layers. We evaluate the performance of ResFlow on UCF101 dataset as well as HMDB51 dataset and compare ResFlow with the state-of-the-art which are trained from scratch. . . . .	49
5.4	<b>Action Recognition Evaluation.</b> We finetune our ResFlow thoroughly, which is pretrained on optical flow dataset, on UCF101 dataset. We evaluate the performance of ResFlow on UCF101 dataset and compare ResFlow with the state-of-the-art. . . . .	50
5.5	<b>Multitasking comparison.</b> Multitasking comparison on optical flow estimation and action recognition. . . . .	51



# Chapter 1

## Introduction

In this chapter, we first describe our motivation in details in Section 1.1. An overview of action recognition is illustrated in Section 1.2. Challenges and contributions of this work is elaborated in Section 1.3 and Section 1.4, respectively, and the organization of this thesis is listed in Section 1.5.

### 1.1 Motivation

Action recognition is a popular topic in recent years and it is still a very challenging task with high computational cost in computer vision field. As a matter of fact, action recognition is crucial for our daily life, e.g., it will help a lot for families taking care of elderly people due to the fact that people can understand what happened to the elders and if some accidents occurred, they can access the information right away with the surveillance system. Also, some actions are unusual in certain circumstance, so the action recognition system can help to alarm the facilities to watch certain area and prevent the occurrence of events happen in advance.

Deep learning has been widely used to solve problems in various fields, including action recognition, object recognition, etc. Moreover, models generated by deep learning methods are more generic and stable compared to traditional methods. Despite the fact that the extracted features by traditional methods has more physical semantic meanings, that by deep learning methods has much more flexibility to learn some abstract features. The



deeper the architecture is, the more abstract the extracted features will be. Consequently, most researchers use Convolutional Neural Network (CNN) to extract spatial and temporal features to predict action recognition.

From our point of view, recent researches on action recognition utilize CNN to extract the spatial and temporal features and pass these features through a designed mechanism so as to aggregate these local features into global features. Finally, with a fully connected layer, action recognition result is predicted. Therefore, designing a suitable and robust mechanism is another important point to be focused on.

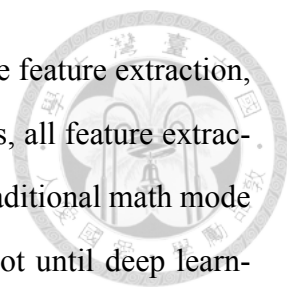
To sum up, we aim to design a CNN architecture to extract spatial and temporal features and a mechanism to aggregate these features to predict action class. In the thesis, we propose a network structure named ResFlow which generates both optical flow estimation and action recognition result with a well-designed CNN architecture and a novel mechanism named Sequentially Pooling Mechanism.

## 1.2 Literature Review

In Section 1.2.1 and Section 1.2.2, we give a brief introduction of action understanding and action recognition, respectively. Applications based on Convolution Neural Networks are introduced in Section 1.2.3.

### 1.2.1 Action Understanding

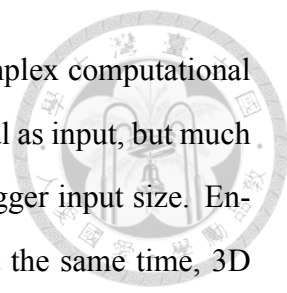
Basically, deep learning methods via Convolution Neural Network (CNN) dominates traditional machine learning methods. However, traditional methods are still vital to improve performance, and action recognition results are on the basis of the extracted features generated by these methods. STIP in [1] is one of the widely-used method to improve action recognition accuracy. Noticeably, action recognition results of traditional methods are complemented to that of deep learning methods. To sum up, the design of architecture is extremely important while traditional methods can be useful auxiliary tools to enhance performance.



Normally, the pipeline can be separated into three parts, which are feature extraction, spatial and temporal features encoding, and classification. Nowadays, all feature extraction methods can be organized into two main categories, which are traditional math mode and deep learning Convolution Neural Network (CNN) features. Not until deep learning starting to be popular, the traditional method had been the main-stream of predicting optical flow. Nevertheless, more and more researchers devoted into investigating action recognition via deep learning methods in recent years.

### 1.2.2 Action Recognition

Apart from action detection or action segmentation, action recognition is attempted to analyze which action occurred in a video clip, while action detection needs additional work to predict not only what but also where the action occurred. Action segmentation is defined as detecting when the action happened. In this thesis, we aims at action recognition as our final goal. Generally, there are two main categories to solve action recognition problem via CNN, which are two-stream, [2–9], and 3D ConvNet, [10–12]. Two-stream architecture, on the one hand, takes a RGB image and stacking optical flows as input separately. Theoretically, they extract spatial features through the network with RGB image as input while the other stream extracts temporal features. Usually, they fuse spatial information from RGB stream and temporal information from optical flow stream into spatiotemporal features. Objectively, speaking two-stream architecture method outperforms the other one, but optical flows need to be calculated beforehand in order to be fed into two-stream network. In other words, they consider RGB stream as extracting spatial information and flow stream as extracting temporal information, respectively. In general, there are two thoughts to deal with these features, one is focus on integrating extracted features and the other one is designing a mechanism to predict accurate action recognition. For instance, integrating spatial and temporal feature into spatiotemporal feature among the convolution layers includes insertion, concatenating, adding, etc. Contrarily, by utilizing the design mechanism, spatiotemporal information is generated from spatial and temporal information to predict action recognition, such as [2], [4], [6], [7]. On the other hand, 3D



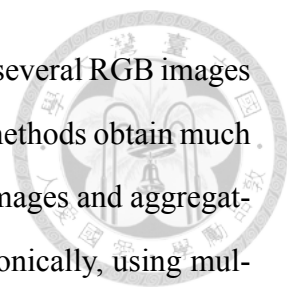
ConvNet stacks several RGB images together as input. With no complex computational preprocessing, 3D ConvNet only stacks RGB images in a time interval as input, but much more memory and time is required for 3D convolution due to the bigger input size. Encoding both spatial and temporal information via 3D convolution at the same time, 3D ConvNet still has its limits due to the fact that the vital information may vary in each time interval while 3D ConvNet treats them as the same since the same 3D convolution kernel issued. In 3D ConvNets, there is no exact differences dealing with important and useless information, but extract the spatiotemporal feature in a fixed pattern without flexibility.

### **1.2.3 Applications based on Convolution Neural Network**

Applications based on Convolution Neural Networks are varied, e.g. object classification, object detection, action recognition, action detection, image super-resolution, etc. Recently, as the rising of popularity in deep learning, more and more applications are on the basis of CNN to extract features since it is easier to implement only if we can access sufficient data. Moreover, the trained model is much more generic and it can be further utilized in other fields.

## **1.3 Challenge**

Nevertheless, there is still no a certain conclusion whether two stream approaches dominate 3D ConvNet approaches or vice versa. Now that many studies have shown the optical flow has strong representation for temporal information, why shall not we train an autoencoder with multiple pairs of RGB images as input and optical flows as output? Spatiotemporal features will be encoded due to the characteristic of autoencoder; otherwise, optical flow should never been decoded. Under this assumption, we may be able to leverage the spaiotemporal features from each time step for action recognition. However, unlike 3D ConvNet that stacks several RGB frames together at each time step to get longer period of temporal information, it is challenging to use only multiple pairs of RGB frames to predict action recognition. That is to say, two stream architecture normally has



more than one optical flow stacking and 3D ConvNet architecture has several RGB images stacking together which means that at each time step both of the two methods obtain much more temporal information rather than using multiple pairs of RGB images and aggregating the local spatiotemporal features in higher dimension domain. Ironically, using multiple pairs of RGB frames from each time step rather than stacking RGB frames together is much more reasonable because each time interval has a unique local spatiotemporal information which should be treated individually, and the stacking optical flows include the temporal information afterward which is not applicable in real-world. Still, designing a good mechanism to aggregate these local spatiotemporal features remains difficulties.

From the perspective of [13], it is feasible to combine a two stream architecture into the 3D ConvNet way. Via training an autoencoder with optical flow, it is reasonable to get spatiotemporal information which is important for action recognition. Although local spatiotemporal feature in a time interval has been encoded, it is still challenging to aggregate ordered local spatiotemporal features generated at each time step of a video clip. Furthermore, due to the fact that we only consider a pair of RGB images as input, each local spatiotemporal feature represents the information of previous frame and current frame. Therefore, how to aggregate these local spatiotemporal features sequentially in temporal order by a well-designed mechanism is another challenging issue.

## 1.4 Contribution

ResFlow, the proposed method, is capable of generating spatiotemporal features without feeding optical flows as input into network and can aggregate local spatiotemporal features into global spatiotemporal features by a well-designed Sequential Mechanism. Training an autoencoder with multiple pairs of RGB images as input and optical flow as output, ResFlow is able to obtain local spatiotemporal features from each image pair. Despite the fact that local spatiotemporal features are generated, there are still some issues. ResFlow consider the spatiotemporal feature in each time step as local spatiotemporal feature which is fed to a Sequentially Pooling Mechanism, *SPM*, so as to aggregate these local spatiotemporal features into a global spatiotemporal feature. By leveraging *SPM*

which sequentially gives a confidence score to each local spatiotemporal feature, global spatiotemporal feature has strong strength in predicting action recognition which encodes sequentially temporal spatiotemporal information other than ignoring temporal ordered spatiotemporal feature, i.e., conduct averaging, or max-pooling on local spatiotemporal features as global spatiotemporal feature.

There are several special features of the proposed system in this thesis. They are listed as below:

1. ResFlow is able to generate optical flow estimation and action recognition results simultaneously by multi-tasking;
2. ResFlow uses only RGB images to predict action recognition as well as optical flows in real-time efficiently;
3. A smart yet simple way is proposed to extract spatiotemporal features via an autoencoder.
4. Local spatiotemporal features are sequentially aggregated into global spatiotemporal features with a specially designed Sequentially Pooling Mechanism and the mechanism can be feasible for a number of applications.

## **1.5 Thesis Organization**

The organization is listed as follow. In Chapter 2, we introduce the concept of optical flow and algorithms to estimate optical flow. In Chapter 3, we introduce our designed autoencoder for estimating optical flow, and our overall ResFlow is elaborated in details in Chapter 4. Experimental results and comparison are shown in Chapter 5. Finally, this thesis is concluded in Chapter 6.



## Chapter 2

# Preliminary

In this chapter, we first describe what optical flow is and how optical flow is generated from both the traditional methods and the deep learning methods in Section 2.1. Action recognition is illustrated in Section 2.2 including classical deep learning methods for action recognition, two stream and 3D ConvNet, in Section 2.2.1 and Section 2.2.2, respectively. Finally, joint learning is elaborated in Section 2.3 with two design related to action recognition which will be introduced in Section 2.3.1 and Section 2.3.2.

## 2.1 Optical Flow

Optical flow is a crucial factor for predicting action recognition. Optical flow is defined as the pattern of apparent motion of objects, edges, and surfaces in a scene which is caused by the relative motion between an observer and a scene, as illustrated in [14], [15]. Also, it is used to describe the visual motion, e.g., an object is in the middle-left of a static scene at first, but it moves to middle-right of the same scene at next time step. Then, optical flow stores two channels value which are moving displacement along  $X$  and  $Y$  axis, respectively, denoted as  $u$  and  $v$ . Therefore, optical flow has been widely used in the field of action related works. Examples of optical flow can be seen in Fig. 2.1. Literally,  $image1$  and  $image2$  are the sequenced images at time  $t$  and time  $t + 1$ , respectively. On the right-hand side, the optical flow corresponding to these two images is shown. Since optical flow is based on first image to calculate the displacement of each pixel along  $X$

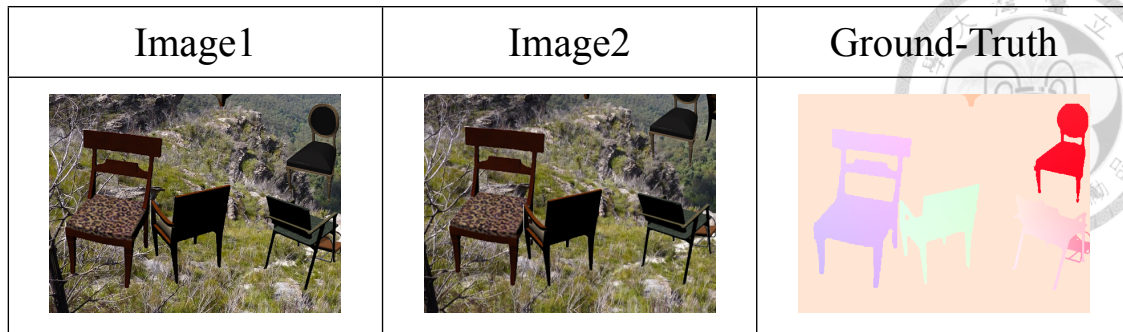


Figure 2.1: **Optical Flow.** From left to right, they are the first image, the second image, and the optical flow estimation.

and  $Y$  axis, the position of moving objects are obviously the same as where they belongs to *image1*. The colored optical flow are based on a color wheel to be generated due to the fact that optical flow has only two channels originally.

First, traditional methods to calculate optical flow are introduced in Section 2.1.1. In Section 2.1.2, we illustrate features and recent works using deep learning methods to generate optical flow in details.

### 2.1.1 Traditional Methods

Lucas-Kanade, [16], provides a naive method to calculate optical flow in year 2,000. The proposed algorithm is based on the least square criterion in a small sub-area which represents all pixels in the image to calculate optical flow between two image under the condition of assuming optical flow in a local neighborhood is essentially constant. Due to only considering a set of pixels, the generated optical flow is sparse. Oppositely, Farneback, [17], provides another classical method for computing dense optical flow which considers all pixels in an image. Viewing an image as two dimensional signal, Farneback utilizes binomial equation to calculate dense optical flow. Above methods are based on mathematical equations, but has some drawbacks to consider all kinds of condition into account. Therefore, some researchers start focusing on deep learning to solve this problem.

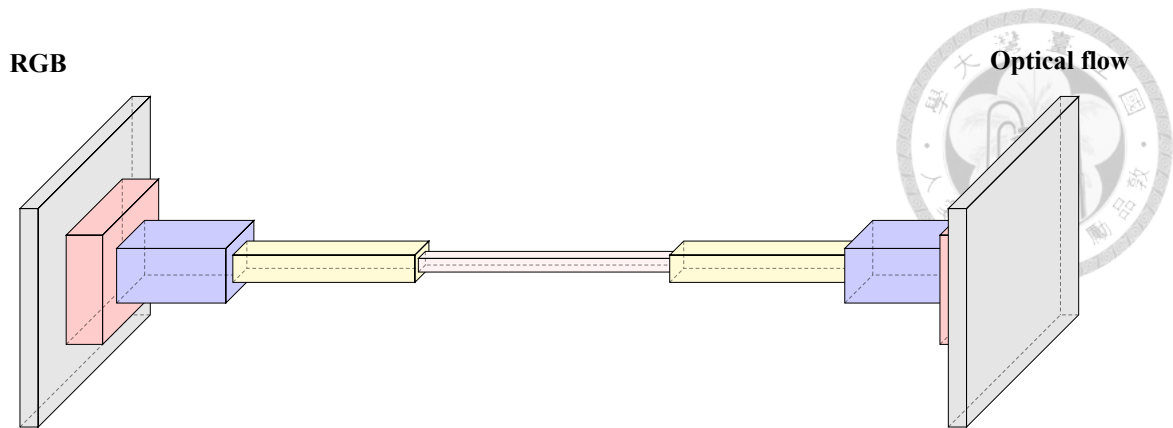


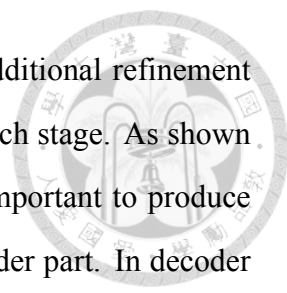
Figure 2.2: **Optical Flow Autoencoder.** Traditional autoencoder using deep learning architecture.

