

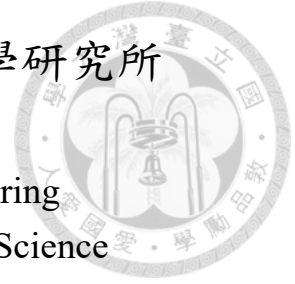
國立臺灣大學電機資訊學院電信工程學研究所

碩士論文

Graduate Institute of Communication Engineering  
College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis



藉由多邊界盒多任務學習網路辨識遠距離動作  
Recognizing Distant Actions via Multi-box Multi-task Networks

吳兆倫

Chao-Lun Wu

指導教授：陳銘憲博士

Advisor: Ming-Syan Chen, Ph.D.

中華民國 107 年 7 月

July, 2018

國立臺灣大學碩士學位論文  
口試委員會審定書

藉由多邊界盒多任務學習網路辨識遠距離動作  
Recognizing Distant Actions via Multi-box Multi-task  
Networks

本論文係吳兆倫君 (R04942143) 在國立臺灣大學電信工程學研究所完成之碩士學位論文，於民國 107 年 7 月 24 日承下列考試委員審查通過及口試及格，特此證明

口試委員：

陳銘堯

(簽名)

(指導教授)

邱震翽

李宜修

楊得年

陳怡伶

所長

吳宗霖

(簽名)



## Acknowledgments

Pursuing master degree is a long and challenging journey. First of all, I want to appreciate my advisor professor Ming-Syan Chen. He gives me many useful suggestions and lots of help. Without his support, it's impossible for me to finish this thesis. Also, I want to thank K. T. Lai and C. K. Chou, who give me insightful ideas and profound discussions. They help me complete this work. Furthermore, the laboratory members motivate me to perform this research. They create a great environment in which I can easily focus on my thesis. I really appreciate them. Moreover, when I was in difficult, my undergraduate friends keep encouraging me. They give me confidence to get through the hard times. Finally, I want to thank my family, who support me spiritually and financially. They teach me the manner of dealing with people, helping me pursue the master degree.



## 中文摘要

無人機的技術在近幾年有著突破性的進展，並且在諸多領域有豐富的應用如監視、救援、運輸以及軍事方面等等。本研究的目標是建立一個能夠在無人機上偵測、辨識事件的模型。這個研究問題困難的部分有兩點：第一，無人機相關的錄影資料非常稀少，要能夠以少量資料訓練出一個泛化能力高的模型十分困難。第二，由於無人機的位置通常離地面較遠，拍攝到的人物動作占畫面的比例很小，會令模型難以辨認人物動作。為了解決這些問題，我們提出了兩步驟的模型。首先先以 SSD 偵測出人物的位置，之後再藉由多任務學習架構，跟大型人物動作資料庫一起訓練的模型來辨識無人機影像中人物的動作。我們以自己提出的無人機人物動作影像資料來驗證我們的模型。這個影像資料包含 14 種類型的人物動作。實驗結果說明我們提出的方法可以增加無人機影像的人物動作辨識率。

關鍵字 - 動作辨識, 卷積類神經網路, 多任務學習



# Abstract

The technology of drone has advanced significantly during the last few years, which enables drones to be deployed in many tasks including video surveillance, search and rescue, last-mile delivery and military operation. The great potentials attract many researchers to study visual recognition technologies for drone, e.g. object detection in aerial images. However, there is not much research related to action recognition in drone videos. In this thesis, we aim to develop a real-time action detector of drone that can recognize complex human actions such as running, eating, walking, etc. Action recognition in drone is a challenging task due to the following reasons. First, there is no large-scale action dataset of drone, and the scarcity of training data makes learning accurate neural networks difficult. Second, the actions happen at a distance and are hard to be localized. To address this first issue, we propose a multi-box multi-task network architecture for recognizing actions at a distance. The multi-box network is used to generate human location proposal, and the action recognition network is then applied to the proposed locations to detect actions. In terms of the data scarcity, we attach this problem by leveraging the existing large human action databases with multi-task learning. To evaluate the effectiveness of our method, we create a new drone action dataset with 138 videos and 14 different distant actions. Experimental results show that our proposed method can increase the action recognition rate in drone.

Keywords - Action Recognition, Convolutional Neural Networks, Multi-

task Learning





# Contents

口試委員會審定書	i
Acknowledgments	ii
中文摘要	iii
Abstract	iv
Contents	vi
List of Figures	viii
List of Tables	x
<b>1 Introduction</b>	<b>1</b>
<b>2 Related Work</b>	<b>4</b>
2.1 Convolutional Neural Networks for Image Recognition . . . . .	4
2.2 Convolutional Neural Networks for Action Recognition . . . . .	5
2.3 Transfer Learning for Video Recognition . . . . .	7
<b>3 Proposed Method</b>	<b>8</b>
3.1 Preliminaries . . . . .	8
3.2 Training Framework . . . . .	9
3.2.1 Detection Step . . . . .	9
3.2.2 MTL Step . . . . .	11

<b>4 Experiment</b>	<b>13</b>
4.1 Datasets . . . . .	13
4.1.1 Large Scale Human Action Datasets . . . . .	13
4.1.2 Drone Dataset . . . . .	15
4.2 Experiment Settings . . . . .	17
4.2.1 Environment . . . . .	17
4.2.2 Training Details . . . . .	17
4.2.3 Testing Details . . . . .	20
4.3 Results . . . . .	20
4.4 Discussion . . . . .	24
<b>5 Conclusion</b>	<b>27</b>
<b>Bibliography</b>	<b>28</b>







# List of Figures

1.1	Examples from Drone dataset. Each picture represents an action category.	2
3.1	An illustration of our proposed method.	10
3.2	In standard action recognition problem, images are processed with data augmentation directly.	10
3.3	Detection step we proposed.	11
3.4	Multitask learning framework.	12
4.1	HMDB51 dataset.	14
4.2	UCF101 dataset.	14
4.3	Duration of each action in Drone dataset.	16
4.4	Examples recorded by Bebop2.	16
4.5	Examples from Dronestagram.	17
4.6	Directly training on scarce data.	18
4.7	Training on both datasets in the same time in order to show the effectiveness of MTL.	19
4.8	Training on drone dataset with the detection step in order to show the effectiveness of this step.	19
4.9	Complete form.	20
4.10	For fairness, we use the same setting to test our model.	21
4.11	An illustration to show how accuracy is evaluated.	21
4.12	HMDB51 performance versus weights of loss functions.	24
4.13	Failed cases for detectors.	25

4.14 An example of optical flows from drone videos. . . . . 26





# List of Tables

3.1	Notation Table . . . . .	9
4.1	Action categories of drone dataset . . . . .	15
4.2	Evaluation of CNN models on UCF101 . . . . .	22
4.3	Evaluation of CNN models on HMDB51 . . . . .	22
4.4	Evaluation of MTL on HMDB51 dataset . . . . .	23
4.5	Evaluation of Two-stream on HMDB51 dataset under MTL framework . . . . .	23
4.6	Evaluation of the detection step and the MTL step on Drone dataset . . . . .	23



# Chapter 1

## Introduction

Unmanned aerial vehicles (UAV) have important applications in robotics and automation control. Huge progress has been made for drone technology in recent years. Theoretically, drones can be used to accomplish many advanced jobs. For instance, drones can patrol streets, scout in risky areas, deliver goods efficiently or detect terrorists' locations. In this work, our goal is to develop an real-time action detector that enables drone to recognize human actions for surveillance purpose. If there are dangerous events or illegal activities. Drones with action recognition technologies can detect those events and then report messages to the police immediately. Action recognition is a popular research field in computer vision. Recently, many powerful recognition model based on convolutional neural networks (CNN) are proposed [6, 18, 24]. The objective of action recognition is to recognize human activities in videos and classify their actions. There are many applications which utilize action recognition methodologies. To perform this task, we collected human action videos of drones through Bebop2 and the Internet. Then, we studied in these videos to find effective models for this task.

However, while collecting drone action videos, we observe that recent action recognition methodologies cannot be applied directly to this task for two reasons. First, recent advances are based on massive human recognition datasets such as UCF101 [28] and HMDB51 [17]. These models don' t need to focus on human' s location because the size of human is quite large in video frames. Moreover, the camera of a drone is far from human and objects, the size of human is relatively small in drone videos. Therefore, the back-

ground information not related to human can reduce recognition performance of models. Second, the video data of drone are extremely scarce. If we train the data on CNN models, the model will easily overfit the drone videos.

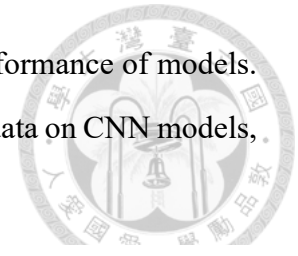
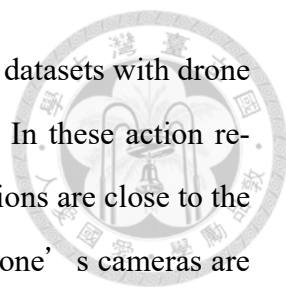


Figure 1.1: Examples from Drone dataset. Each picture represents an action category.

Our idea to solve the above problems is to design a two-step model, and these steps include human detection and multitask networks. The first step is the detection step. This step is to localize human in video frames. Training procedure used in action recognition problems takes the whole frame as input. The detection step in our method can help training model focus on human bodies and decrease interference from background noise. We construct an architecture based on Single-Shot Multibox Detector (SSD) [22] as our detection step algorithm. SSD is state-of-the-art model in image detection. It takes both region proposal and object recognition as a regression problem. The model gives possible object locations and confidence scores in the same time. The second step is a network architecture depending on multitask learning (MTL). The reason we utilize this framework is that the size of drone data is extremely limited. There are only few drone datasets in computer vision research [1]. In addition, the datasets we found are not suitable for our task. Although we collect many drone data by ourselves, the data size is still not enough for training an action recognition model well. To solve this, we need more related data



to support the model. Nonetheless, we cannot directly combine these datasets with drone data because the features of these datasets are somewhat different. In these action recognition datasets, the cameras' position is static and the human actions are close to the cameras; while the cameras of drones are moving and the level of drone's cameras are usually higher than human. An effective way to train different domain datasets is to apply a multitask network. This network allows different domain datasets to be trained together. The dataset from different tasks uses its own loss function, while all tasks share the same network parameters. With this mechanism, the task with massive data can increase the performance of the task with extremely few data.

To evaluate our model, we create a drone dataset for our experiments. This dataset includes 14 human actions. These categories include some common events like walking, running, smoking and eating. The duration of each action is 100 second on average. We compare several models on this dataset to show that our proposed method is effective.

In this work, we have the following contributions. First, we propose a new learning framework for action recognition model that increases recognition accuracy of drone videos. This framework includes a detection mechanism based on SSD. Moreover, we deploy a multitask network to increase generalization and avoid overfitting on our drone dataset. Second, we present a new dataset for this research problem. This dataset contains 14 actions. Experimental results show that our proposed method can improve the recognition accuracy on this dataset.

The structure of this thesis is organized as follows. Chapter 2 introduces the research and techniques related to our work. Chapter 3 formally describes background knowledge, the problem and the proposed method. Chapter 4 shows experimental results of our method and gives some observations and discussions. Chapter 5 summarizes our work and future plans.



## Chapter 2

### Related Work

In this chapter, we introduce some works related to our research. There are 3 sections describing main stream research in this field. First, we recap some history and important techniques about deep convolutional neural networks for image recognition. Then, we simply show evaluations of video recognition methods with deep neural networks. Finally, we review recent advances about transfer learning in video recognition.

#### 2.1 Convolutional Neural Networks for Image Recognition

The theory of convolutional neural networks was developed before 20th century and a few networks have been proposed recently. The first convolutional network is LeNet, containing a 5-layer convolutional neural network to solve handwritten digits recognition problems [19]. Recently, AlexNet having an 8-layer structure outperformed traditional image classification methods in ImageNet competition [16]. Furthermore, there are some applications and works based on AlexNet [42]. To construct a deeper network, VGG uses a 16-19 weighted layer convolutional network to reach a higher performance in image recognition tasks [27]. GoogLeNet with 22 layers uses a module called inception combining three different convolutional structures [33]. Then, some works investigate and improve GoogLeNet [12, 32, 34]. However, these models compared to newer convolutional net-

works are relatively shallow.

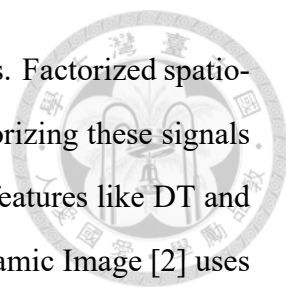
When a neural network is going deeper, there will be a gradient vanishing problem. This is because in each iteration of backpropagation, gradient components will be smaller due to the multiplication from the weights of the latter layer. If there are too many layers, gradient components are gradually close to zero; thus, it is more difficult to train the deeper networks. In order to train a deeper network model, ResNet proposed a skip connection to avoid this problem during training [10]. A skip connection simply adds inputs from the previous layer to outputs of the latter layer. These connections allow the networks to remain gradient components during the backpropagation and easily to be trained. ResNet with 152-layer structure is not only deeper but also more accurate than previous models. ResNet is investigated and improved in some works [41].

Recently, DenseNet is proposed and it has an architecture of densely connected convolutional layers [11]. A densely connected convolutional layer means that every convolutional layer is connected to each other; thus, a  $L$ -layer network has  $L(L+1)/2$  connections. Unlike ResNet using adding connections to pass features forward, DenseNet uses concatenation to combine features and passes them forward. These connections make DenseNet reach higher classification accuracy with fewer parameters and a deeper architecture. The deepest DenseNet is 250 layers. So far, the work based on DenseNet is few because DenseNet is relatively new. The related research includes Tiramisu [13] and DSOD [25].

## 2.2 Convolutional Neural Networks for Action Recognition

With the impressive performance of CNN, many studies applied CNN to action recognition problem. Before CNN-based model was developed widely, dense trajectories (DT) [37] and improved dense trajectory (iDT) [38] were the best features to recognize human actions in massive videos. [14] first studied and proposed CNN model for action recognition problem. Two-stream networks [26] use two independent CNN with different features that are frames colors and optical flows. 3D ConvNets [35] use three-dimensional

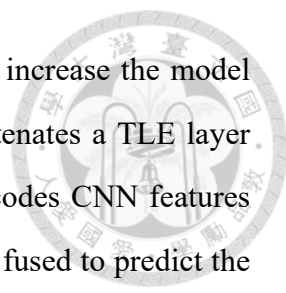




convolutional kernels to learn spatiotemporal features of action videos. Factorized spatio-temporal convolutional networks [31] take video as 3D signals, factorizing these signals to learn spatio-temporal features. TDD [39] combines hand-crafted features like DT and deep-learned features from CNN to learn video representations. Dynamic Image [2] uses rank-pooling and CNN to summarize video sequences from RGB frames. [45] detects key volume to remove irrelevant features of videos. LTC [36] extends CNN structure to learn long-term temporal features from videos.

Some works proposed RNN-based (Recurrent Neural Network) models or hybrid models. [29] uses LSTM to learn representations from video sequences. [23] extracts CNN features from raw frames and optical flows, aggregating these features by LSTM. [5] constructs a hybrid model containing unsupervised layers and supervised layers and utilizing iDT features. RNN-FV [20] learns fisher vectors (FV) from videos by applying LSTM.

Most recent models based on two-stream networks can reach impressive results in action recognition tasks. Two-stream networks contain a spatial network and temporal network, utilizing both spatial and temporal features of action videos. Spatial features are RGB of each pixel in videos, and temporal features are stacked optical flows of multiple frames in videos. The structure of both networks is AlexNet. Based on Two-stream network and VGG networks, [9] investigated fusion method and fusion layers between both networks. In addition, this research extends 2D kernels in temporal networks to 3D filters to handle input video frames in a serial time. TSN [40] provides a consensus function to learn information from long-range temporal features in videos and avoid overfitting due to the limited data size. The research tested different network structures, finding that partial BN-Inception with dropout reaches the best performance for TSN. In addition, the research applied not only RGB frames and optical flows but also RGB difference and warped optical flows. The result shows that RGB frames, optical flows and warped optical flows are the best combination. ST-ResNet [7] redesigned Two-stream networks with ResNet. The network contains an appearance stream and motion stream originated from spatial networks and temporal networks. A motion stream uses skip connections of ResNet to connect a appearance stream. ST-ResNet is further improved by another research [8].