### 2.1.2 Autoencoders

FlowNet, [18], as one of state-of-the-art in the field of optical flow estimation provides a novel idea to generate accurate optical flow via an autoencoder. Autoencoder, [19], is a useful unsupervised architecture to learn and extract vital features in the learning progress. FlowNet leverages the characteristic of autoencoder and changes the architecture to supervised learning by feeding two RGB images as input data and the corresponding optical flow as ground truth. Moreover, they argue that adding variational image resolution into the learning procedure increases the performance rather than using naive interpolation methods. This is the first proposed method to generate optical flow by deep learning method.

As shown in Fig. 2.2, the overall architecture is an autoencoder using a pair of RGB images as input to generate corresponding optical flow with refinement network. Leveraging optical flow with autoencoder architecture, it is feasible to solve the optical flow estimation problem. Encoder is composed of several convolution layers and activations so as to extract spatial features and temporal features simultaneously. The encoded features should include the relationship among the same pixel in two images which represent the moving displacement of that pixel. Furthermore, the shape of each objects or background is encoded as well due to the fact that the edges should be detected and recognized so as to generate accurate optical flow. In the decoder part, not only the decoded features but also the features from encoder are used to generate final optical flow at each stage.





Since the quality of predicted optical flow should be guaranteed, additional refinement network is inserted which constraints the predicted optical flow at each stage. As shown in figure, features are extracted via convolutional layers which is important to produce from large-scale to fine-grained information for optical flow in encoder part. In decoder part, optical flow are generated stage by stage with refinement network. As going deeper, the feature map size decreases while the channels increases until reaching the middle of the architecture. The architecture is roughly symmetric, which means the feature map size corresponding to the same layer is equivalent.

Except for Flownet, there are still other solutions to this problem. Flownet2.0, [20], is an advanced version of Flownet with several submodules to estimate accurate optical flow. Each submodule is an autoencoder for optical flow estimation, that is to say, each submodule generates a predicted optical flow but incrementally improves the quality of optical flow estimation. As a result, the final optical flow is predicted stage by stage. Due to the fact that the concatenating submodules can only deal with larger scale of displacement, so [20] argues that adding a small-displacement submodules handling small-displacement parallel to the other concatenating submodules. Finally, the final predicted optical flow is produced by fusing two optical flow together. SPynet, [21], provides another point of view, feeding original resized RGB images corresponding to the resolution at each stage to estimate optical flow by adding and upsampling them element-wisely. In short, SPynet utilizes each optical flow estimation at each stage with a upsampling layer to generate final optical flow, like residually adding one to each other. Not surprisingly, this kind of design focuses on large-scale displacement optical flow estimation because the input of the smallest resolution is a RGB image and it is hard to generate fine-grained optical flow in larger resolution because there is insufficient convolution kernels to extract detailed features. Another work, [22], argues that unsupervising is benefit to train an optical flow estimation network.

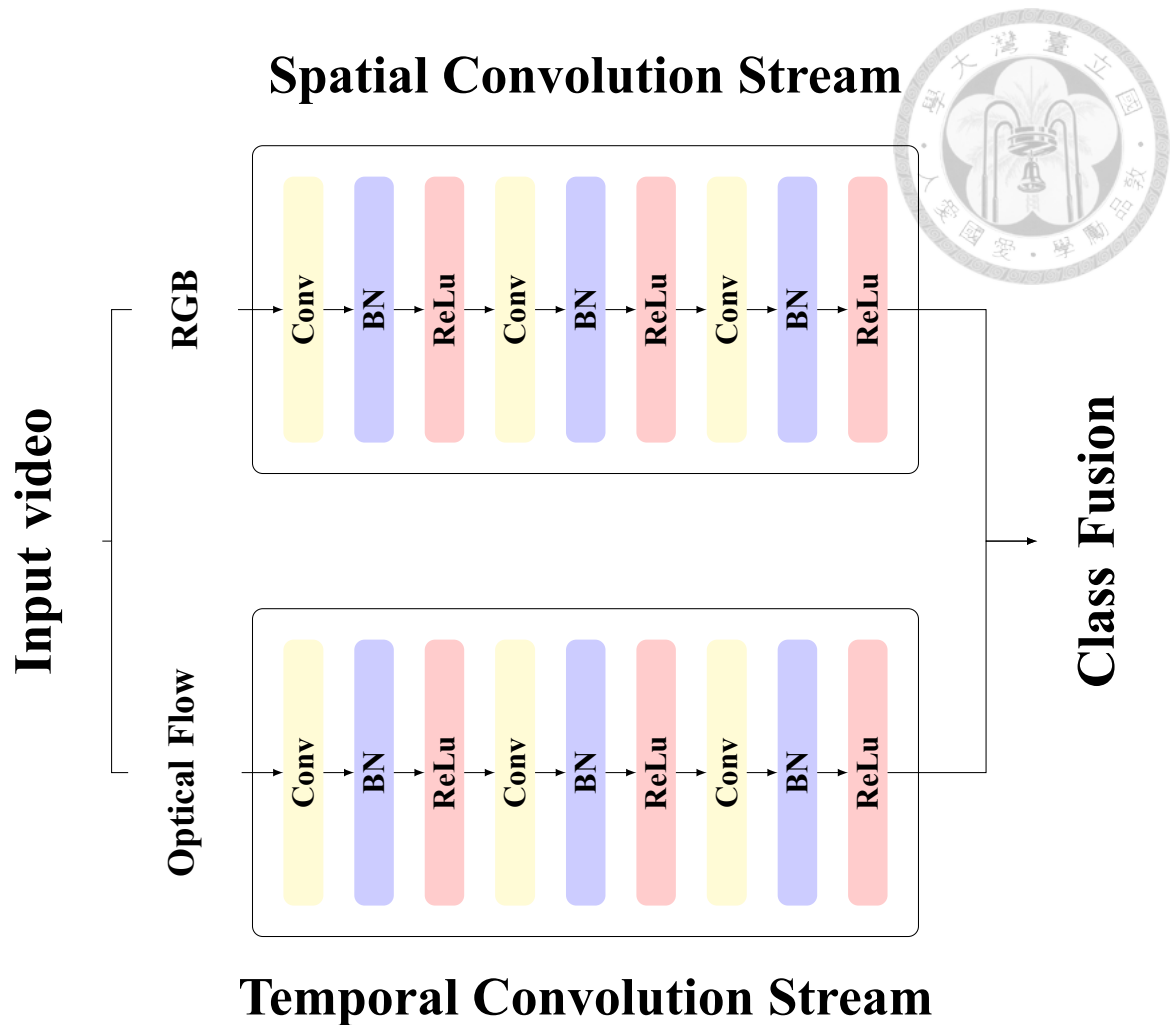
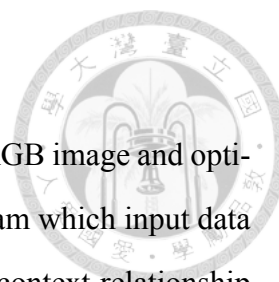


Figure 2.3: **Two Stream**. Two stream architecture include two stream, spatial stream and temporal stream.

## 2.2 Action Recognition

Action recognition using deep learning methods are introduced in this section. There are two perspectives, two-stream and 3D ConvNet, to solve action recognition problems. Generally, two-stream architecture includes two kinds of input, RGB and optical flow from which extracts spatial and temporal features respectively. 3D ConvNet takes an opposite way, stacking several RGB images, to extract spatial and temporal features simultaneously. In Section 2.2.1 and Section 2.2.2, two stream architecture and 3D ConvNet are well explained in details.

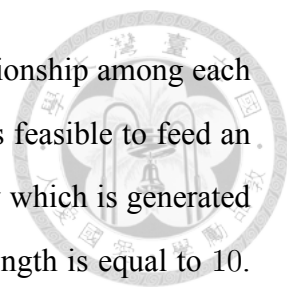


### 2.2.1 Two-Stream

Two-stream architecture literally means two types of input data, RGB image and optical flows, feeding into two parallel stream. On the one hand, the stream which input data is RGB frame represent spatial stream. Spatial stream focus on the context relationship among pixels in a single RGB image and extract vital cue in spatial information which leads to accurate action recognition. On the other hand, optical flow represents the temporal information between two RGB images, so the stream with optical flow as input data is temporal stream. Temporal stream mainly focus on the context relationship between images in a sequence of video frames. This stream extracts crucial temporal information for improving action recognition accuracy. As a matter of fact, after important features are extracted from each stream, action recognition relies heavily on aggregated method, in other words, the way to fuse spatial features and temporal features is extremely important for a machine to recognize an action. As shown in Figure 2.3, each stream contains several convolution unit which includes convolution layer, activation function, etc. The input of two stream architecture are RGB image and stacked optical flows which is fed into spatial stream and temporal stream. Due to the fact that optical flow are encoded with temporal information, the stream with optical flow as input represents temporal stream. Likewise, the spatial stream aims to extract spatial relationship among pixels in a single RGB image, so it named after it.

Originally, in [23], the fusion is inserted after fully connected layer, but [24] discover that fusion after the last convolution layer boosts the performance. Afterward, [8] and [6] argue that early interaction between spatial stream and temporal stream improves performance. Not only after the last convolution layer before fully-connected but also previous convolution layers are built with connection to generate spatiotemporal features in the earlier stage. To sum up, extracting spatiotemporal features which is composed of spatial features and temporal features is extremely important for action recognition problems.

As shown in Fig. 2.3, two stream architecture has two streams, spatial stream and temporal stream. The original input video can be pre-processed into RGB frame and optical flows. The spatial stream is mainly focus on extracting spatial features in an image



while temporal convolution stream is mainly focus on temporal relationship among each time step due to the input is stacked optical flows. For instance, it is feasible to feed an RGB image of a video clip at time  $t$ , and send a stacked optical flow which is generated from a video clip from time  $t - 5$  to  $t + 5$  if stacked optical flow length is equal to 10. Each stream contains several convolution layers and activations to extract features, but with some minor changes depending on each work.

Despite the fact that spatiotemporal features is vital for action recognition, integrating spatiotemporal features in each time step as well as fusing spatial features and temporal features account for a large proportion of action recognition prediction. [7] compare several aggregation methods for two-stream architecture, e.g. mean-pooling, max-pooling, VLAD [25], etc. For different dataset, aggregation methods influence the performance significantly with the same architecture, so choosing suitable aggregation method is another challenging issue. [2] provide a novel mechanism to adaptively pool useful features so as to integrate each local spatiotemporal features into global spatiotemporal features for action recognition prediction. [4] suggest that all features generated from both spatial stream and temporal stream should be categorized to specified attributes which is initialized VLAD. Leveraging unsupervised method to extract useful information from each attribute in the learning progress, [4] result in a good performance.

Although two-stream have shown promising results, optical flow estimation should be pre-calculated for temporal stream. Due to this limit, two-stream architecture is hard to achieve real-time. Also, the learned spatiotemporal information is fused by other mechanism or special design unlike 3D ConvNet directly extracting spatiotemporal information via 3D kernel. To sum up, two-stream is still one of the best architecture for action recognition though it has some drawbacks.

### **2.2.2 3D ConvNet**

3D ConvNet is another deep learning architecture, apart from simply using two-dimensional convolution kernel to extract spatial and temporal features as illustrated in two-stream architecture, it utilizes three-dimensional convolution encoding spatial and temporal in-

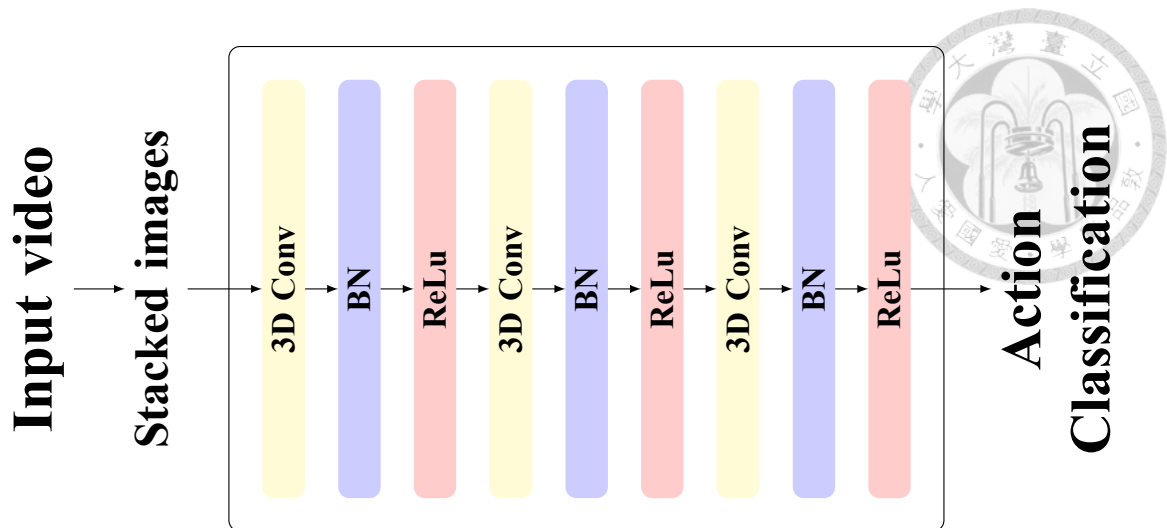


Figure 2.4: **3D ConvNet**. 3D ConvNets architecture.

formation together. Despite the fact that 3D ConvNet is capable of extracting context features in both spatial domain and temporal domain simultaneously, 3D ConvNet has its limit of using too much memory and requires high computability. Also, an action can be separated into three key-poses generally, e.g., starting pose, transition pose, and ending pose. Now that, it requires sufficient three dimensional kernels to encode and learn the features of action while the used memory often exceeds the limit of hardware upper bound. Furthermore, a video clip is often too noisy due to some irrelevant frames between action and action. Consequently, modeling a suitable architecture with proper kernel design is extremely vital for 3D ConvNet due to the constraints.

As shown in Fig. 2.4, the input video is separated into several continuous frames. The strength of 3D convolution is to extract spatiotemporal features which is vital for action recognition. Similar to normal 2D convolution, 3D convolution consider extra dimension, time axis, into account. 3D ConvNet has 3D convolution kernel to extract both spatial and temporal features together which can be viewed as spatiotemporal features. Similar to 2D convolution, 3D convolution conduct convolution on *height*, *width* as well as *time* axis, so 3D convolution consider additional features on time axis. The input of 3D ConvNet are stacked RGB images which is similar to temporal stream of two stream architecture except that 3D ConvNet use RGB images.