Recently, some advances proposed new aggregation methods to increase the model performance considerably. Time Linear Encoding (TLE) [6] concatenates a TLE layer after spatial net and temporal net. A TLE layer aggregates and encodes CNN features extracted from multiple video segments. Then, prediction scores are fused to predict the action. FV-VAE [24] proposed a new deep generative model to generate and aggregate fisher vectors and variational auto-encoding (VAE). Deep local features (DOVF) [18] extracted from trained CNN layers can be aggregated to get higher performance.

## 2.3 Transfer Learning for Video Recognition

Some advances aim to transfer knowledge from different domain sources to improve video recognition accuracy with transfer learning methodology. [3] uses domain adaptation to transfer knowledge on tennis video annotation problems. [44] provided a discriminative cross domain dictionary to transfer information from HMDB51 [17] dataset to UCF101 [28] dataset domain. [4] utilized multitask learning including virtual world data and real world data with TSN to increase overall video recognition performance. Transferring knowledge of image data is another method to enhance video recognition accuracy. Sharing CNN weights between large scale image data and limited video data is an efficient way to augment video recognition task with scarce training dataset [30]. [21] proposed an attention-based CNN model to transfer web images to action recognition task.



## Chapter 3

# Proposed Method

In this chapter, we propose the method to solve action recognition of drones. We first explain the notations and definition of the problem in the thesis. After that, we present the training framework including 2 crucial steps in this method.

### 3.1 Preliminaries

We list all notations and the problem definition in the thesis. In transfer learning framework, we have a source domain  $\mathcal{D}_S$  and a target domain  $\mathcal{D}_T$ .  $\mathcal{D}_S$  is a set of source data features in  $d_S$ -dimensional space and their corresponding labels. That is,  $\mathcal{D}_S = \{\mathcal{X}_S, \mathcal{Y}_S\} = \{(x_{S,1}, y_{S,1}), \dots, (x_{S,n_S}, y_{S,n_S})\}$ , where  $x_S \in \mathbb{R}^{d_S}$  and  $y_S \in \{1, \dots, C_S\}$ . The number of source domain data is  $n_S$ , and a source domain label  $y_S$  has  $C_S$  different classes. Similarly,  $\mathcal{D}_T$  is defined in the same way.

We use multitask learning, which is a subset of transfer learning, to enhance the performance of a hypothesis  $\mathfrak{H}$  on scarce data. In definition, the feature spaces of both domains are the same, while the distributions are different.  $\mathcal{X}_S = \mathcal{X}_T$  (This implies  $d_S = d_T$ ) but  $\mathbb{P}_S \neq \mathbb{P}_T$ . The label spaces  $\mathcal{Y}_S, \mathcal{Y}_T$  are also different due to  $C_S \neq C_T$  and the definition of categories. The problem aims to find a hypothesis  $\mathfrak{H}$  that can predict  $\mathcal{Y}_T$  by training on a large-scale source domain  $\mathcal{D}_S$  and a limited target domain  $\mathcal{D}_T$ . Note that  $n_S \gg n_T$ .



Table 3.1: Notation Table

Notation	Meaning
$\mathcal{D}_S = \{\mathcal{X}_S, \mathcal{Y}_S\}$	Source domain
$\mathcal{D}_T = \{\mathcal{X}_T, \mathcal{Y}_T\}$	Target domain
$\mathcal{X}$	Feature space
$\mathcal{Y}$	Label space
$\mathbb{P}$	Distribution of data
$d$	Dimension of feature space
$C$	the number of classes in label space
$\mathfrak{H} : \mathcal{X} \rightarrow \mathcal{Y}$	A hypothesis
$n$	the total number of data
$\mathcal{L}$	Loss function

## 3.2 Training Framework

The training framework we proposed is shown in Figure 3.1. There are two important steps in the framework. The first is the detection step. This step is for videos recorded by drones. In this step, an object detector SSD [22] locates human inside video frames. After the localization of human objects, an image containing the human will be cropped in the frames and background information will be removed. Then, the cropped images will be the input of the next step which is the MTL step. The MTL step is to share representations and transfer knowledge between massive dataset and limited dataset. This step can improve the model performance on the target task. In our problem, the target task is action recognition of drones, and we take large-scale datasets as source tasks. The transfer learning method of this step can be finetuning or multitask learning.

### 3.2.1 Detection Step

We illustrate how the detection step works. The Figure. 3.3 simply shows the input image and the output cropped image of the detection step. In our task, the original image size is (1280, 720). Our detector based on SSD [22] detects the position of the human inside the image. Before detecting images, the detector divides input images into multiple boxes. Then, the detector gives three outputs, bounding boxes, objects classification and confidence scores for each box. The detector outputs bounding boxes and their corresponding classification only if the confidence scores are greater than 0.5. Then, all objects

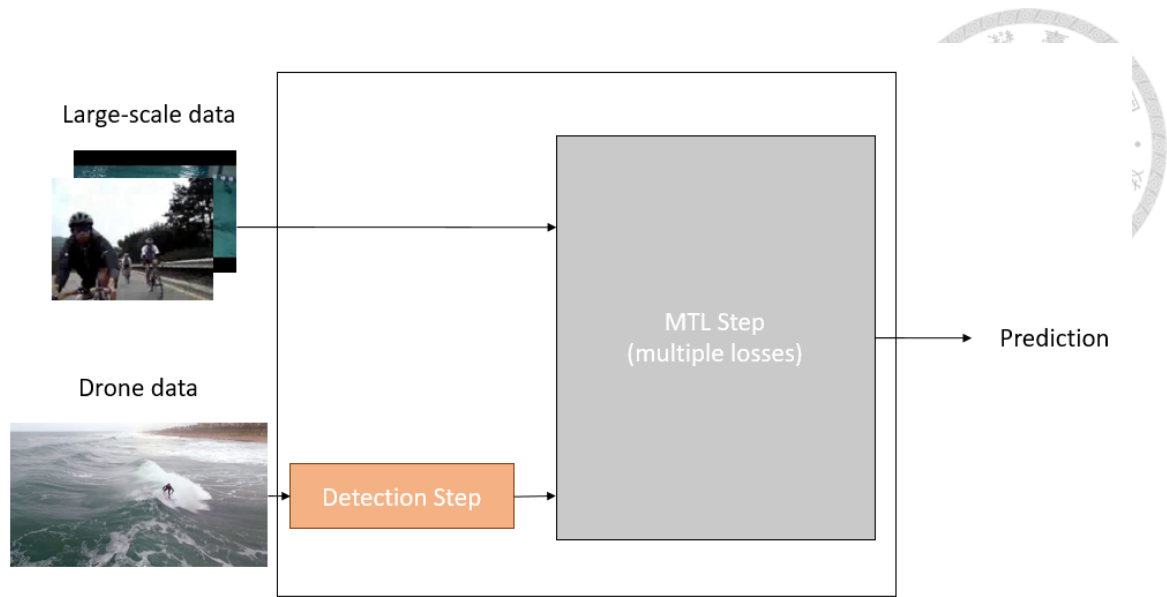


Figure 3.1: An illustration of our proposed method.

except human objects are removed. The center of bounding box that belongs to human objects will be remained. If there are multiple human positions, we randomly pick one of them. We take this point as a center of  $(340, 256)$  rectangle, and using this rectangle to crop the image of human. Finally, this cropped image will be the inputs of the MTL step. If the detector cannot detect human objects in some specific frames, these frames will be augmented with data augmentation techniques and then sent to the MTL step.



Figure 3.2: In standard action recognition problem, images are processed with data augmentation directly.