Firstly, [10] provide a novel idea that applying 3D convolution on stacking continuous RGB images is benefit to extract spatiotemporal feature encoded both spatial and tempo-

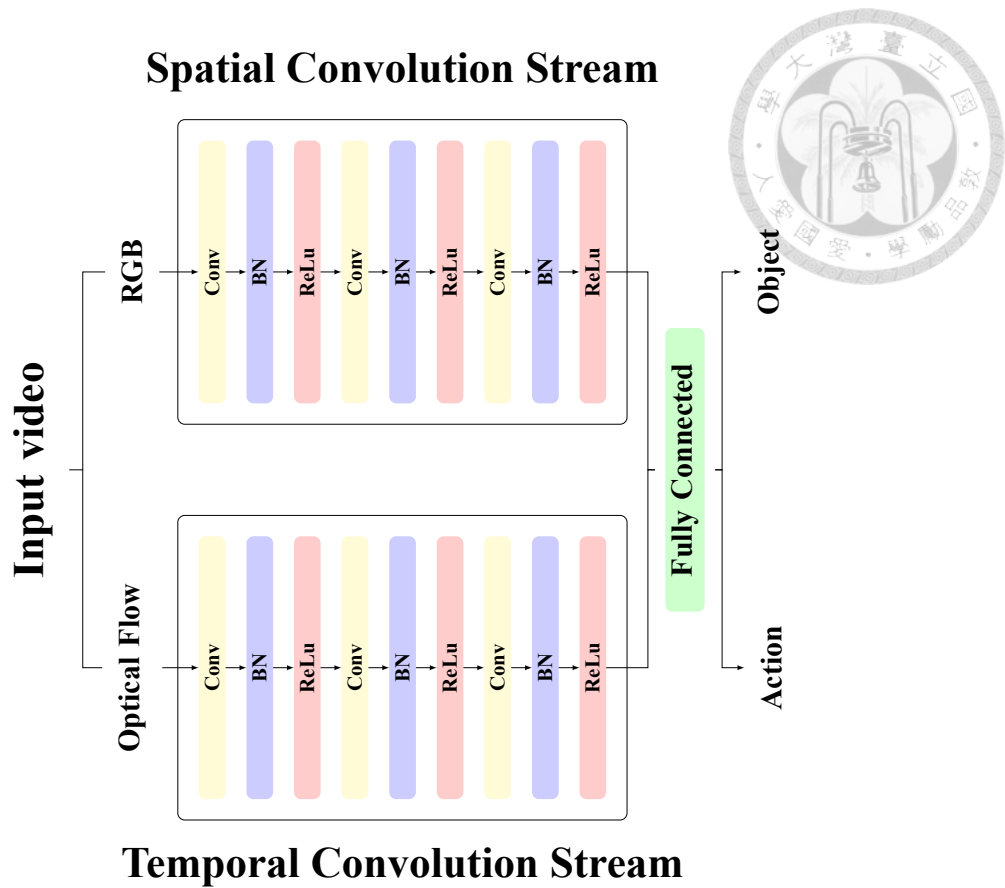


Figure 2.5: **Joint Learning of Object Detection and Action Recognition.** Joint learning of object detection and action recognition is described in the figure.

ral features at the same time. They find out the best suitable temporal kernel length for analyzing action dataset. 3D ConvNet has been widely-used as a feature extractor. Afterward, by utilizing the concept of [10] and [26], [11] is able to predict action as well as the action bounding box. Since [26] is the state-of-the-art in the field of object detection, [11] replace the 2D convolution layer in [26] with 3D convolution layer in order to extract spatiotemporal features.

Most 3D ConvNet methods use single 3D convolutiona kernel to extract spatiotemporal features, [12] make good use of residual unit as inserting into 3D ConvNet to get fine results. Since [27] and [28] have shown robustness in extracting features, [12] replace traditional convolution layer into residual unit to grantee the extracted features are sufficiently vital. To sum up, 3D ConvNet are used to extract spatiotemporal features directly though it often has hardware limitations.

## 2.3 Joint Learning

In this section, we are going to describe joint learning methods for action recognition. Joint learning means learning two related things at the same time due to using same domain features in the latent space. That is to say, simultaneously learning two subjects by using the same features. In section 2.3.1, we introduce joint learning on object and action. In section 2.3.2, joint learning optical flow and action is illustrated in 2.3.2.

### 2.3.1 Object and Action

According to [29], object detection in single image and action recognition are related due to some human actions are interacted with static objects. In this perspective, it is inspiring that finding the interaction among the objects and trying to encode the relationships between them should be vital to distinguish which action occurs. For instance, a man has a cigarette in his right hand. it is obviously that an action has occurred which is "smoking". Consequently, the object detection problem is feasible to solve with action recognition problem. Surprisingly, learning object detection and action recognition at the same time has shown promising results, while learning only object detection or action recognition individually shows inferior performance. Furthermore, based on joint learning, zero-shot learning can be marginally fulfill due to shared information of both object detection and action detection.

The full designed architecture can be simplified as shown in Figure. 2.5 which jointly learns object detection and action recognition. By utilizing two stream architecture, spatial stream extracts relationship in a RGB image and temporal stream extracts temporal features among stacked optical flow. That is to say, spatial stream and temporal stream are used to extract spatial features from a single RGB image and temporal features from stacked optical flows, respectively. Then, fusing the features encoded from each stream into spatiotemporal features, sharing information is fed into fully connected layer which outputs the integrated sharing information for two classifier so as to jointly learn action recognition and object detection simultaneously, which means the fully connected fused the features from both spatial stream and temporal stream into integrated spatiotemporal

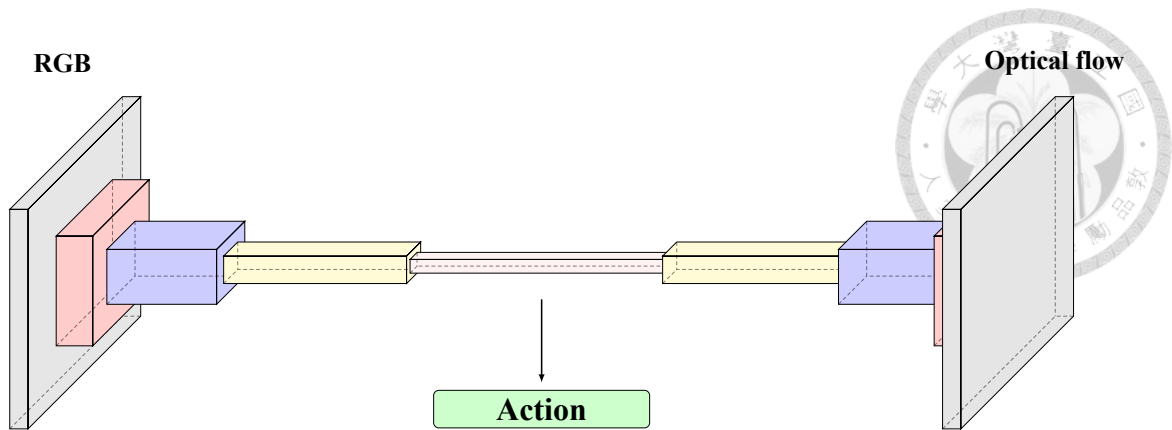


Figure 2.6: **Joint Learning of Optical Flow and Action.** Joint learning of optical flow estimation and action recognition.

features. Also, from the sharing features integrated by the fully connected layers, object detection classifier and action recognition classifier learned jointly due to this design. In the light of this, joint learning of action recognition and object detection boosts performance and is benefit to extract vital features which is composed of object information and the interaction among these objects. As a result, this kind of design is capable of multi-tasking by sharing the same features.

### 2.3.2 Optical Flow and Action

Action recognition relies on spatiotemporal features in order to distinguish accurate action class. Since joint learning has shown promising results in several fields, it is feasible to state the action recognition problem by jointly learning both action recognition and optical flow simultaneously due to the fact that optical flow is the crucial information in the field of action recognition. [13] utilize an autoencoder similar to [18] which is able to generate optical flow, and add extra fully connected layer after the encoder so as to jointly learn action recognition and optical flow.

As shown in 2.6, action recognition and optical flow estimation are jointly learned with this novel design. Action recognition is heavily relies on encoded spatiotemporal features. Since optical flow is the crucial information in the field of action recognition, the encoded spatiotemporal features should be composed of spatial features of each RGB image and the pixel relationship, temporal features, among two RGB images; otherwise, the optical flow



cannot be estimated via this architecture. Since joint learning has shown promising results, learning action recognition and optical flow is feasible. With an autoencoder extracting spatiotemporal features for action recognition, the optical flow can be decoded via the same network due to the design of [18]. Using pairs of RGB images as input, this network is able to jointly generate corresponding optical flows as well as action class. The encoded features from encoder are shared by both optical flow estimation and action recognition.



## Chapter 3

# Optical Flow Estimation

In this chapter, we first illustrate our design concept briefly with some prior knowledges in Section 3.1. In Section 3.2, we introduce our autoencoder network and minor details of each unit. Afterwards, we introduce our self-designed refinement network in Section 3.3. Finally, we elaborate some knowledges and concerns of our design in details in Section 3.4.

### 3.1 Design Concept

Inspired by [13] and [18], we design a novel autoencoder architecture which is similar to FlownetS in [18] with multiple pairs of RGB images as input and the corresponding optical flows as output, but we replace each convolution layer with residual unit, introduced in [27] and [28]. As a matter of fact, Residual Network *ResNet*, [27], [28], has shown strong capability of addressing relationship between pixels and pixels. Not only in image super-resolution but also in image denoising, ResNet is applicable and suitable for action recognition problems, so it is reasonable to leverage the characteristics of ResNet for optical flow estimation. However, most of the existing networks designed for optical flow estimation do not insert batch normalization layers, [30]. Inserting batch normalization layer after convolution layer is a risky thing which leads to rescaling the distribution of the input values while at the same time, however, it helps the training procedure converge faster and includes regularization term automatically due to the scaling.

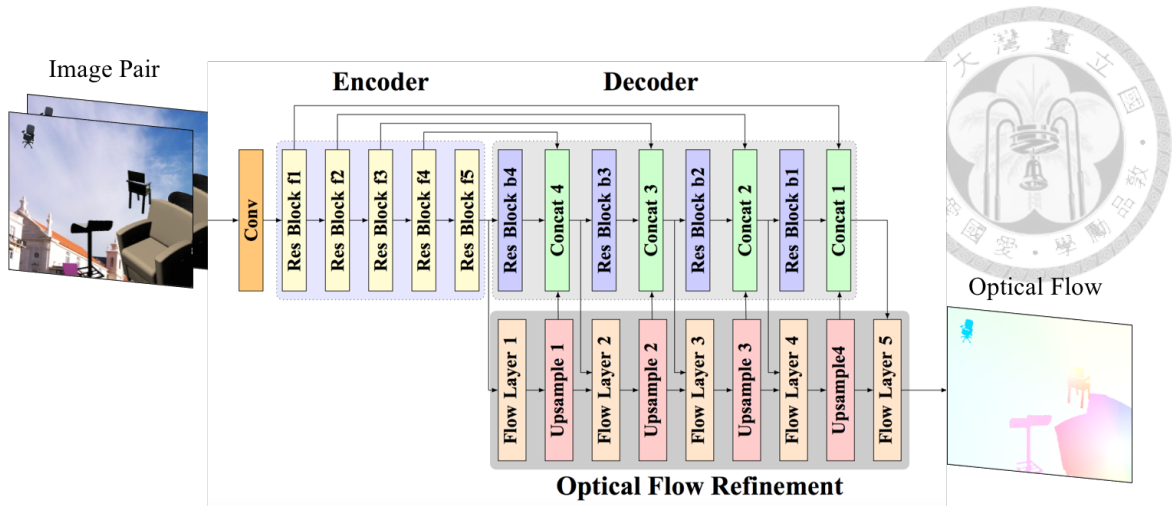


Figure 3.1: **Autoencoder Architecture** Architecture of our designed autoencoder.

As a result, we argue that adding residual unit into the autoencoder will increase the capability of finding strong connection between pixels but constraints to avoid the side-effect that brings from batch normalization are required. Optical flow estimation generated from each stage should not add batch normalization layer after convolution layer to avoid rescaling the values because optical flow is the real vector value that describes the displacement of one pixel at one image to the other image in  $X$  – and  $Y$  – axis. Moreover, in the decoder part, we add an extra relatively strong connection between each resolution of predicted optical flow.

The whole autoencoder architecture can be seen in Figure 3.1. Using five residual unit as encoder to extract spatiotemporal features and four residual unit as decoder, we then feed the encoded spatiotemporal features to decoder as well as refinement network. Noticeably, we down-sample or upsample when going through each residual unit in encoder or decoder. The refinement network is used to ensure the quality of the predicted optical flow at each stage. Also, taking features from encoder, decoder network is able to capture fine-grained details from encoder and decode better features. In Section 3.2, architecture of the autoencoder will be illustrated and more optical flow estimation details will be elaborated in Section 3.3.

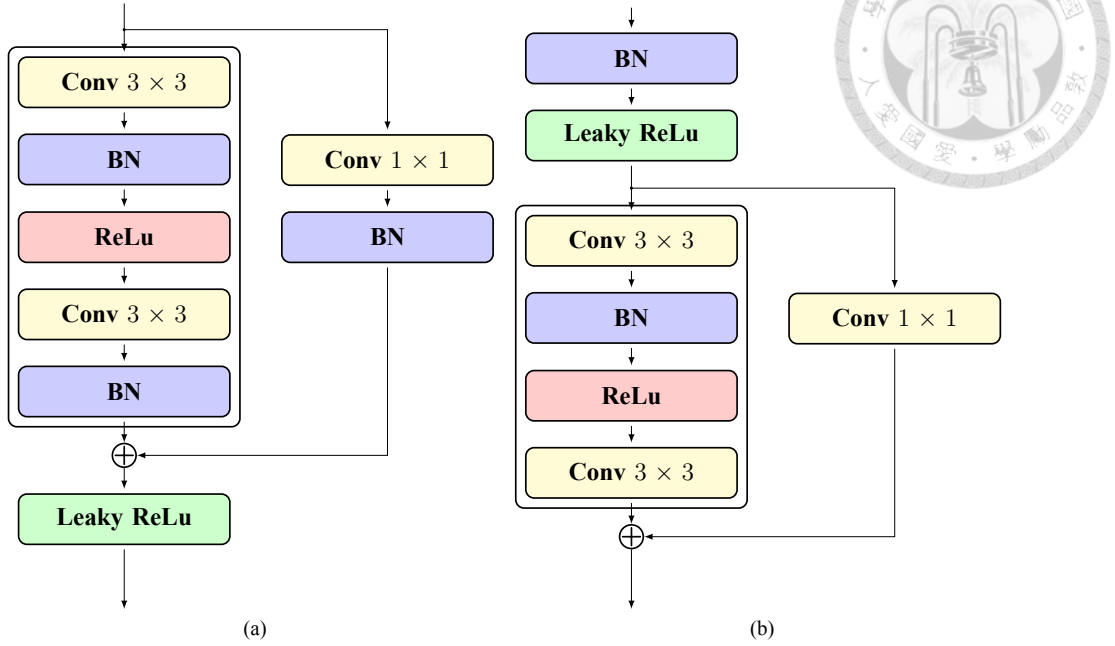


Figure 3.2: **Encoder (a)** Architecture of encoder.

## 3.2 Architecture

In this section, encoder of the autoencoder is introduced in Section 3.2.1. Decoder of the autoencoder is illustrated in Section 3.2.2. In each section, minor details will be well explained.

### 3.2.1 Encoder

We utilize two kinds of residual unit, described in [27] and [28], and the way which we implement is as shown in Figure. 3.2. We use the same channels for each residual unit corresponding to the same layers in FlownetS, [18]. Leveraging residual unit, the spatiotemporal features are extracted more and more explicitly via layer by layer with two stacked RGB images as input. Due to the fact that optical flow represents the moving displacement of the same pixel in two images, the encoded features should be able to describe the spatial relationship in a single image of two pixel among two images. Consequently, spatiotemporal features should be completely extracted via the network. Furthermore, we use the same design of residual unit as described in [27] and [28], but we replace the original activation, ReLu, with Leaky ReLu. Due to the fact that Leaky ReLu takes neg-

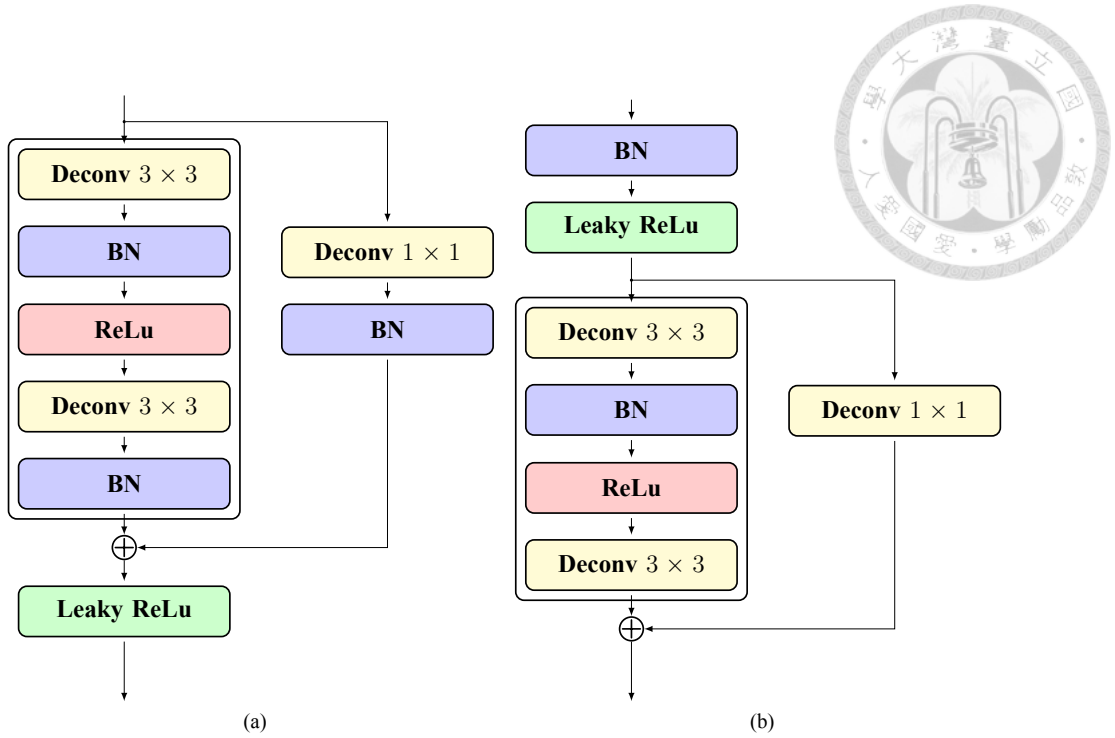


Figure 3.3: **Decoder (a)** Architecture of Decoder.

ative terms into account while ReLu eliminates negative terms. In the light of this, we use Leaky ReLu as integrated activation function in order to extract much better features. Noticeably, we use five residual unit for encoder because adding more extra residual unit may run out of memory.

### 3.2.2 Decoder

Unlike *FlownetS* using only one deconvolution layer to decode higher feature domain, we additionally implement two kinds of residual unit which is similar to the encoder part as shown in Figure. 3.3. Despite the fact that adding residual unit into the autoencoder increases the performance, it is not enough to generate fine-grained optical flow. For *FlownetS* [18], for instance, the predicted optical flow at each resolution only has indirect connection which only pass the upsampled optical flow as a feature map to the concatenation layer. However, the upsampled optical flow has not been fully utilized. In order to build the direct connection between each stage, it is reasonable to resize the smaller predicted optical flow to the bigger resolution as an basis of predicted optical flow at that stage. As shown in Figure. 3.1, decoding spatiotemporal feature needs to extract

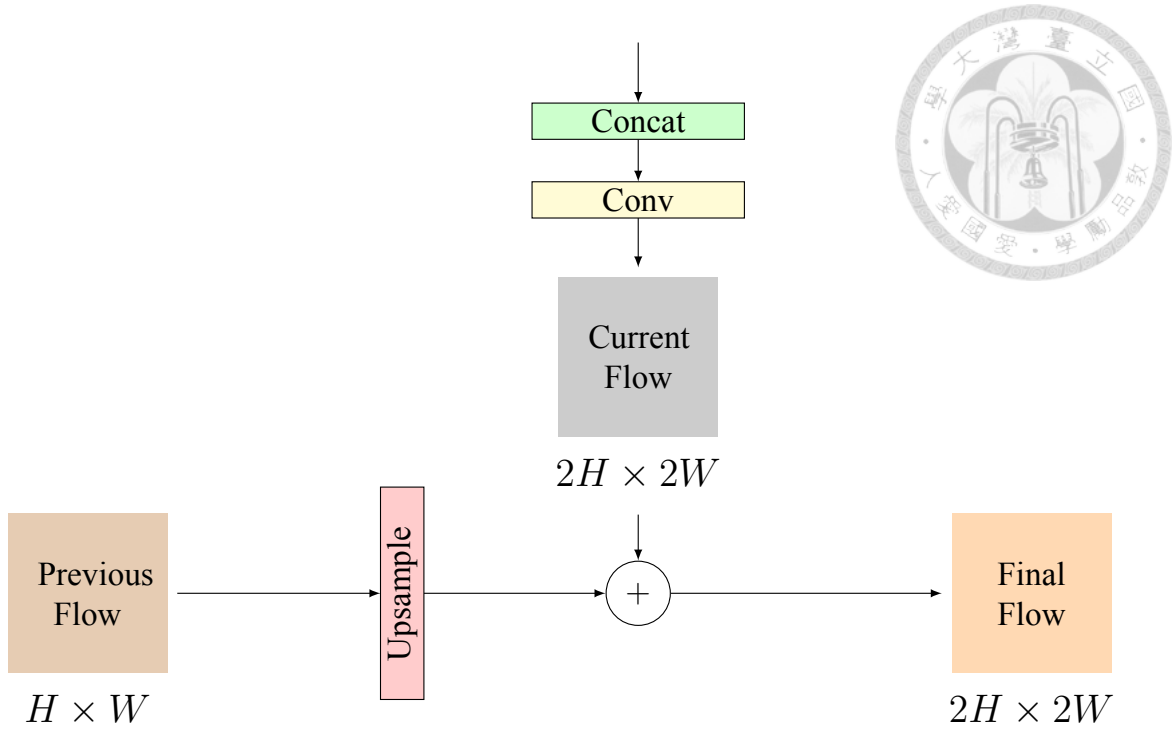


Figure 3.4: **Optical Flow Refinement** Process flow of optical flow refinement.

spatiotemporal features and some fine-grained details from both higher feature space and encoder feature space in order to predict accurate optical flow. Therefore, using variational resolution of optical flow is able to guarantee spatiotemporal features which is the vital part for predicting optical flow. Furthermore, the spatiotemporal features are simultaneously crucial for action recognition.

### 3.3 Refinement Network

Now that we are able to produce high-quality spatiotemporal feature, decoding is another vital issue. Since we utilize variational resolution refinement of optical flow in order to ensure getting sufficient spatiotemporal information at each stage, it is reasonable to use previous predicted optical flow as an basis of current predicted optical flow at next resolution stage using the concept described in [21]. That is to say, we let a predicted optical flow go through a up-sample layer, deconvolution layer, with stride 2 which increase the feature map size by a factor of 2, and we leverage this up-sampled optical flow as a basis of the predicted optical flow at this bigger resolution. As shown in Figure 3.4, the architecture of final optical flow at each stage is generated via the procedure. The up-sample

layer is used to resize the smaller optical flow to fit current size and the final optical flow is generated by both up-sampled optical flow and current predicted optical flow from concatenation layer. That is to say, we obtain another predicted optical flow produced from the concatenating feature map, so we use a weighted sum mechanism so as to generate the final optical flow of this resolution.

In order to train an autoencoder for optical flow estimation, we use end-point-end error,  $EPE$ , as an error index or loss function to train our network and evaluate the performance. Note that  $EPE$  is defined as

$$EPE(U, V, U', V') = \sum_{i,j} \sqrt{(u_{i,j} - u'_{i,j})^2 + (v_{i,j} - v'_{i,j})^2} \quad (3.1)$$

where  $U, V$  are the displacement on  $X$ - and  $Y$ - direction of the predicted optical flow estimation, respectively, and  $U', V'$  are that of ground-truth optical flow estimation. The error can be viewed as moving distance error between correct and predicted moving distance which is equivalent to the magnitude of the subtraction of two vector value.

Nevertheless, training the optical flow with this design has an implicit question which is that the up-sampled optical flow after deconvolution should be as close to the ground truth as possible. As a result, we use a weighted loss function to address this issue, so the total loss at each stage  $k$  is described as

$$\begin{aligned} SubLoss_k(U_k, V_k, U'_k, V'_k, U_{k-1}^{up}, V_{k-1}^{up}) = \\ EPE(U_k, V_k, U'_k, V'_k) + \gamma_k \times EPE(U_{k-1}^{up}, V_{k-1}^{up}, U'_k, V'_k) \end{aligned} \quad (3.2)$$

where  $U_{k-1}^{up}$  and  $V_{k-1}^{up}$  represent the displacements on  $X$ - and  $Y$ - direction of the upsampled optical flow estimation at stage  $k$ , respectively.  $\gamma_k$  is the coefficient to control the tendency to upsampled optical flow or new generated optical flow. The overall error for training this network can be categorized as

$$\begin{aligned} FlowLoss = \lambda_1 \times EPE(U_1, V_1, U'_1, V'_1) + \\ \sum_{k=2}^{k=5} \lambda_k \times SubLoss_k(U_k, V_k, U'_k, V'_k, U_{k-1}^{up}, V_{k-1}^{up}) \end{aligned} \quad (3.3)$$

where  $\lambda_k$  is the coefficient at each stage  $k$  which is 0.32, 0.08, 0.02, 0.01, 0.005, respectively.



### 3.4 Remarks

Adding extra connection between optical flows at the previous and the current stage has an extraordinary effect on convergence of loss training and performance enhancement. Originally, FlownetS [18] provides a good solution to accurate optical flow estimation via an autoencoder. We implement the same autoencoder network by adding residual unit with additional batch normalization layer after each convolution layer except generating optical flow ones. As a matter of fact, the value of the optical flow should be a true pixel displacement value not supported to be rescaled by batch normalization layer. Also, we implement the revised version of FlownetS by adding batch normalization layer after each convolution layer except optical flow generating layer which also shows similar results to the version without batch normalization. However, adding batch normalization indeed accelerate the converging speed. Since we ensure the feasibility of adding batch normalization after convolution layer, we adopt residual unit by replacing the simple convolution layer. Also, we compare the architecture which adds residual unit into both encoder and decoder with that which only inserts residual unit into encoder. Moreover, we look into whether adding upsample error increase the performance.

From our perspective, adding residual unit in both encoder and decoder is able to generate better quality optical flow. With the constraints of upsampled error, defined in Equation. (3.2), the predicted optical flow are double guaranteed. The total loss is defined as Equation. (3.3). More details and experimental results will be elaborated in Chapter 5.





## Chapter 4

# Action Recognition

In this chapter, we first illustrate how to bridge the gap between optical flow estimation and action recognition in Section 4.1. Secondly, we introduce our proposed ResFlow in Section 4.2 with minor details. In Section 4.3, we elaborated our self-designed mechanism to aggregate local spatiotemporal features into global spatiotemporal features for action recognition.

### 4.1 Optical Flow to Action Recognition

In Section 3, we introduce an autoencoder for estimating optical flow. In short, optical flow is composed of two channels which record the displacement along  $X$ - and  $Y$ - axis. Furthermore, our autotencoder predicts optical flow at each stage in order to ensure that sufficient features are extracted. Since optical flow represents the temporal displacement from the first image to the second image, the temporal displacement should be encoded via the autoencoder. Furthermore, not only the temporal features but also the spatial features are encoded via the autoencoder due to the fact that the object position inside the two images should be corresponding to each other. Consequently, spatiotemporal features are automatically encoded due to the characteristic of optical flow and autoencoder.

Now that we are able to access spatiotemporal features from each frame via an autotencoder, each spatiotemporal feature represents an instant motion variation. How to predict action using these extracted spatiotemporal feature is an open question. In the

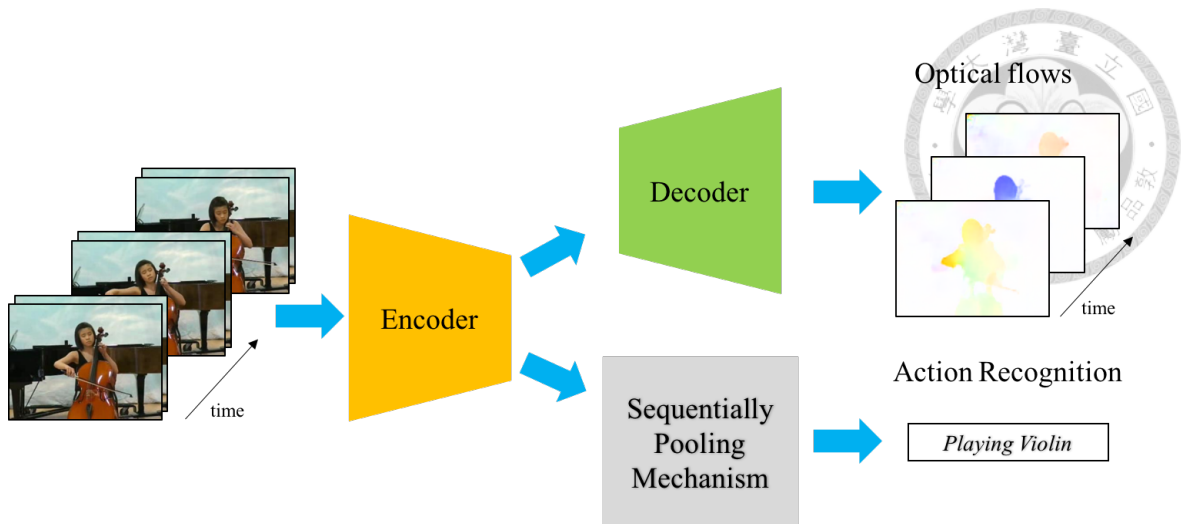


Figure 4.1: **Overall Architecture.** The overall Architecture which contains both optical flow estimation task and action recognition task.

viewpoint of frame-level, each spatiotemporal feature is a cue which leads to a unique action and they are independent of each other. Contrarily, viewing them as a sequence of cues in video-level is totally different. All spatiotemporal features related to the same video has only one label, which means only one prediction is produced. Naive methods, *e.g.*, average-pooling, max-pooling, *etc.*, are widely-used but they are too biased because of the following reasons: Firstly, max-pooling works fine if a particular motion feature is sufficient for recognizing an action, however, actions are normally composed of several consecutive segments in a video. Due to the same reason, mean-pooling may encounter the condition that only a few features are important, but the rest are noisy features. In light of this, we must find a suitable way to aggregate all of these spatiotemporal features. Not only integrate all these spatiotemporal features, but also consider the temporal ordering because the order matters, *e.g.*, a man sitting down and standing up may be viewed as the same action which is not correct literally.