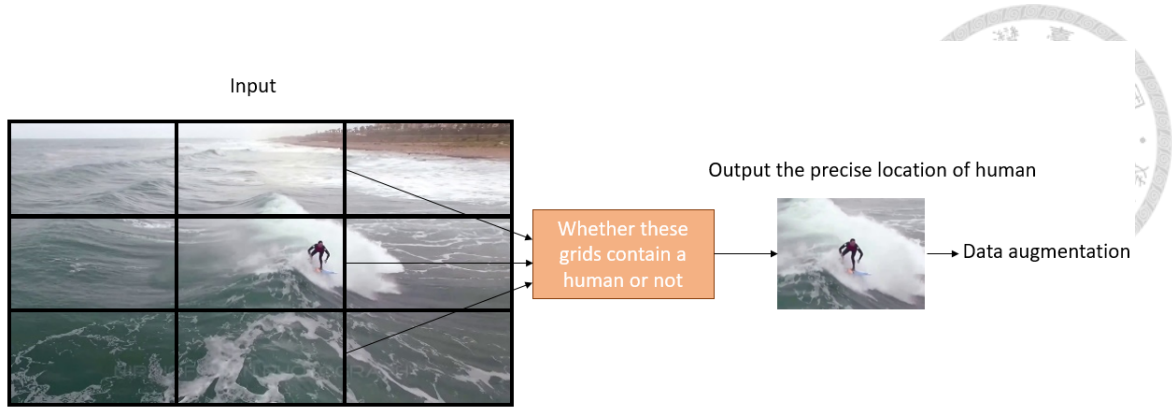


Figure 3.3: Detection step we proposed.

### 3.2.2 MTL Step

In the MTL step, the target task with limited data can learn knowledge from the source task with a large number of data. Some transfer learning techniques can be applied. For instance, finetuning is an efficient way to transfer knowledge. Here, we propose a multitask learning model for the MTL step. Multitask learning is an effective framework to augment limited size training data. The main idea of multitask learning is to share weights and representations between different domains and tasks. In our framework, we use different datasets to share the same CNN model. The input of the CNN model is mixed video frames, and there are multiple softmax outputs for each dataset. The loss scores will be computed according to the domain of input video frame. Figure 3.4 shows our framework.

The total loss function of this framework is defined as follows.

$$\mathcal{L} = \sum_{i=1}^n \sum_{M=\{S,T\}} \delta_M w_M l_M(y_{M,i}, \mathfrak{H}_M(x_{M,i})) \quad (3.1)$$

In equation 3.1, the input  $\{(x_{M,1}, y_{M,1}), \dots, (x_{M,n}, y_{M,n})\}$  is sampled and randomly permuted from both domains  $\mathcal{D}_S$  and  $\mathcal{D}_T$ .  $M$  stands for data domains.  $\delta_M$  is an indicator function to indicate the domain of the input video frame.  $w_M$  is the weight of each domain.  $l_M$  represents loss function for each domain, and we take categorical cross entropy as loss scores. We define  $l_M$  by equation 3.2.

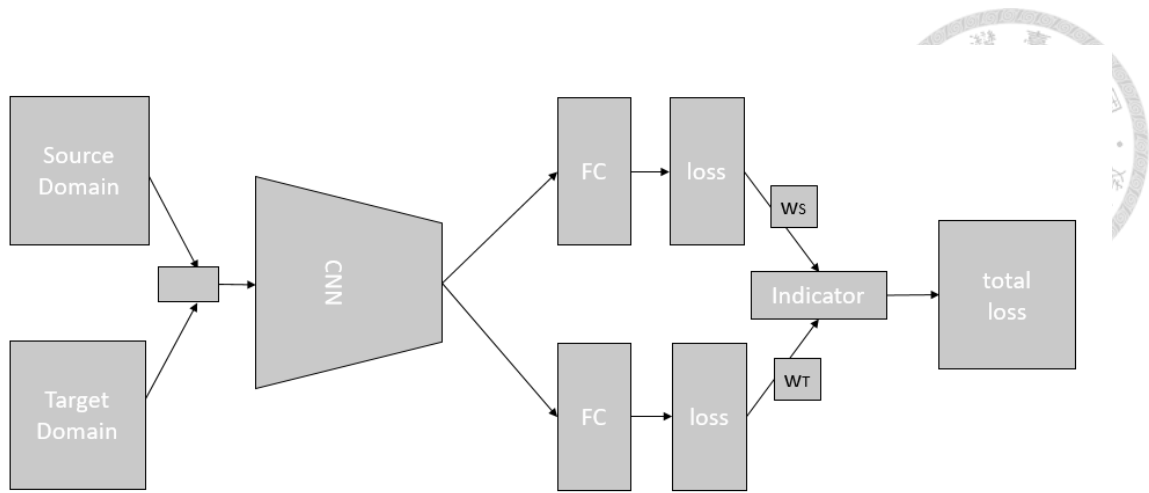


Figure 3.4: Multitask learning framework.

$$l_M(y_{M,i}, \mathfrak{H}(x_{M,i})) = - \sum_{j=1}^{C_M} y_{M,i,j} \log(\mathfrak{H}_M(x_{M,i})) \quad (3.2)$$

where  $j$  is the dimension of  $y_{M,i}$  in label space  $\mathcal{Y}_M$ .



## Chapter 4

# Experiment

In this chapter, we perform the experiments and show our results. First of all, we introduce all datasets used in experiments. And we describe all details and settings of experiments. Next, we present experimental results and some discussions. Finally, we use visualization methods to show the effectiveness of our training framework.

### 4.1 Datasets

There are two kind of datasets we used in experiments. The first is large scale human action datasets including UCF101 [28] and HMDB51 [17]. These datasets are widely used in action recognition research. The second is Drone dataset collected by ourselves. Details of these datasets are introduced below.

#### 4.1.1 Large Scale Human Action Datasets

We choose two challenging datasets HMDB51 [17] and UCF101 [28] to evaluate our model. HMDB51 contains 51 action categories including facial expressions and body movements. This dataset has total 6869 videos. According to the evaluation rules, the dataset has 3 different splits for training models and testing models. Each split contains 3.5k training videos and 1.5k testing videos. On the other hand, UCF101 contains total 101 action categories such as facial expressions, body movements, sports and playing instruments. There are total 13320 videos in the dataset. This dataset also has 3 different



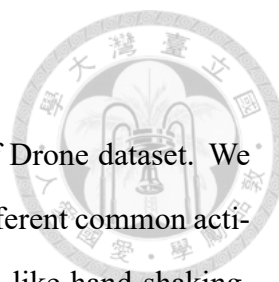
splits for training models and testing models. Each split has 9.5k training videos and 3.7k testing videos.



Figure 4.1: HMDB51 dataset.



Figure 4.2: UCF101 dataset.



### 4.1.2 Drone Dataset

In this subsection, we describe the collection and organization of Drone dataset. We use Drone dataset collected by ourselves. This dataset contains 14 different common actions. There are three main groups including human to human action like hand-shaking, human to object like eating, and independent actions like jumping. All videos include human actions from aerial view. The sources of these videos are from drone Bebop2 and Dronestagram, which is a website for users to share their drone videos.

In the next step, we cut these videos into short clips. Some videos from Dronestagram have shot transitions. We remove all these transitions. And if a video is longer than 30 second, we divide it into clips. About 15 second for each clip. Then, we manually label these clips according to the defined actions. We use an annotation tool to help labeling process. Finally, we split these data into 3 groups including train data and test data.

There total 138 video clips. These clips are divided into a training split and a testing split. The training split has 26 clips and the testing split has 112 clips. Detailed statistics is showed in Figure. 4.3.

All action categories are classified into 3 groups that are human to human actions, human to object actions and human independent actions. First, human to human actions represent interactive activities between humans. Second, human to object actions mean that humans have interactions with objects. Third, human independent actions stand for activities only a human involve in. Details are listed in Table 4.1.

Table 4.1: Action categories of drone dataset

Human to human	Human to object	Human independent action
base ball	bike	eat
basket ball	climbing stairs	jumping
shaking hands	rafting	run
tennis	riding motors	walking
	skiing	
	surfing	

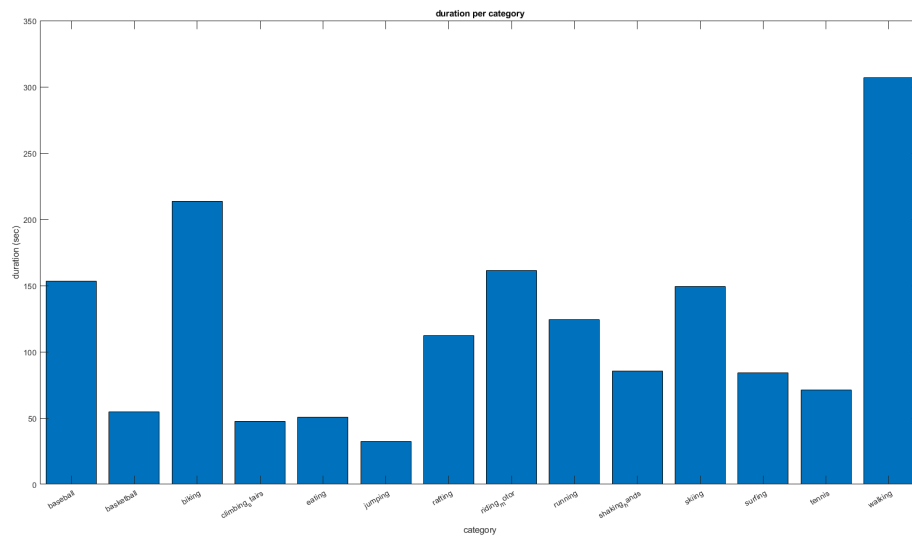
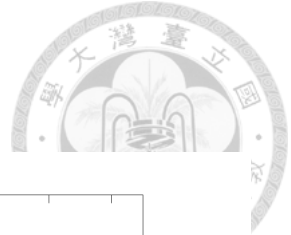


Figure 4.3: Duration of each action in Drone dataset.



Figure 4.4: Examples recorded by Bebop2.

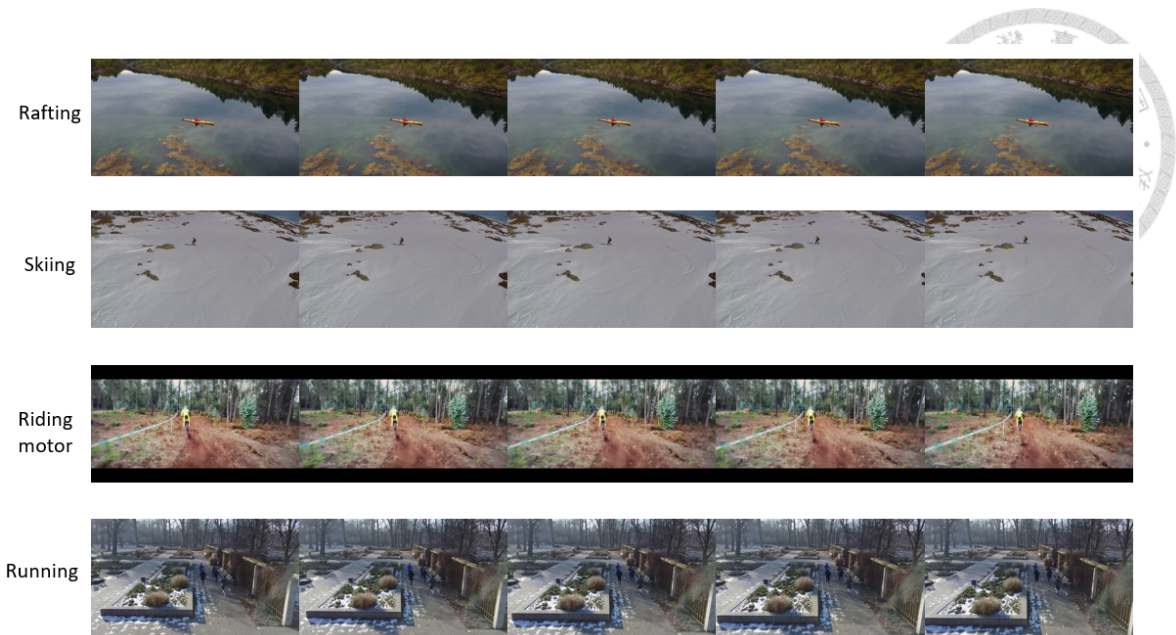


Figure 4.5: Examples from Dronestagram.

## 4.2 Experiment Settings

### 4.2.1 Environment

We deploy a deep learning toolkit Keras to build our model, choosing tensorflow backend with GPU accelerating. We perform experiments on a machine containing an Intel i7-7820 CPU, 64 GB RAM and 2 GTX 1080 Ti GPUs. All video data are preprocessed by OpenCV libraries in Python, and the operating system is Ubuntu 16.04.

### 4.2.2 Training Details

In the detection step, we use a VGG [27] model to detect human locations. This model is trained on two popular object detection datasets PASCAL VOC2007 and COCO. Although there are many categories in these datasets, we choose human objects for our task. In the MTL step, we use an initial weights pretrained with ImageNet data. The input shape of CNN is  $(224, 224, 3)$ , which represents height, width and color channels of image frames. We randomly sample 1 frames for each training videos in a epoch. To increase data size and avoid overfitting, we use some data augmentation techniques such as random horizontal flipping, random cropping and scale jittering [40]. In addition, we normalize

input values by decreasing 128 and dividing 128, rescaling the color range to  $[-1, 1]$ . In the training procedure for evaluating CNN models, the optimizer applies SGD with initial learning rate 0.001 and Netserov momentum 0.9. After 50 epochs, the learning rate decreases to 0.0001. After 120 epochs, the learning rate reduces to 0.00001. Total 150 epochs for training. For finetuning and MTL, we use fewer training epochs and smaller learning rate. The total epoch is 50, and the learning rate is fixed to 0.0001.

In order to demonstrate the effectiveness of the detection step and the MTL step, we perform experiments under different configurations. In the first experiment, we show that MTL is effective. We compare 2 training configurations. The first is training the model with scarce data directly, and the other is training on large-scale data and scarce data under MTL framework. We also compare MTL with other transfer learning method like finetuning. In the second experiment, we aim to show the combination of the detection step and the MTL step can improve model performance for drone dataset. We compare 3 configurations. First, we directly train the model with drone data. Next, we train the model with drone data, and the model contains the detection step. Third, we combine the detection step and the MTL step to train the model with drone data and large-scale data together. The illustrations of these configurations are shown in Figure 4.6, Figure 4.7, Figure 4.8 and Figure 4.9.

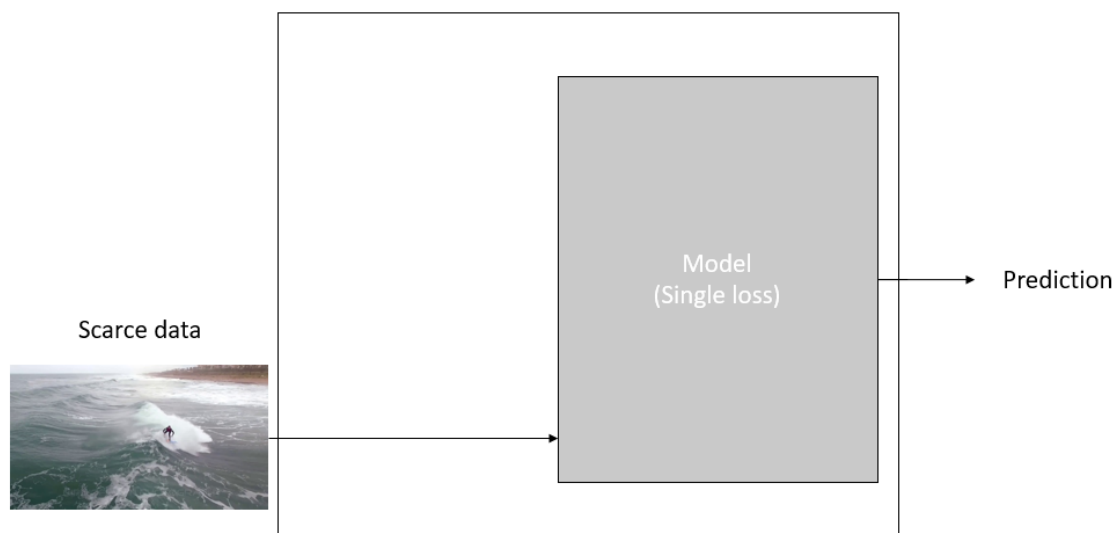


Figure 4.6: Directly training on scarce data.