## 4.2 ResFlow

We propose a novel design, ResFlow, which predicts optical flow estimation and proceeds action recognition simultaneously. The overall architecture is shown in Figure 4.1. As a matter of fact, using autoencoder to extract spatiotemporal feature is not only suitable for estimating optical flow but also extracting spatiotemporal features for action recogni-

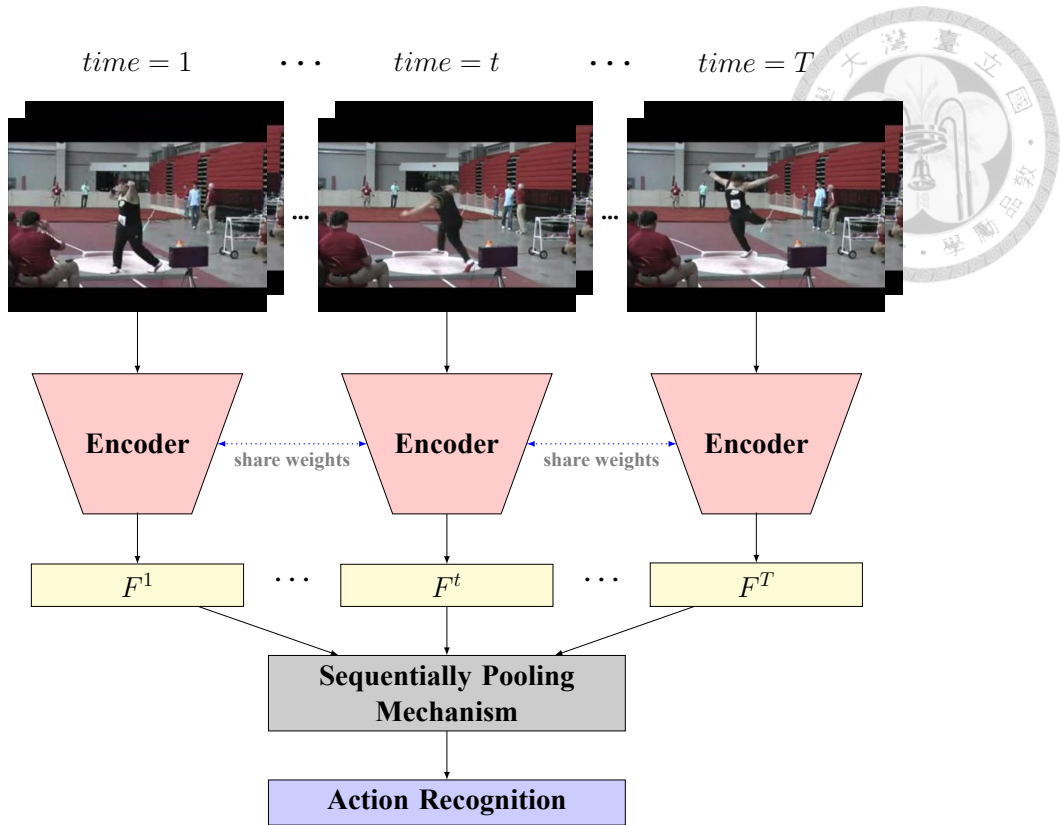
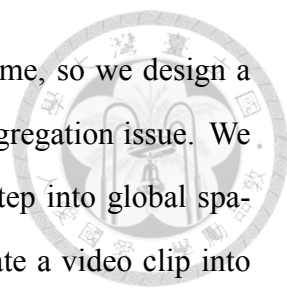


Figure 4.2: **Aggregation of Spatiotemporal Features.** Process of aggregating spatiotemporal features for action recognition.

tion. Furthermore, multitasking has been proved to be helpful for both parts due to the fact that common sharing features are enhanced by both tasks. Since optical flow is of the key factor of action recognition and it represents the temporal features of the video, estimating optical flow and action recognition at the same time is reasonable and realistic. We fully utilize the characteristic of the autoencoder architecture for optical flow estimation to extract spatiotemporal features for action recognition. In order to aggregate local spatiotemporal features at each time step, we leverage our self-designed Sequentially Pooling Mechanism as a feature selector to generate the best global spatiotemporal feature. Finally, we get action recognition result via feeding the global spatiotemporal feature into the fully connected layer.

Sharing the same encoder which extract spatiotemporal features, we branches the output to two objectives which are optical flow estimation and action recognition. The branch of optical flow estimation is concatenated with decoder, described in previous chapter, after the encoder, while the other branch is concatenated with a aggregation mechanism. In



light of this, there are several spatiotemporal features from each frame, so we design a novel mechanism, Sequentially Pooling Mechanism, to deal with aggregation issue. We aim at aggregating all the spatiotemporal features from each time step into global spatiotemporal feature to predict final action. That is to say, we separate a video clip into several image pairs as input of ResFlow and outputs both corresponding optical flows and an action label.

As mentioned previously, two stream architectures include spatial stream and temporal stream, so we figure a wise way to fuse two streams into one via the multitasking mechanism. Also, 3D ConvNet uses 3D convolution layer to extract spatiotemporal features, while we use autoencoder to extract spatiotemporal features. The 3D ConvNet considers a relatively larger period of time due to the fact that the receptive field becomes bigger as the network goes deeper. In order to consider the temporal ordering, we make good use of SPM so that we are able to produce global spatiotemporal feature through this aggregation mechanism to integrate all local spatiotemporal features from each time step.

As shown in Figure. 4.2, we train the network via a batch of pairing images of a video clip as input and an action label as output. In the light of this, the network infers one local spatiotemporal features at each time step. We adopt Sequentially Pooling Mechanism to aggregate these local spatiotemporal features into global spatiotemporal features as shown in the figure. We recursively get a score for each local spatiotemporal feature as an indicator to make sure whether this feature is vital or not.

### 4.3 Sequentially Pooling Mechanism

Due to the fact that the local temporal spatiotemporal feature contains temporal information in a time interval and each spatiotemporal feature at each time interval should be treated discriminatingly because not all the local features contribute to action recognition. That is, each confidence score for corresponding local spatiotemporal feature should be trained and learned via a fair and precise mechanism. As a matter of fact, there are some noisy frames in a video clip, *e.g.*, a *batting* action is composed of two segments which are throwing ball and batting. Therefore, two key action segments are throwing ball and

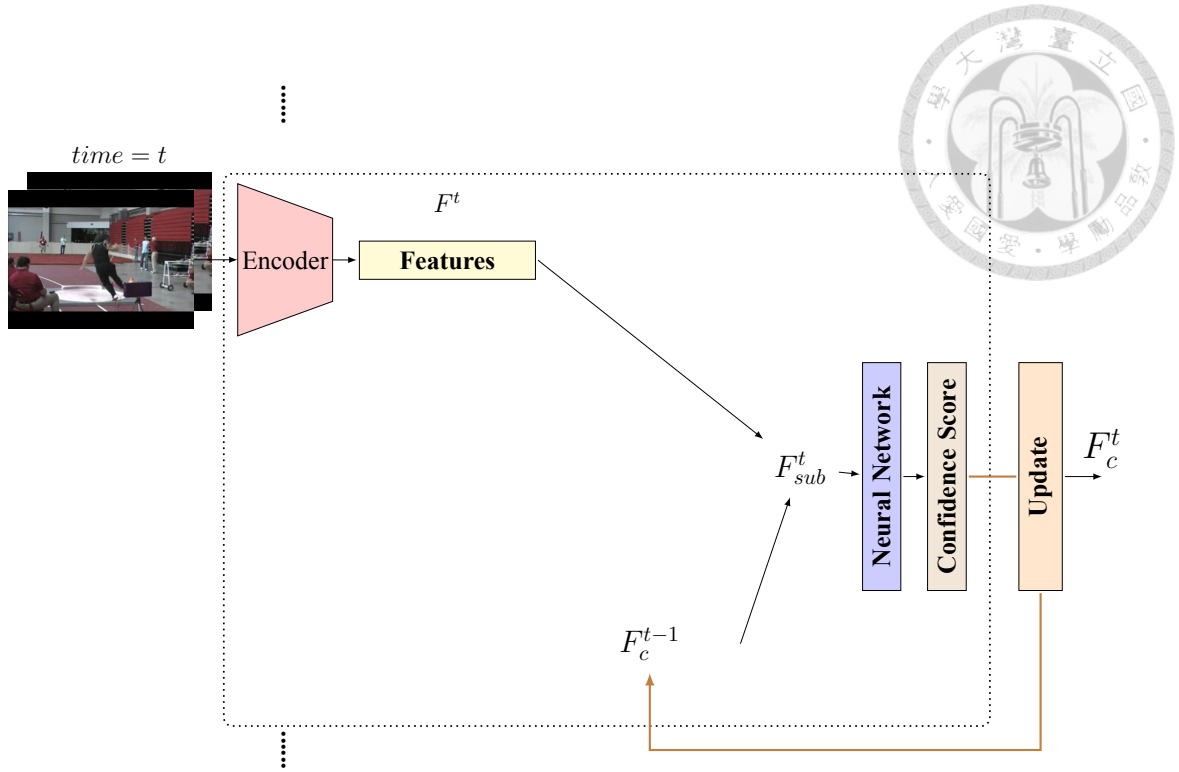


Figure 4.3: **Sequentially Pooling Mechanism.** Process flow of Sequentially Pooling Mechanism at time  $t$ .

batting, but if we use naive method, such as average-pooling or max-pooling, it may be hard for the network to learn correct features because, while the ball is flying, the frames are negligible. Also, these noisy frames may harm the performance.

Consequently, inspired by [2], we design Sequentially Pooling Mechanism, SPM, which generates the confidence score,  $S^t$ , of each residual input,  $F_{sub}^t$ , via three fully connected layers at time  $t$ , and each residual input is calculated by subtracting  $F_c^{t-1}$  with  $F^t$ . The SPM architecture can be seen in Figure 4.3. As shown in Figure 4.3, at time  $t$ , we input an image pair to the encoder and produce a feature vector, denoted as  $F^t$ . Then, we subtract the input feature with the condensed feature, denoted as  $F_c^{t-1}$ . Via a neural network, we are able to get a confidence score, denoted as  $S^t$ , which represents the importance of this subtraction input. Then, we use a weighted sum mechanism to update the new global spatiotemporal feature, denoted as  $F_c^t$ , which is also the condensed feature for next time step. Above all, the equation is written as

$$F_{sub}^t = F^t - F_c^{t-1} \quad (4.1)$$



$$S^t = \sigma(F_{sub}^t), S^t \in [0, 1] \quad (4.2)$$

$$\begin{aligned} F_c^t &= F_c^{t-1} + S^t \times F_{sub}^t \\ &= S^t \times F^t + (1 - S^t) \times F_c^{t-1} \end{aligned} \quad (4.3)$$

where  $S^t$  and  $F^t$  are denoted as the confidence score and the feature map generated after the last convolution layer in encoder, *Res block f5*, at time step  $t$ ,  $t \in 2, \dots, T$ , respectively. Initially, the condensed global spatiotemporal feature,  $F_c^1$ , is equal to the first spatiotemporal feature,  $F^1$ , at time  $t = 1$ . The neural network used in SPM consists of three fully-connected layers. We use *tanh* as the activation function of the first two fully-connected layer, and the last one uses *sigmoid* as activation function. Due to the fact that we aim at predicting the importance score of the input feature in a range of  $[0, 1]$ . Under this circumstance, SPM has capability of judging whether the current residual input is relevant or not so as to give a confidence score based on both accumulated condensed spatiotemporal feature and residual input. To sum up,  $F_c$  is computed by aggregating local spatiotemporal features,  $F^t$ , at each time step,  $t$ , and the equation of which is shown as

$$\begin{aligned} F_c &= S^T \times F^T + (1 - S^T) \times F_c^{T-1} \\ &= S^T \times F^T + (1 - S^T) \times [(S^{(T-1)} \times F^{(T-1)} + (1 - S^{(T-1)}) \times F_c^{T-2}] \\ &= S^T \times F^T + S^{(T-1)}(1 - S^T) \times F^{(T-1)} + (1 - S^T)(1 - S^{(T-1)}) \times F_c^{T-2} \end{aligned} \quad (4.4)$$

by induction, which is equivalent to

$$F_c = \sum_{t=1}^{t=T} (w^t \times F^t) \quad \text{and} \quad \sum_{t=1}^{t=T} w^t = 1 \quad (4.5)$$

where  $w$  represents the proportion of each local spatiotemporal features to the global spatiotemporal feature. Also, the summation of confidence score corresponding to each local spatiotemporal feature is equal to 1. As a result, SPM leverages share-weighted fully connected layers to fairly calculate confidence score. The equation 4.5 demonstrates that the global spatiotemporal feature is composed of local spatiotemporal features from each time

step. From equation 4.4, we show the global spatiotemporal feature is composed of input feature  $F^T$  and previous condensed feature  $F_c^{T-1}$ . Also,  $F_c^{T-1}$  is composed of  $F^{T-1}$  and  $F_c^{T-2}$ . Therefore, by induction, we can get a sequence of feature vectors from time step 1 to  $T$ . The coefficient of feature vector at time step is  $w^t$  which can be interpreted as a series of  $S^k$ , where  $k \in [2, t]$ .

**Remark.** Looking into SPM, we find out that SPM can not only judge the importance of spatiotemporal feature from each time step, but also aggregate them using weighted-sum mechanism. Furthermore, SPM fully utilizes the characteristic of sequential order that it aggregates them sequentially along time axis. In other words, SPM considers the temporal order of an action sequence and collect the needed important spatiotemporal feature from each time step. As shown in Figure 4.4, we recurrently output the confidence score and the condensed spatiotemporal feature, which contributes to the global spatiotemporal feature. As a consequence, this solve the issue that naive methods cannot handle with. Furthermore, SPM boosts the performance of the action recognition accuracy.

Compared to using LSTM to aggregate features from each time step, our Sequentially Pooling Mechanism is much simpler and easier to implement. Furthermore, we can see that the forgetting rate of each unit is steady which depends on manual setting. Also, the concept of our design is that we hope to produce global features based on previous knowledge, but treat each input feature at each time step independently. That is to say, the forgetting rate of LSTM will somehow fix the flexibility of aggregating features from each time step.

## 4.4 Implementation Details.

We select Tensorflow [31] as our training platform because of the large manipulability. We select Adam [32] optimizer to optimize our training loss. We define our training loss by a weighted sum of optical flow estimation loss described in Section 3.3 and cross entropy for action recognition task.

We choose the same training schedule, called *lonschedule*, described in [20] to train the optical flow autoencoder so as to get a pre-trained model for extracting spatiotemporal

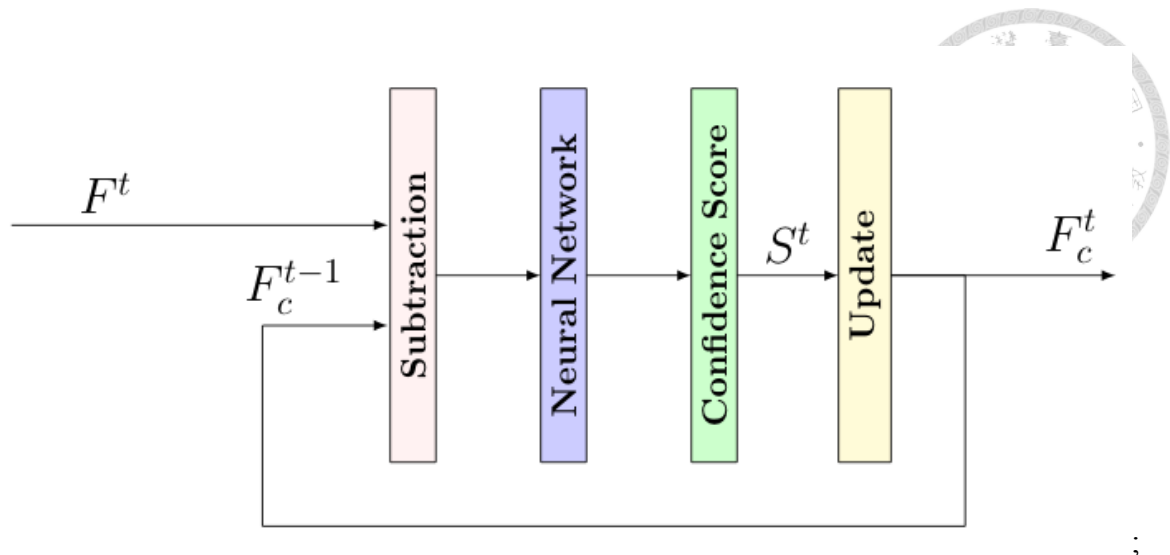


Figure 4.4: **Details of SPM.** This is a simplified visualization of SPM.

features. The learning rate is set to  $10^{-4}$  with a degradation of a factor of 2 after first 300K iteration and 100K iteration afterward. We also use  $l_2$  normalization to prevent overfitting with a factor  $10 - 4$ . Furthermore, we figure out that data augmentation is crucial to train an autoencoder. Data augmentation includes random brightness, saturation, hue, contrast, Gaussian noise by Gaussian distribution with a delta of 0.3. Also, we conduct horizontal and vertical flipping, translation by a factor between  $[-0.2, 0.2]$  with the image height and image width respectively, scaling in a range of  $[0.9, 2.0]$ , and rotation in a range of  $[-17, 17]$  degrees. We end up training procedure after 1000K iterations.