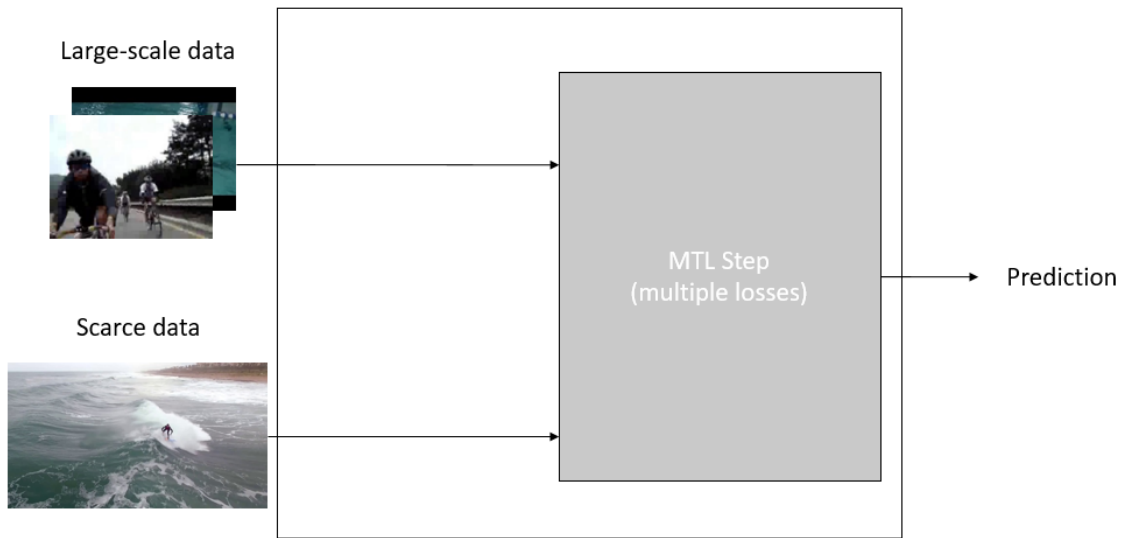


Figure 4.7: Training on both datasets in the same time in order to show the effectiveness of MTL.

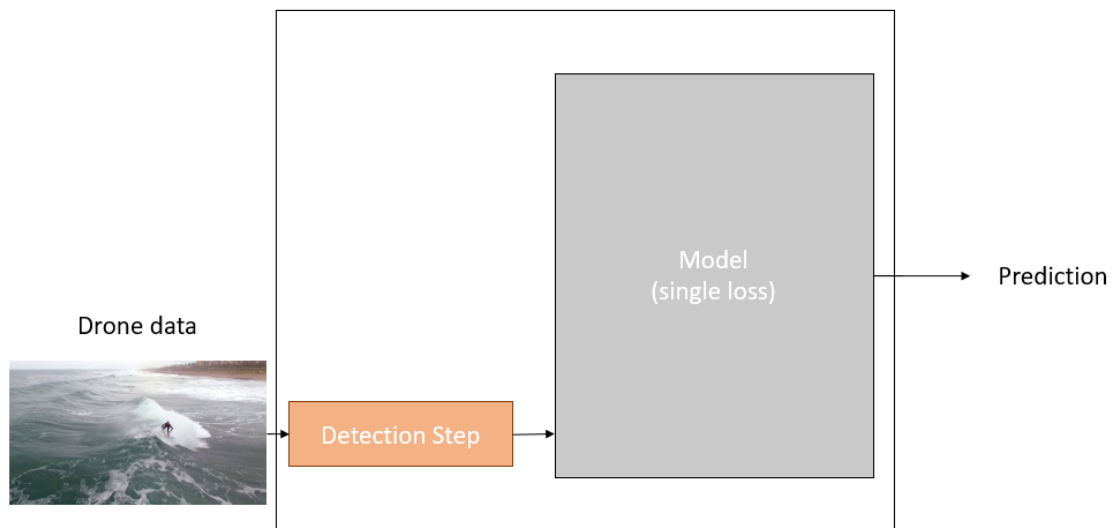


Figure 4.8: Training on drone dataset with the detection step in order to show the effectiveness of this step.

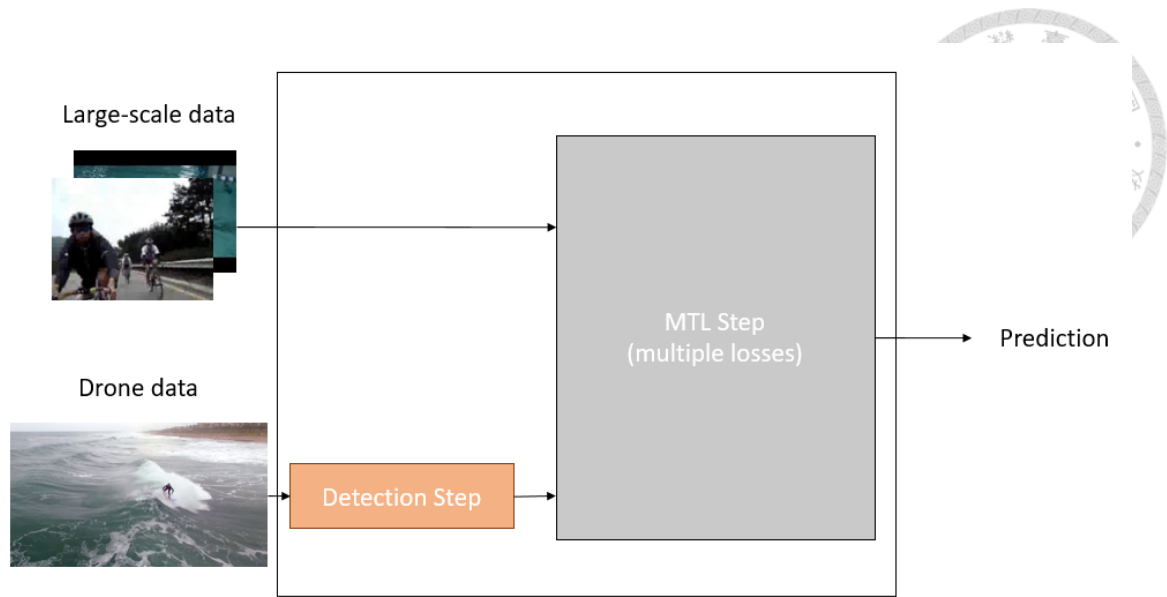


Figure 4.9: Complete form.

### 4.2.3 Testing Details

We follow the testing procedure of previous work [26]. For each testing video, we uniformly sample 25 frames and the corresponding features including RGB frames and stacking optical flows. Then we crop these frames and their flipping at 4 corners and the center with a  $(224, 224)$  square. The model predicts an action for each square; thus, a video has 250 predicted action probabilities. We average these probabilities across all frames; then, the maximum probability among the actions represents the action of the video. Therefore, we can compare predicted results and ground truth to evaluate an accuracy of the model. The overall accuracy is video wise. That is, the number of correctly classified videos over the number of all testing videos.

In every experiment, we use the same condition for fairness. We only test the performance of CNN model. This condition is illustrated in Figure 4.10.

## 4.3 Results

First, we want to choose a baseline CNN for our task. Our candidate models contain DenseNet [11], Inception V3 [34] and ResNet [10]. We use UCF101 split 1 and HMDB51 split 1 to evaluate these models, and we compare the performance on different features

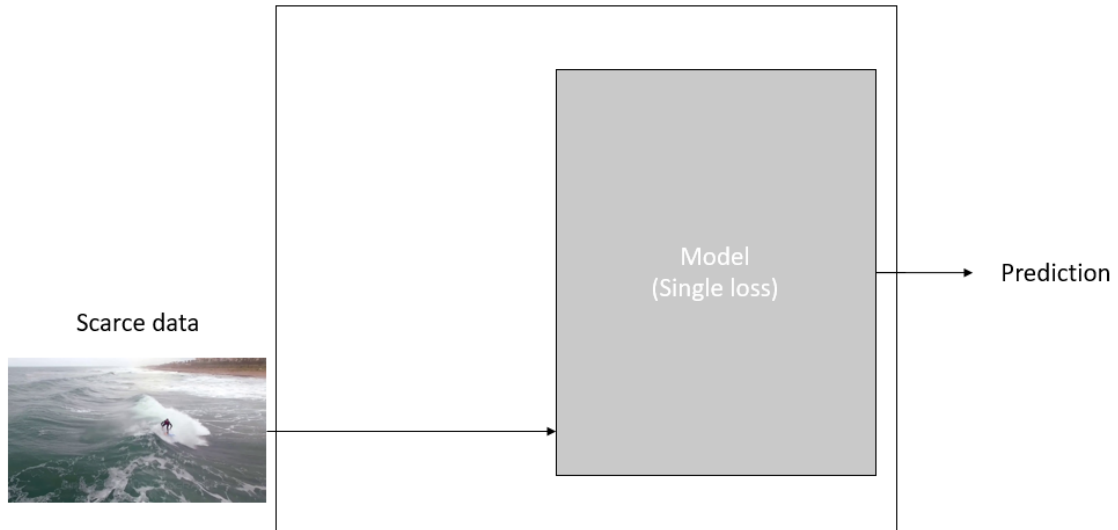


Figure 4.10: For fairness, we use the same setting to test our model.