For training action classification problem, we use separated training which is described in [2] that use  $10^{-3}$  as the learning rate for training SPM and  $10^{-6}$  as the learning rate for training convolution layers, respectively. We find it hard to train SPM since it converges too quickly and easily overfits to the training data. Therefore, we also use image horizontal flipping to augment data. Also, we confine the maximum gradient value to 10 in each update iteration. This training technique has been adopted by several works before, *e.g.*, [2], [4]. We stop training after 50 iterations.





## Chapter 5

# Experiment

In this chapter, we introduce two popular optical flow datasets, *FlyingChairs* dataset and *SintelFinal* dataset, and their characteristics in Section 5.1. *UCF101* dataset and *HMDB51* dataset, two action recognition benchmark, are elaborated in details in Section 5.2. We evaluate our model for optical flow estimation on both *FlyingChairs* dataset and *SintelFinal* dataset in Section 5.3. In Section 5.4, we evaluate our proposed architecture on both *UCF101* dataset and *HMDB51* dataset. Also, we show the strength of our proposed Sequentially Pooling Mechanism, SPM.

### 5.1 Optical Flow Dataset

Training optical flow in deep learning architecture requires a sufficient large dataset with precise ground-truth which is hard to obtain data from real-world scene. Consequently, optical flow datasets are all made artificially, however, the quality and precision of the generated optical flows are absolutely correct due to the fact that it is precisely calculated by pre-built software. Thanks to the synthetic data, we are able to train a network for optical flow estimation. In Section 5.1.1, we describe the features of *FlyingChairs* dataset.




Optical Flow Estimation Visualization		
Image1	Image2	Ground-Truth
		

Figure 5.1: **Optical Flow.** Optical flow is calculated from evaluating the motion of first image and second image. Based on the first image, optical flow represents the moving displacement of the pixel along  $X$ – and  $Y$  axis.

### 5.1.1 FlyingChairs Dataset

Before going into the details of optical flow dataset, we first introduce the visualization of optical flow. Due to the fact that optical flow is a two channel value which cannot be visualized in the form of RGB image. Therefore, some researchers leverage the Munsell Color System to visualize optical flow estimation. Due to the fact that the Munsell Color System is able to describe a vector in the form of Value, Hue, and Chroma, optical flow can be visualized using Munsell Color System. As a matter of fact, each pixel of an optical flow contains  $X$ – and  $Y$ – displacement. Therefore, the vector direction of each pixel can be corresponded to a hue color and the magnitude of the displacement in  $X$ – and  $Y$ – direction is corresponding to a chroma in the form of Munsell Color System. As a result, we are able to use Munsell Color System to describe an optical flow.

FlyingChairs dataset, [18], is an optical flow dataset which is composed of synthetic data. Literally, there are several chairs inside images and the background of the images are usually complex. As a matter of fact, optical flow is aimed for estimating the moving displacement in  $X$  and  $Y$  axis of a pixel from first image to second image, so the designed network in training should learn and focus on the moving displacement of the pixels in images no matter how complex of the background is and what shape of the objects are.

As shown in Figure 5.1, the background are various and complicated and so are the moving objects. The moving objects contains several motions, e.g., rotation, translation, etc. Also, the background of each image pair also rotates and translates in a certain range.



<b>FlyingChairs Dataset Visualization</b>		
<b>Image1</b>	<b>Image2</b>	<b>Ground-Truth</b>

Figure 5.2: **FlyingChairs Dataset Visualization.** We visualize the FlyingChairs dataset which is an optical flow dataset. From left to right, there are the first image, second image, and corresponding optical flow ground-truth in each row.

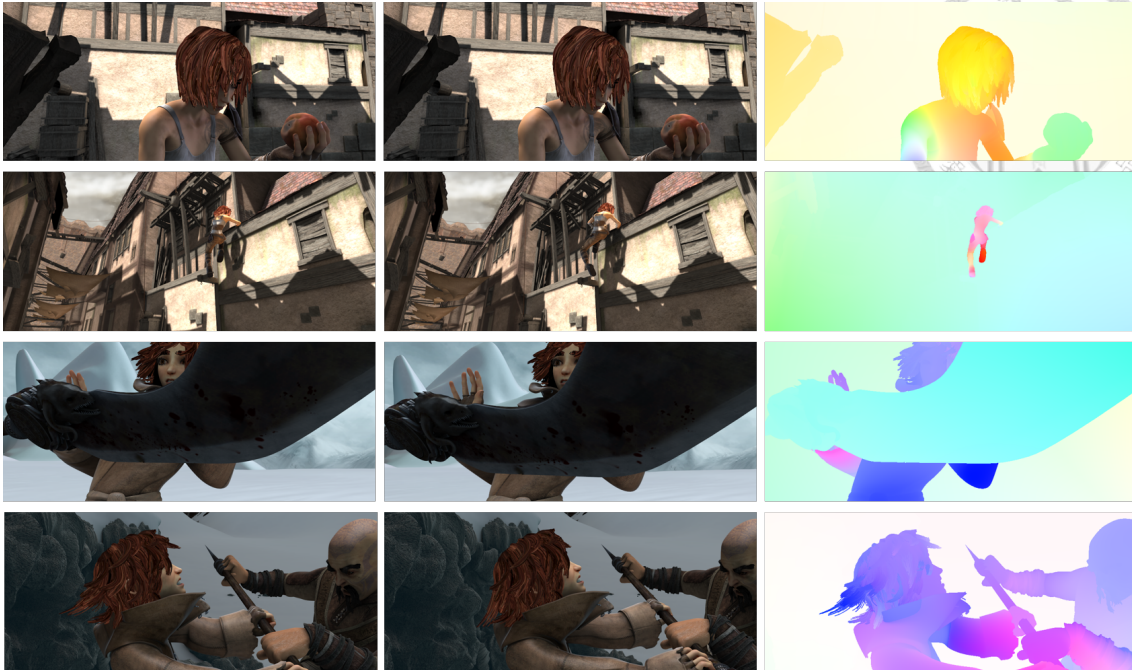


Figure 5.3: **Sintel dataset.** We visualize Sintel dataset which is an optical flow dataset. Each row represents a sequence of first image, second image, and corresponding optical flow from left to right.

Furthermore, the color and texture are almost roughly the same to the background and the shape of some chairs has really thin legs which is hard to detect and estimate the moving displacement. More samples can be seen in Figure 5.2.

In the light of this, FlyingChairs optical flow dataset is sufficient big and complex enough to prevent from overfitting. However, adding data augmentation is also crucial to improve the performance. Training this optical flow dataset is a challenging problem because of complex background and the color and texture of the chairs are similar to the background. Also, the training architecture are hard to design. More training details are elaborated in Chapter 5.

### 5.1.2 Sintel Dataset

Sintel dataset, [33], is an optical flow dataset. Sintel dataset utilizes the movie scenes of a movie named "Sintel" made by Blender, which are made by animation. As a consequence, the optical flow of two consecutive images is easily accessible because animation is made by computer which can be precisely positioned.

As shown in Figure 5.1.2, the moving displacement of arbitrary two consecutive images is relatively large. The characteristic of this dataset is that the background is complex and the moving objects are large. Since it is an animation movie, the dataset contains several relevant image pairs. Therefore, in the training procedure, the network is easily to overfit on the training data, so how to balance it is another question.

Looking into the images in this dataset. The image colors are saturated and the brightness is sufficiently light. Also, the images do not include the situation of changing camera perspective, so the optical flow estimation task of this dataset do not need to handle this kind of problem while in real world this is a big problem.

## **5.2 Action Recognition Dataset**

Action recognition is a challenging problem not only in the field of computer vision but also in the daily life. Identifying the action is helpful for observers to get the information of the status under surveillance system. There are two benchmark action dataset for action recognition which are UCF101 dataset and HMDB51 dataset. UCF101 dataset mostly focus on the outdoor activity, e.g., running, shooting, fielding, playing basketball, baseball, etc. Consequently, motions between each frame has small displacement which is not suitable to use FlyingChairs dataset in the pre-training stage.

### **5.2.1 UCF101 Dataset**

UCF101 is one of the competitive benchmark in the field of action recognition. Most actions in this dataset are outdoor activity. The actions included in this dataset can be categorized into five parts, which are

1. Human-Object Interaction
2. Body-Motion Only
3. Human-Human Interaction
4. Playing Musical Instruments





Figure 5.4: **UCF101**. UCF101 dataset is an action dataset which contains 101 action categories with 13320 total action video clips.

## 5. Sports

There are total three training testing splits for this dataset with total 13320 video clips with 101 action categories. These action categories are divided into 25 groups and there are 4 to 7 video clips of an action in each group. As shown in Figure 5.4, there are 101 action classes which sports action classes take the major proportion. Most action classes in this dataset include the motion of the whole human body, e.g., skiing, baby crawling, push ups, etc. Consequently, extracting meaningful and useful features from a human body in a video clip is crucial for action recognition.

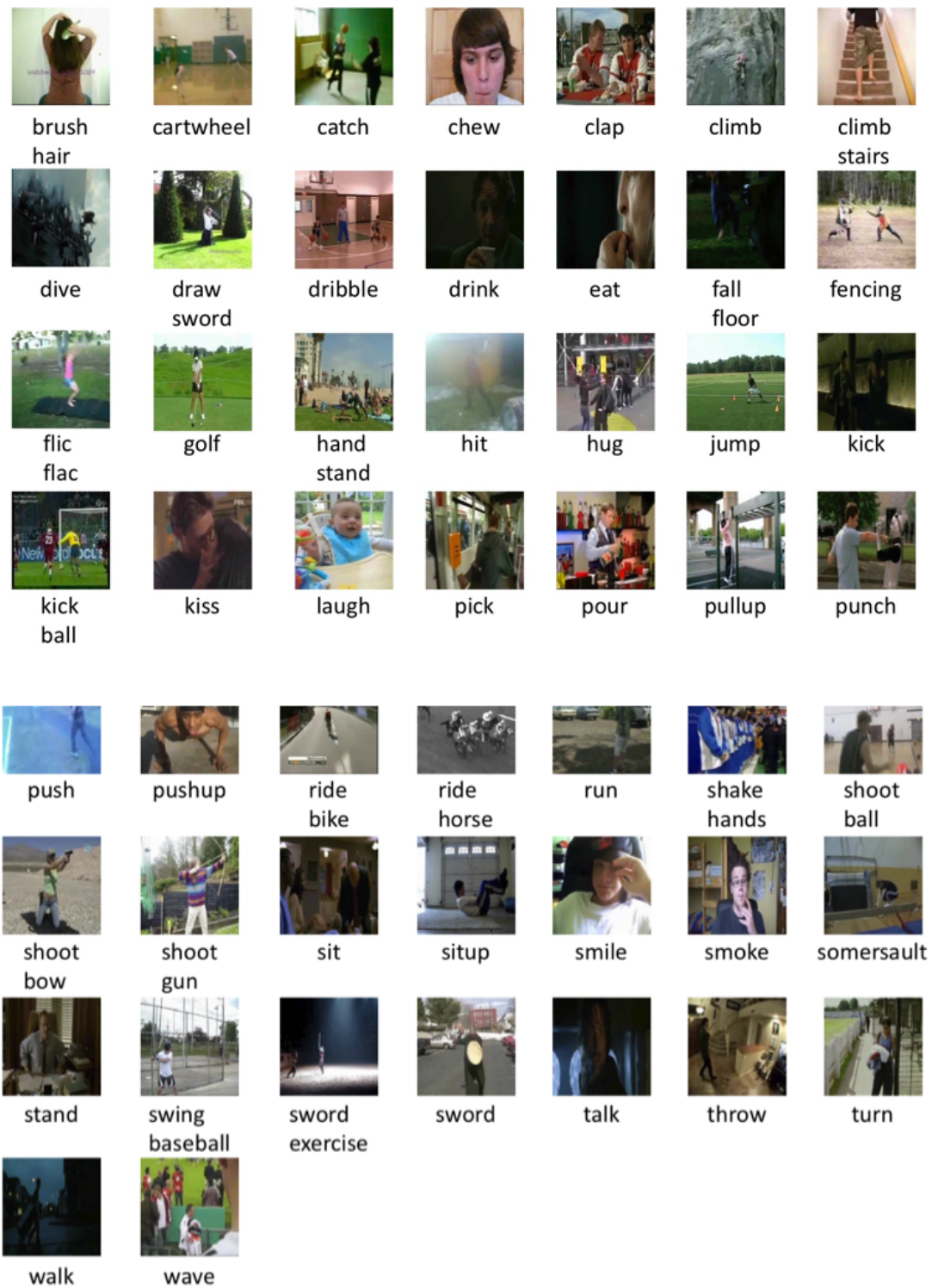
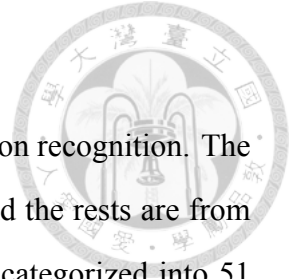


Figure 5.5: **HMDB51**. HMDB51 dataset is an action dataset which contains 51 action categories with 6766 total action video clips.



### 5.2.2 HMDB51 Dataset

HMDB51 is one of the competitive benchmark in the field of action recognition. The actions included in this dataset are collected mostly from movies, and the rests are from public databases. There are 6849 video clips in total which can be categorized into 51 action classes, and each action contains at least 101 video clips. Furthermore, HMDB51 can be categorized into five parts, which are

1. General facial actions
2. Facial actions with object manipulation
3. General body movements
4. Body movements with object interaction
5. Body movements for human interaction

As shown in Figure 5.5, most action classes in this dataset contains some fine-grained actions which are hard to recognize and some action classes

### 5.2.3 Discussion

UCF101 dataset and HMDB51 dataset are one of the most existing largest dataset for action recognition. Nevertheless, the training video clips in UCF101 dataset are highly related to background which leads to the consequence that the model only learns the relationship between action category and background rather than the action motion and action category. HMDB51 dataset has fewer video clips with less action categories, but the resolution quality is unstable which is hard for CNN to extract sufficient information for action recognition.

## 5.3 Optical Flow Estimation

We evaluate the performance of our proposed method, ResFlow, on estimating optical flow by using *EPE* which is described in 3.1. Using this error as an indicator is fairly



enough to speculate the quality of predicted optical flow. Particularly, we show the average  $EPE$  to do the comparison because each resolution at different stage is different. The average  $EPE$  can be written as follow,

$$EPE(U, V, U', V') = \frac{1}{h \times w} \sum_{i,j} \sqrt{(u_{i,j} - u'_{i,j})^2 + (v_{i,j} - v'_{i,j})^2} \quad (5.1)$$

, where  $h$  and  $w$  represents the image height and image width respectively. Also, we denote the average  $EPE$  as  $EPE$  for further usage. As a matter of fact, the smaller the  $EPE$  is, the better quality it is because the difference of the similarity is smaller. We choose FlyingChairs as our benchmark which is a crucial dataset made of synthetic data. Furthermore, FlyingChairs dataset has absolute ground-truth because it is a synthetic dataset. We compare our proposed ResFlow with state-of-the-art and show the experimental results in the following sections.

### 5.3.1 Comparison

We first compare our proposed ResFlow v1 and ResFlow v2 on FlyingChairs dataset. The difference between ResFlow v1 and ResFlow v2 is that they use different kind of residual unit as encoder and decoder. The performance of two proposed architecture can be seen in Table 5.1. From our perspective, ResFlow v2 outperforms ResFlow v1 with better result. We suspect that ResFlow v2 shows better results is due to the design of the residual block. The skip connection of the residual unit is different, one is after batch normalization layer while the other one is after convolution layer. As illustrated in [28], adding skip connection after convolution layer enlarge the features' variety so as to get better result. Although batch normalization layer gives benefit to training, it confines the output values in a certain range which cannot highlight the key features compared to other features.

Furthermore, the performance of both ResFlow v1 and ResFlow v2 outperform the state-of-the-art, FlownetS [18], FlownetC [18], and SpyNet [21]. Consequently, we figure out that using residual unit rather than simple convolution layer has improved optical

Table 5.1: **Optical Flow Estimation Comparison.** We evaluate the performance of optical flow estimation of our proposed ResFlow v1 and ResFlow v2 on FlyingChairs dataset and compare them with the state-of-the-art FlowNetS, FlowNetC, and SpyNet.

<b>Optical Flow Estimation Comparison</b>					
Architecture	FlowNetS	FlowNetC	SpyNet	ResFlow v1	ResFlow v2
<i>EPE</i>	2.86	2.61	2.63	2.41	2.29

flow estimation but requires some extra refinements as explained in Section 3.4. As shown in Table 5.2, the performance of ResFlow v2 outperforms that of ResFlow v1. Moreover, ResFlow v2 dominates ResFlow v1 at all stages. Noticeably, we have tried to use residual unit without adding extra upsample error, Equation 3.2, but found it hard to converge. We suspect that using batch normalization will rescale the values of each feature map after doing convolution which is contradicted to optical flow estimation due to the fact that the value of each pixel in an optical flow is a distance value. Therefore, we do not insert batch normalization after the convolution layer of predicting optical flow estimation in our decoder and the upsampled convolution layer as well. In the light of the characteristic of optical flow, we do not want to modify the value of the predicted optical flow tremendously.

### 5.3.2 Refinement

Referred to Section 3.3, we introduce a novel design which refines upsampled optical flow and builds a strong connection of optical flow estimation at each stage. As shown in Table 5.2, the performance of ResFlow v2 outperforms that of ResFlow v1 due to the fact that the smaller *EPE* is, the better it is. Also, ResFlow v2 dominates ResFlow v1 at all stages. Consequently, we conclude our ResFlow v2 is a well-designed architecture for optical flow estimation. As a matter of fact, the performance will catastrophically drop without adding the proposed refinement network, more particularly, the skip connection of optical flow. This also As shown in Figure. 5.7, it is obvious to observe that the optical flow estimation is improving gradually and is closer to the ground-truth. Firstly, using the smaller optical flow generated from previous layer, we up-sample this smaller opti-

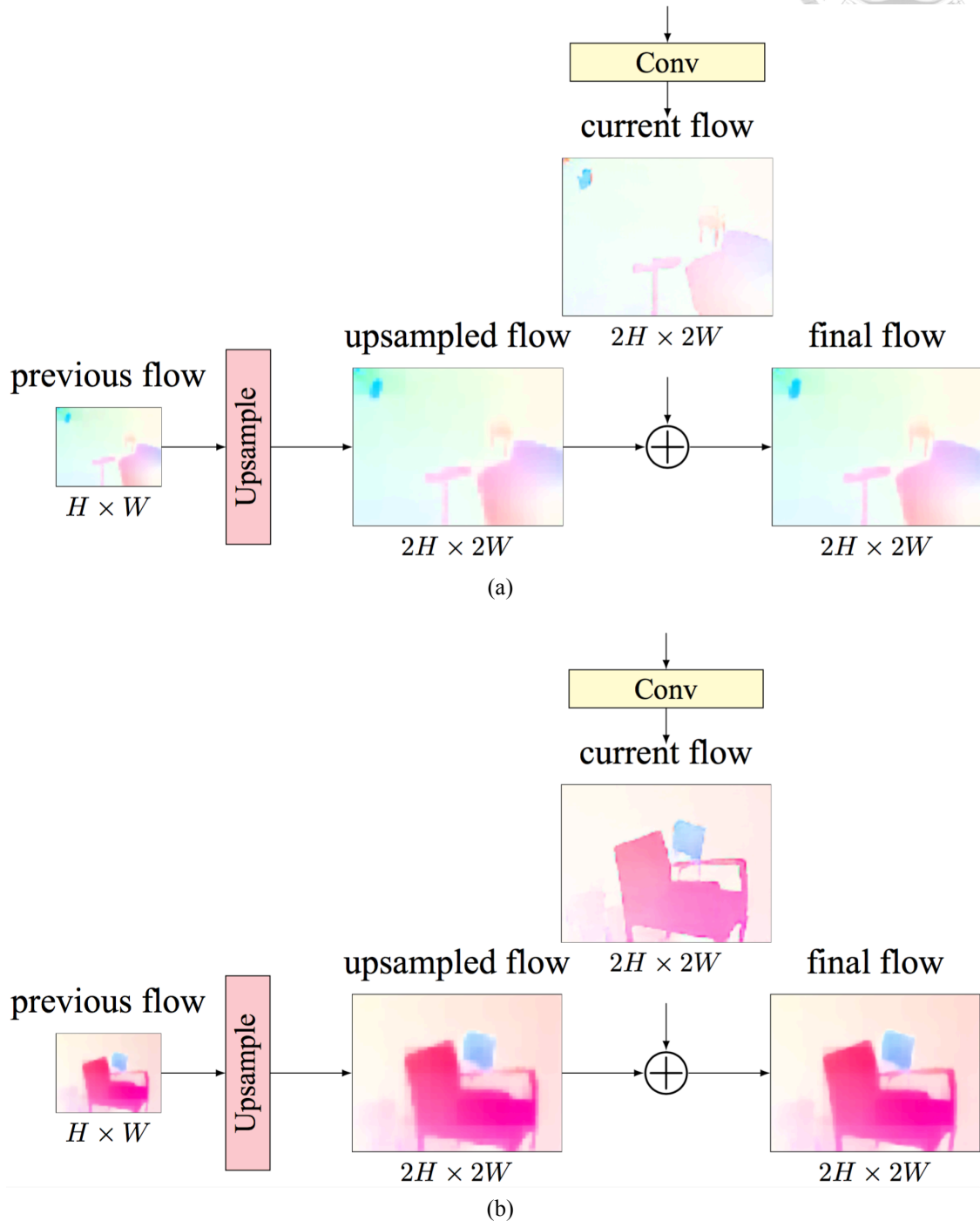


Figure 5.6: **Optical Flow Refinement Visualization.** The progress of optical flow refinement are shown as the figure.

Table 5.2: **Optical Flow Refinement Evaluation.** We evaluate the performance of optical flow estimation of our proposed ResFlow v1 and ResFlow v2 on FlyingChairs dataset at all stages. ResFlow v2 obviously outperforms ResFlow v1 due to the  $EPE$  is smaller.

<b>Optical Flow Refinement Evaluation</b>					
<i>FlyingChairs</i>					
Architecture	stage1	stage2	stage3	stage4	stage5
ResFlow v1	4.13	3.57	3.14	2.67	2.41
ResFlow v2	3.86	3.29	2.91	2.48	2.29

cal flow to be accordance with current predicted optical flow from concatenation layer. Afterward, we calculate the final optical flow using the weighted-sum mechanism of the up-sampled optical flow and predicted optical flow. Since the value of a pixel in an optical flow represents the moving displacement of that pixel among two images, it is reasonable to element-wisely sum these two optical flow together. Under the circumstance of eliminating the skip connection, we figure out that the  $EPE$  is relatively higher than adding the skip connection, so we finally add this skip connection into our network. Besides, skip connection is firstly proposed by ResNet which shows good performance. We take the advantages of the characteristic of adding skip connection into our design which builds a bridge between estimated optical flow at each stage. We also prove the robustness of using this design.

### 5.3.3 Sintel dataset

We visualize two version of optical flow estimation. One is that we pretrain our ResFlow on FlyingChairs dataset and directly predict optical flow estimation on Sintel dataset without finetuning. The other one is that we pretrain our ResFlow on FlyingChairs dataset and finetune ResFlow on Sintel dataset.

As shown in Figure 5.8, the predicted optical flow is slightly different to the ground-truth. Also, the fine-grained details are rough as well. For instance, in the third row, the predicted optical flow is similar to ground-truth. The weapon of the predicted optical flow is estimated well but the edge is a little coarse. Also, the human motion is not accurate enough compared to the ground-truth optical flow.












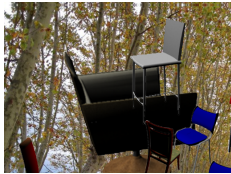

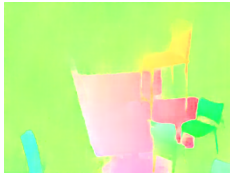




Optical Flow Estimation Evaluation			
Image1	Image2	Ground-Truth	Estimated
			
			
			
			

Figure 5.7: **FlyingChairs Visualization.** We visualize the optical flow estimation of our proposed ResFlow v2 on FlyingChairs dataset.

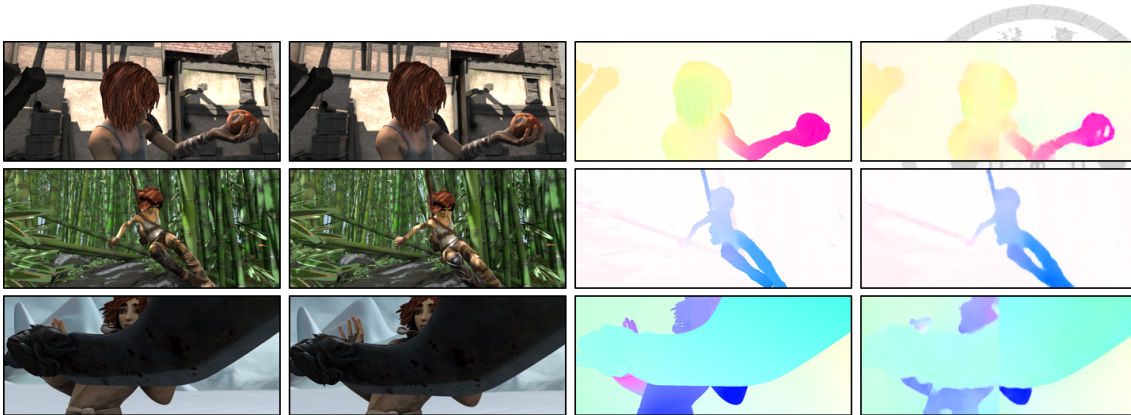
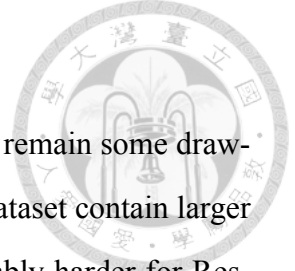


Figure 5.8: **Visualization of Sintel Optical Flow Estimation.** The images are the optical flow estimation on Sintel dataset without finetuning but pre-train on FlyingChairs.



Figure 5.9: **Visualization of Sintel Optical Flow Estimation.** The images are the optical flow estimation on Sintel dataset and finetuning on Sintel dataset.

In light of this, finetuning on Sintel dataset is important to get accurate optical flow. As shown in Figure 5.9, not only the human or object motion, but also the background translation are estimated pretty well. Looking into the predicted flow, some fine-grained details still is not good enough. In the second row, the predicted flow estimated by ResFlow is hard to estimate the hand motion. We suspect that this is because we use autoencoder to generate optical flow. After several convolution layer, the receptive field is growing, but only the first few convolution layers are able to capture fine-grained features. Our proposed method utilizes the forward pass to transmit the encoder features to the decoder part which enhances the ability of estimating optical flow. However, we only decode the predicted optical flow to the resolution of  $96 \times 128$  while the original resolution is equal to  $384 \times 512$ .



### 5.3.4 Remark

Although two versions of ResFlow show promising result, it still remain some drawbacks. Due to the fact that the training image pairs in FlyingChairs dataset contain larger displacement, predicting small displacement optical flow is comparably harder for ResFlow. We experiment our ResFlow on ChairsSDHom dataset [20] which is a dataset for estimating small displacement optical flow. The predicted optical flow is shown in Table 5.7. Via visualization, we are able to see the robustness of ResFlow v2 on estimating optical flow. Comparing ground-truth and predicted flow, we can see that the predicted flow is much similar to ground-truth, but some fine-grained details still remains improvement. As for training small displacement optical flow, we adopt the same training procedure described in [20] on ChairsSDHom dataset. However, we found out that replacing simple convolution layer with residual unit does not improve the performance, but gets a worsen result. We believe that this is contributed to batch normalization layer which rescale the values that influences the predicted optical flow. Due to the fact that the values of the small displacement optical flow are normally in range of 0.001 to 2, the values after batch normalization layer are normally between -1.0 to 1.0. As a consequence, it is hard to predict the values of small displacement optical flow which overlaps the input features significantly.

## 5.4 Action Recognition

In this section, we conduct our ResFlow on two action recognition dataset, UCF101 and HMDB51. Firstly, we show the importance of spatiotemporal feature in Section 5.4.1 and show the effective of SPM. We compare our ResFlow with the state-of-the-art in Section 5.4.2. We also compare ResFlow with the state-of-the-art on multitasking problem in Section 5.4.3. Lastly, we remains some discussion in Section 5.4.4.

Table 5.3: **Spatiotemporal Features Impact.** We directly use ResFlow which is pre-trained on optical flow dataset to predict action recognition in condition of fixing the convolution layers. We evaluate the performance of ResFlow on UCF101 dataset as well as HMDB51 dataset and compare ResFlow with the state-of-the-art which are trained from scratch.

<b>Sequentially Pooling Mechanism Evaluation</b>		
Architecture	UCF101	HMDB51
C3D Scratch	45.3%	-
ResFlow Scratch	51.1%	22.9%
Two-Stream Scratch	52.3%	-
VGG-M-2048 Scratch [13]	52.9%	-
Flownet	54.5%	27.6%
ResFlow v2 (w/o SPM)	60.2%	30.5%
ResFlow v2	62.3%	31.8%

#### 5.4.1 Spatiotemporal Feature

Only pre-trained with optical flow dataset, our proposed network, ResFlow, encodes local spatiotemporal features at each time step of a video clip. We examine our robustness of proposed Sequentially Pooling Mechanism which computes the confidence score of each spatiotemporal feature and updates the global spatiotemporal feature for action recognition on both UCF101 dataset and HMDB51 dataset. As shown in Table 5.3, ResFlow v2 outperforms other state-of-the-art due to the fact that ResFlow v2 has pre-trained knowledge of extracting spatiotemporal feature. This comparison shows that spatiotemporal feature extracted by autoencoder is extremely important for action recognition.