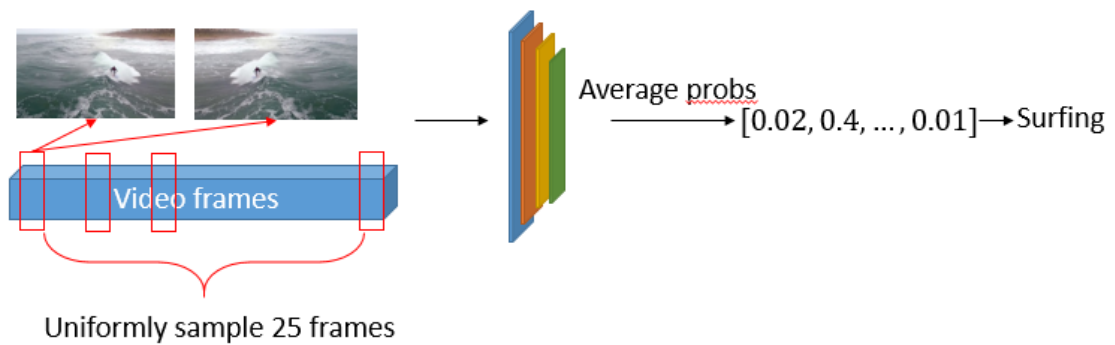


Figure 4.11: An illustration to show how accuracy is evaluated.



such as rgb, optical flow and Two-stream model. Results are shown in Table 4.2 and Table 4.3.

Table 4.2: Evaluation of CNN models on UCF101

CNN model	RGB(%)	Optical flow(%)	Two stream(%)
DenseNet-121	79.62	75.68	86.10
DenseNet-169	82.66	76.16	87.95
DenseNet-201	83.32	76.45	88.08
Inception-V3	82.87	75.52	87.87
ResNet-50	78.83	78.59	86.34

Table 4.3: Evaluation of CNN models on HMDB51

CNN model	RGB(%)	Optical flow(%)	Two stream(%)
DenseNet-121	40.20	37.32	44.44
DenseNet-169	41.31	41.83	46.27
DenseNet-201	41.44	41.70	46.54
Inception-V3	43.14	42.29	49.61
ResNet-50	40.13	44.05	47.51

As can be seen, Inception V3 has relatively good performance on human action recognition datasets. The performance of Two-stream is 87.87% for UCF101 and 49.61% for HMDB51. Furthermore, the total training time of Inception V3 is about 12 hours while the other networks take about 24 hours. Therefore, we choose Inception V3 for our task due to the training efficiency and accuracy.

In the next evaluation, we demonstrate that multitask learning framework is effective. We take UCF101 split 1 as source data, and HMDB51 split 1 as target data. Under MTL framework, recognition performance on HMDB51 can be improved. The improvement is better than finetuning and training directly on UCF101. We use Inception V3, and we set loss weights  $w_S = 0.375$  and  $w_T = 1.0$  according to the relative portion of both datasets. The comparison result is shown in Table 4.4.

In addition, we demonstrate that MTL is effective for Two-stream networks [26]. A two-stream network contains 2 major parts including a spatial network and a temporal network. A spatial network takes RGB frames as input, while a temporal network takes optical flows as input. The outputs of both networks are fused to predict the action. We



Table 4.4: Evaluation of MTL on HMDB51 dataset

CNN model	HMDB51(%)
Inception-V3	43.14
finetuning	46.21
MTL	49.54

use MTL to improve the accuracy of a temporal network and Two-stream networks. Both networks are based on Inception V3 and UCF101 is used to increase the performance of HMDB51. The result is shown in Table 4.5.

Table 4.5: Evaluation of Two-stream on HMDB51 dataset under MTL framework

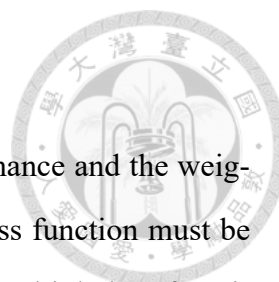
CNN model	HMDB51(%)	MTL(%)
Spatial Network	43.14	49.54
Temporal Network	42.29	44.71
Two-stream	49.61	56.14

We observe that a two-stream model can be improved under MTL framework. Before we used MTL, the accuracy of HMDB51 is 49.61%. With the benefits of MTL, the overall accuracy increase to 56.14%.

We evaluate our model on Drone dataset. There are four settings for this experiment. First, we directly train Inception V3 on Drone dataset. Next, we show that detecting actions can improve recognition performance on Inception V3. Third, we combine MTL and Inception V3 to show the effectiveness of MTL and compare the result with the previous setting. Finally, we combine detection and MTL to increase the generalization of our model. The result is shown in Table 4.6.

Table 4.6: Evaluation of the detection step and the MTL step on Drone dataset

CNN model	Drone dataset(%)
Inception-V3	36.61
Detection+Inception-V3	41.96
Inception-V3+MTL	42.86
All	44.64



## 4.4 Discussion

In this section, we discuss the relation between the MTL performance and the weights  $w$  of each loss function. In the MTL step, the weight of each loss function must be set before training. We determine how the weight portions between multiple loss functions can affect recognition performance. To verify this, we use UCF101 data to improve HMDB51 task in the MTL step. We set  $W_T = 1.0$  and change the parameter  $W_S$  from 0.125 to 0.875. The HMDB51 performance is shown in Figure 4.12.

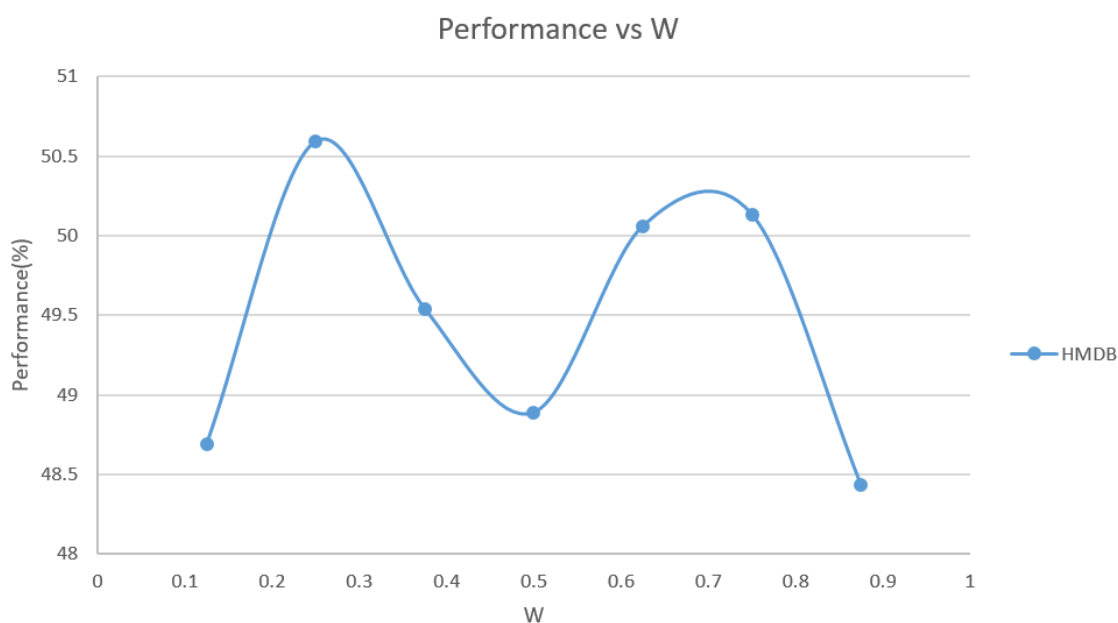


Figure 4.12: HMDB51 performance versus weights of loss functions.