Moreover, we use average pooling on the time axis as the baseline of our network and compare with the one adding Sequential Mechanism, denoted as ResFlow v2 (w/o SPM) and ResFlow v2, respectively. The experimental results show that SPM is able to judge the confidence score of each spatiotemporal feature from each time step and improves the recognition accuracy from 60.2% to 62.3%.



Table 5.4: **Action Recognition Evaluation.** We finetune our ResFlow thoroughly, which is pretrained on optical flow dataset, on UCF101 dataset. We evaluate the performance of ResFlow on UCF101 dataset and compare ResFlow with the state-of-the-art.

<b>Action Recognition Comparison</b>	
Architecture	UCF101
ResNet-18 Scratch [13]	51.3%
VGG-M-2048 Scratch [13]	52.9%
Flownet + <i>ft</i>	66.0%
Flownet + SPM + <i>ft</i>	68.2%
Actionflownet-2f	71.0%
Two-Stream(Spatial) [23]	73.0%
ResFlow v2 (w/o SPM) + <i>ft</i>	72.0%
ResFlow v2 + <i>ft</i>	73.6%

#### 5.4.2 Comparison

We compare our ResFlow with state-of-the-art on UCF101 dataset. Due to the fact that our novel design for action recognition only consider a pair of image at each time step, we compare with a competitive baseline model, Two-Stream [23]. As shown in Table 5.4, we outperform the state-of-the-art. Moreover, we compare ResFlow v2 with the baseline model Flownet+*ft* which is finetuned on UCF101 dataset. We show that ResFlow improves the action recognition accuracy by 7.6% which prove the fact that adding residual unit as well as SPM model help the network to get better result. Also, with finetuning the convolution layers on UCF101 dataset, the action recognition accuracy improves from 62.3% to 73.6%, which grows by 11.3%.

Besides, we prove that our new design Sequentially Pooling Mechanism is able to aggregate features from each time step into global features. As we can see in the table, adding SPM to Flownet architecture is able to increase the accuracy by 2.2% in UCF101 dataset. As for our ResFlow, it improves by 1.6% while having SPM compared to using mean-pooling only. As a matter of fact, Two-Stream [23] uses pre-trained model on a large ILSVRC dataset which is an object recognition dataset. Therefore, Two-Stream has pre-knowledge of capturing details of objects. However, our ResFlow has only pre-trained on a relatively tiny dataset to capture details of motion. Due to the characteristic

Table 5.5: **Multitasking comparison.** Multitasking comparison on optical flow estimation and action recognition.

<b>Multitasking Comparison</b>		
Architecture	ResFlow (ours)	Actionflownet-2f
UCF101 dataset	73.6%	71.0%
HMDB51 dataset	43.5%	42.6%
FlyingChairs dataset	2.29 (EPE)	-
Sintel Final dataset	5.43 (EPE)	9.12 (EPE)

of UCF101 dataset, the action category of each video clip has strong dependency on the background. In the light of this, using pre-trained model on ILSVRC dataset [34], Two-Stream [23] predicts action category based on the background in each frame of a video clip. However, this phenomenon contradicts to the concept of capturing motion segments to predict action recognition. On the contrary, we pre-train our ResFlow on a small optical flow dataset, *FlyingChairs* dataset, which captures details of motion with spatiotemporal features encoded in each time step. Leveraging relatively tiny dataset, our ResFlow still outperforms Two-Stream (Spatial) [23] in *UCF101* dataset.

### 5.4.3 Multitasking

Due to the fact that both of Actionflownet-2f and our ResFlow are able to estimate optical flow and predict action recognition at the same time, we compare ResFlow with Actionflownet-2f on multitasking of both tasks including optical flow estimation and action recognition as shown in Table. 5.5. For action recognition task, the performance of our ResFlow outperforms that of Actionflownet-2f on both *UCF101* dataset and *HMDB51* dataset by 2.6% and 0.9%, respectively. For optical flow estimation task, the *EPE* of ResFlow is obviously lower than that of Actionflownet-2f on Sintel Final dataset. In the light of this, ResFlow shows strong robustness on both optical flow estimation task and action recognition task. The design of our refinement work which builds connection between two neighbored stage helps ResFlow to estimate accurate optical flow. Furthermore, the design of SPM which aggregates spatiotemporal features from each time step into global spatiotemporal features gives benefit to predict accurate action category.

#### 5.4.4 Remark

We visualize our optical flow prediction of action sequence of UCF101 dataset which shows promising results. In Figure 5.10 and Figure 5.11, Resflow is able to capture human motion. Moreover, the quality of predicted optical flow is quite well. However, there is still some unrecognizable predicted optical flow in Figure 5.12 which happens when the camera motion is too strong. Looking into both Figure 5.10 and Figure 5.11, the camera is really stable without any vibration. However, we can see that the camera moves vibrationally in the first few image pairs which cause ResFlow to predict the scene motion that visualize as all green in the predicted optical flow image as shown in Figure. 5.12.

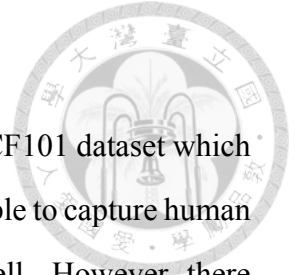




Figure 5.10: **Optical Flow Refinement Visualization.** We visualize the optical flow estimation of an action, *playingviolin*, in the figure. The camera is stable while only the human and the violin is moving.



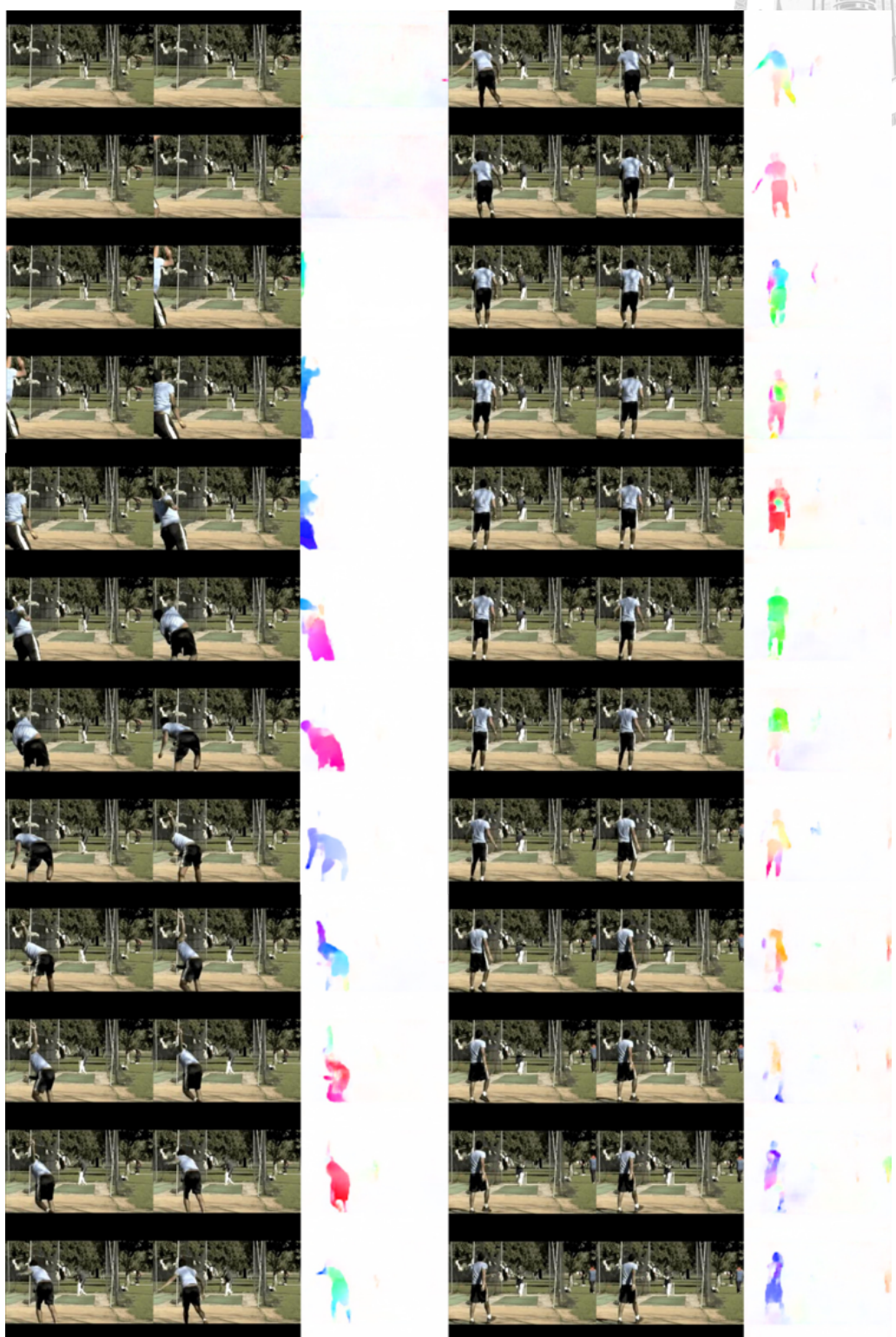
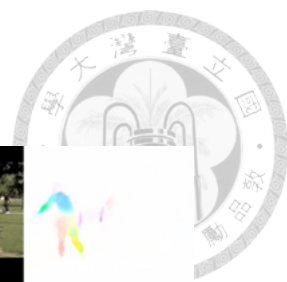


Figure 5.11: **Optical Flow Refinement Visualization.** We visualize the optical flow estimation of an action, *batting*, in the figure. The camera is stable while only the human is moving.

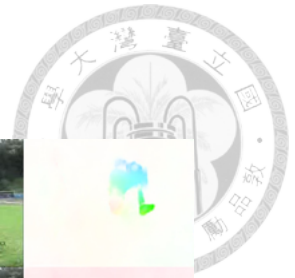


Figure 5.12: **Optical Flow Refinement Visualization.** We visualize the optical flow estimation of an action, *throwingball*, in the figure. The camera is moving which cause camera motions while the human is moving as well.



## Chapter 6

### Conclusion


We propose a novel architecture, ResFlow, for estimating optical flow and predicting action recognition simultaneously. We add an additional refinement network to ensure that sufficient and crucial features are extracted for estimation of optical flow by bridging two stages especially. Due to the characteristic of autoencoder, spatiotemporal feature at each time step is extracted automatically. We proposed Sequentially Pooling Mechanism to aggregate the extracted spatiotemporal feature at each time step into global spatiotemporal feature. That is to say, ResFlow sequentially generates condensed feature based on previous knowledge under a fair criterion to judge the importance of each spatiotemporal feature at each time. Overall, we figure out a new viewpoint to solve action recognition problem task via multi-tasking. Furthermore, ResFlow can be easily integrated with other works, *i.e.*, Two-Stream network, so as to boost the performance. Also, ResFlow is capable of generating optical flow for other applications.




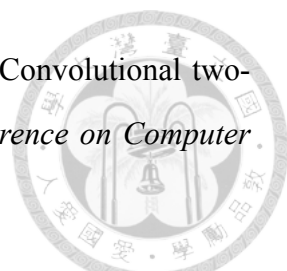
## Reference

- [1] Ivan Laptev and Tony Lindeberg. Space-time interest points. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 432–439. IEEE, 2003.
- [2] Amlan Kar, Nishant Rai, Karan Sikka, and Gaurav Sharma. Adascan: Adaptive scan pooling in deep convolutional neural networks for human action recognition in videos. *CVPR*, 2017.
- [3] X. Yan, S. Hu, and Y. Ye. Multi-task clustering of human actions by sharing information. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4049–4057, July 2017.
- [4] Rohit Girdhar, Deva Ramanan, Abhinav Gupta, Josef Sivic, and Bryan Russell. ActionVLAD: Learning spatio-temporal aggregation for action classification. In *CVPR*, 2017.
- [5] Y. Wang, M. Long, J. Wang, and P. S. Yu. Spatiotemporal pyramid network for video action recognition. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2097–2106, July 2017.
- [6] C. Feichtenhofer, A. Pinz, and R. P. Wildes. Spatiotemporal multiplier networks for video action recognition. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7445–7454, July 2017.



- 
- [7] Z. Lan, Y. Zhu, A. G. Hauptmann, and S. Newsam. Deep local video feature for action recognition. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1219–1225, July 2017.
- [8] Christoph Feichtenhofer, Axel Pinz, and Richard Wildes. Spatiotemporal residual networks for video action recognition. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3468–3476, 2016.
- [9] I. C. Duta, B. Ionescu, K. Aizawa, and N. Sebe. Spatio-temporal vector of locally max pooled features for action recognition in videos. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 00, pages 3205–3214, July 2017.
- [10] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497, Dec 2015.
- [11] Huijuan Xu, Abir Das, and Kate Saenko. R-c3d: Region convolutional 3d network for temporal activity detection. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.
- [12] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? *arXiv preprint*, arXiv:1711.09577, 2017.
- [13] Joe Yue-Hei Ng, Jonghyun Choi, Jan Neumann, and Larry S. Davis. Actionflownet: Learning motion representation for action recognition. *CoRR*, abs/1612.03052, 2016.
- [14] A. Burton and J. Radford. *Thinking in Perspective: Critical Essays in the Study of Thought Processes*. Psychology in progress. Methuen, 1978.
- [15] D.H. Warren and E.R. Strelow. *Electronic Spatial Sensing for the Blind: Contributions from Perception, Rehabilitation, and Computer Vision*. Nato Science Series E.: Springer Netherlands, 1985.

- 
- [16] Jean yves Bouguet. Pyramidal implementation of the lucas kanade feature tracker. *Intel Corporation, Microprocessor Research Labs*, 2000.
- [17] Gunnar Farneback. Two-frame motion estimation based on polynomial expansion. In *Proceedings of the 13th Scandinavian Conference on Image Analysis, SCIA'03*, pages 363–370, Berlin, Heidelberg, 2003. Springer-Verlag.
- [18] Alexey Dosovitskiy, Philipp Fischery, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning optical flow with convolutional networks. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 2758–2766, Washington, DC, USA, 2015. IEEE Computer Society.
- [19] Pierre Baldi. Autoencoders, unsupervised learning and deep architectures. In *Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning Workshop - Volume 27, UTLW'11*, pages 37–50. JMLR.org, 2011.
- [20] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017.
- [21] Anurag Ranjan and Michael Black. Optical flow estimation using a spatial pyramid network. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Piscataway, NJ, USA, July 2017. IEEE.
- [22] Yi Zhu, Zhenzhong Lan, Shawn Newsam, and Alexander G. Hauptmann. Guided Optical Flow Learning. *arXiv preprint arXiv:1702.022952*, 2017.
- [23] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 568–576. Curran Associates, Inc., 2014.

- 
- [24] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [25] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3304–3311, June 2010.
- [26] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, June 2014.
- [27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778. IEEE Computer Society, 2016.
- [28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *ECCV (4)*, volume 9908 of *Lecture Notes in Computer Science*, pages 630–645. Springer, 2016.
- [29] Vicky Kalogeiton, Philippe Weinzaepfel, Vittorio Ferrari, and Cordelia Schmid. Joint learning of object and action detectors. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2001–2010, 2017.
- [30] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37, ICML'15*, pages 448–456. JMLR.org, 2015.
- [31] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster,

Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from [tensorflow.org](http://tensorflow.org).

- [32] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.
- [33] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, *European Conf. on Computer Vision (ECCV)*, Part IV, LNCS 7577, pages 611–625. Springer-Verlag, October 2012.
- [34] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.