Before we investigate this factor, we assume that the weights should be used to balance the data size from different sources. For example, the training data size of UCF101 is about 9.5k and HMDB51 is about 3.5k. We set weights of each loss function according to the reciprocal of their data size. That is,  $\frac{1}{9.5k} : \frac{1}{3.5k} \simeq 0.37 : 1.0$ . So we set  $W_S = 0.375$  and  $W_T = 1.0$ . Although in Figure 4.12, the peak value 50.59% appears at  $W_S = 0.25$ . This value is close to the value 49.54% at  $W_S = 0.375$ . In sum, the weight of each loss function can be set depending on the reciprocal of data sizes from different sources.

Another issue is that the detector could fail to detect human in some cases. For example, in Figure 4.13, the angle of the camera is at the top of the human. In this situation, the

detector is unlikely to detect human because the shapes of human are tiny circles instead of a human body shapes. A possible way to improve is to use more similar cases in the training.



Figure 4.13: Failed cases for detectors.

In addition, optical flows of drone videos are extremely difficult for training a temporal network. The purpose of temporal networks is to detect human motion in videos; however, optical flows of drone videos mainly detect the moving background instead of human motion because the drone camera itself is dynamic. Figure 4.14 simply illustrates this problem. There is a human running in the center but optical flows mainly contain useless background information. Moving camera detects moving directions of the background from the side of camera, and information of human objects becomes weak. Thus, it is challenging to train a temporal network with this issue.



Figure 4.14: An example of optical flows from drone videos.



## Chapter 5

### Conclusion


In conclusion, we present a new learning framework that can improve the recognition accuracy on action recognition problem for drones. This learning framework is two-stage including the detection step and the MTL step. The detection step helps a CNN model focus on human objects, and the MTL step enhances the accuracy on limited drone data. Furthermore, we propose a new human action dataset of drones. The dataset has 14 different action categories. This dataset is challenging due to small human objects and data scarcity.

In future work, we plan to apply two new human action datasets recently proposed in our problem. The first dataset called Kinetics [15] is proposed by Google Deepmind. This dataset contains 400 action categories. The second dataset is SLAC [43] presented by Facebook Research and MIT jointly. We prefer to use the pretrained models since training on these datasets is time-consuming. We will study these datasets for our task after pretrained models are released. In addition, we want to extend our drone dataset from 14 actions to 20 actions including some anomalous human actions. The final goal is to detect anomalous behaviors in real time with drone technologies. However, some action samples such as shooting and stealing are difficult to collect. In order to solve this issue, we will use virtual world data instead of real world data to perform the detection task. Then, we transfer the model to drones in real world to detect human actions.

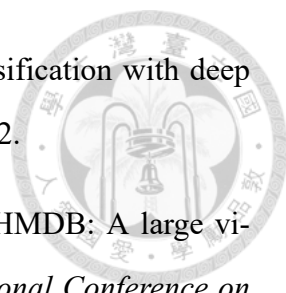


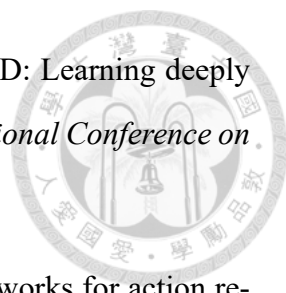
## Bibliography


- [1] M. Barekatin, M. Martí, H.-F. Shih, S. Murray, K. Nakayama, Y. Matsuo, and H. Prendinger. Okutama-action: An aerial view video dataset for concurrent human action detection. In *1st Joint BMTT-PETS Workshop on Tracking and Surveillance, CVPR*, pages 1–8, 2017.
- [2] H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould. Dynamic image networks for action recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3034–3042, June 2016.
- [3] N. F. Davar, T. de Campos, D. Windridge, J. Kittler, and W. Christmas. Domain adaptation in the context of sport video action recognition. In *Domain Adaptation Workshop, in conjunction with NIPS*, 2011.
- [4] C. R. de Souza, A. Gaidon, Y. Cabon, and A. M. López. Procedural generation of videos to train deep action recognition networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2594–2604, July 2017.
- [5] C. R. de Souza, A. Gaidon, E. Vig, and A. M. López. Sympathy for the details: Dense trajectories and hybrid classification architectures for action recognition. In *ECCV*, pages 697–716. Springer, 2016.
- [6] A. Diba, V. Sharma, and L. V. Gool. Deep temporal linear encoding networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1541–1550, July 2017.


- 
- [7] C. Feichtenhofer, A. Pinz, and R. Wildes. Spatiotemporal residual networks for video action recognition. In *NIPS*, pages 3468–3476, 2016.
- [8] 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.
- [9] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1933–1941, June 2016.
- [10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016.
- [11] G. Huang, Z. Liu, L. Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, July 2017.
- [12] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.
- [13] S. Jégou, M. Drozdal, D. Vazquez, A. Romero, and Y. Bengio. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1175–1183, July 2017.
- [14] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1725–1732, June 2014.
- [15] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.



- 
- [16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.
- [17] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: A large video database for human motion recognition. In *2011 International Conference on Computer Vision*, pages 2556–2563, Nov 2011.
- [18] 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.
- [19] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, Dec 1989.
- [20] G. Lev, G. Sadeh, B. Klein, and L. Wolf. RNN fisher vectors for action recognition and image annotation. In *ECCV*, pages 833–850. Springer, 2016.
- [21] J. Li, Y. Wong, Q. Zhao, and M. S. Kankanhalli. Attention transfer from web images for video recognition. In *ACM MM*, pages 1–9. ACM, 2017.
- [22] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.
- [23] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4694–4702, June 2015.
- [24] Z. Qiu, T. Yao, and T. Mei. Deep quantization: Encoding convolutional activations with deep generative model. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4085–4094, July 2017.

- 
- [25] Z. Shen, Z. Liu, J. Li, Y. G. Jiang, Y. Chen, and X. Xue. DSOD: Learning deeply supervised object detectors from scratch. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1937–1945, Oct 2017.
- [26] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, pages 568–576, 2014.
- [27] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.
- [28] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *Technical Report CRCV-TR-12-01, UCF Center for Research in Computer Vision*, 2012.
- [29] N. Srivastava, E. Mansimov, and R. Salakhudinov. Unsupervised learning of video representations using LSTMs. In *ICML*, pages 843–852, 2015.
- [30] Y.-C. Su, T.-H. Chiu, C.-Y. Yeh, H.-F. Huang, and W. H. Hsu. Transfer learning for video recognition with scarce training data for deep convolutional neural network. *arXiv preprint arXiv:1409.4127*, 2014.
- [31] L. Sun, K. Jia, D. Y. Yeung, and B. E. Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4597–4605, Dec 2015.
- [32] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12, 2017.
- [33] C. Szegedy, W. Liu, Y.-Q. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, June 2015.

- 
- [34] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, June 2016.
- [35] 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.
- [36] G. Varol, I. Laptev, and C. Schmid. Long-term temporal convolutions for action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2018.
- [37] H. Wang, A. Kläser, C. Schmid, and C. L. Liu. Action recognition by dense trajectories. In *CVPR 2011*, pages 3169–3176, June 2011.
- [38] H. Wang and C. Schmid. Action recognition with improved trajectories. In *2013 IEEE International Conference on Computer Vision*, pages 3551–3558, Dec 2013.
- [39] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4305–4314, June 2015.
- [40] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. V. Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, pages 20–36. Springer, 2016.
- [41] S. Zagoruyko and N. Komodakis. Wide residual networks. In *BMVC*, 2016.
- [42] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, pages 818–833. Springer, 2014.
- [43] H. Zhao, Z. Yan, H. Wang, L. Torresani, and A. Torralba. SLAC: A sparsely labeled dataset for action classification and localization. *arXiv preprint arXiv:1712.09374*, 2017.

- 
- [44] F. Zhu and L. Shao. Enhancing action recognition by cross-domain dictionary learning. In *BMVC*. Citeseer, 2013.
- [45] W. Zhu, J. Hu, G. Sun, X. Cao, and Y. Qiao. A key volume mining deep framework for action recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1991–1999, June 2016.