國立臺灣大學電機資訊學院資訊工程學系

碩士論文

Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

以神經網路整合雙串流軌跡資料用於計算任意人類動作次數

Aggregating Two-Stream Trajectory using Neural Network for Counting Arbitrary Human Action Repetition

林之宇

Chih-Yu, Lin

指導教授：徐宏民博士

Advisor: Winston Hsu, Ph.D.

中華民國 107 年 7 月

July, 2018

doi:10.6342/NTU201803126

致謝

在完成這篇論文的過程中，首先我要感謝我的指導老師徐宏民教授對我的指引，老師總是鼓勵我們先動手做再從實作中學習，此觀點對於我尋找論文題目以及著手解決問題都非常地有幫助。除此之外，老師也在我犯錯時給予我十分多的包容，告訴我要把握學生時期多嘗試、多犯错，對此我真的非常感激。

另外我也很感謝我的實驗室同期同學們，在研究撞牆期你們總給予我鼓勵與實用的建議，而在談論研究之餘我們奇蹟般地都如此喜歡旅行以及美食，在實驗室討論各自的旅行經驗、行程安排計畫以及美味餐廳的時光真的很開心！

最後感謝我的家人以及男朋友的支持陪伴與幫助，謝謝你們！

doi:10.6342/NTU201803126

中文摘要

雖然深度學習已經在電腦視覺方面取得相當地勝利，辨識人類的動作重複次數仍是十分具挑戰性的問題，在病人進行復健以及人們進行重量訓練等時刻皆需要計算動作重複的次數。解決此問題的困難點在於重複動作時動作間的細微差異、人們在畫面中重複動作時相機的拍攝視角移動以及對應不同重複動作需要不同的處理。為了解決這個問題，我們收集了一個新的資料集以及建立了一個嶄新的網路架構，人類重複動作計算網路，用以計算任意的人類重複動作。我們的網路使用人們隨著時間進行重複動作之軌跡的頻域資訊來計算人類重複動作的次數，實驗結果指出我們的網路在計算人類任意動作的重複次數方面有優於以往的表現。另外由於人類進行重複動作的影片取得較為困難，我們也製作了一個資料集，使用產生波形的方式來模擬人類重複動作時的軌跡，並使用此資料集預訓練我們的網路以獲得更精確的計算能力。人類重複動作計算網路並不僅僅只能計算人類重複動作的次數，在實驗中，以物體進行重複動作時的軌跡輸入網路也可以獲得物體重複動作的次數。

關鍵字：重複動作次數計算

doi:10.6342/NTU201803126

Abstract

Although deep neural network has achieved great success in computer vision recently, the problem of determining repetitions of arbitrary periodic human actions is still challenging. The difficulties lay in varying frame length of repetitions, temporal localization of human beings and different features corresponding to different motions. Moreover, the demand of human action repetition counting is rising in medical rehabilitation and sport events, etc. To address this problem, we construct a human action dataset and propose a brand new framework, Human Action Repetition Counter (HARC), which could work on *arbitrary* human actions with a *single* architecture. Our HARC learns to count repetitions of human action in the time-frequency domain determined after few pilot studies. The experiments show that HARC outperforms previous counting methods on benchmarks. Additionally, we design novel learning strategies by generating effective synthetic data to pretrain our network, which can further boost the performance and reach more accurate results. We also demonstrate that our HARC is also capable of counting the periodic object motions. Our dataset, *YT_Human_Segments* dataset, will be publicly available which will benefit future researches.

Keywords: periodic motion, repetition counter

doi:10.6342/NTU201803126

Contents

doi:10.6342/NTU201803126

doi:10.6342/NTU201803126

List of Figures

doi:10.6342/NTU201803126

doi:10.6342/NTU201803126

doi:10.6342/NTU201803126

List of Tables

doi:10.6342/NTU201803126

Chapter 1

Introduction

1.1 Motivation

Over the past few years, there have been significant advances in the field of video understanding. However, there are very few works discussing how the machine learns to count, especially in human action repetition. Human action repetition counting is crucial for various applications such as physical therapy and workout. For instance, with human action repetition counting, the therapists and the patients do not have to count the repetitions by themselves during the rehabilitation. Please see the demo videos in the supplemental. Unlike the depth camera such as Kinect receiving extra depth information, our goal is to leverage a human action repetition counter which takes a simple RGB video instead of a depth video as input, so we can easily utilize only a camera beside us to count without memorizing how many repetitions we have done so far.

Counting the number can be difficult to model with video-related methods, and prior works in counting still remains few. Recently, [1] tried to solve this problem by using convolutional neural network. This work samples 20 frames as the input of the convolutional neural network and predicts cycle length ranging from 1 to 10. An outer system concept is utilized to integrate the cycle length predicted in every moment and judge the final repetition count. Although this method has demonstrated significant improvements over counting problem and has reached the state-of-the-art performance, there are still some limitation of their strategy. Due to the upper bound of the sampling rate, the maximum

doi:10.6342/NTU201803126

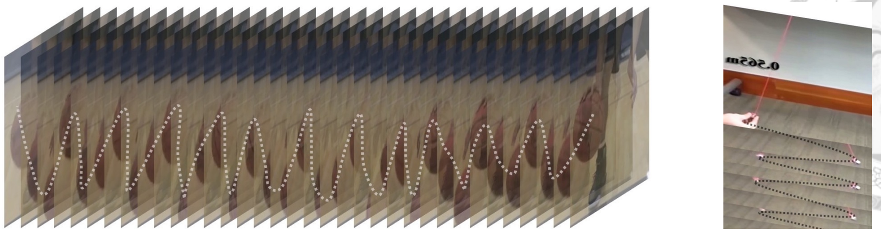


Figure 1.1: **Intuition.** We stack up a series of frames to observe the trajectory of a certain joint. The vertical trajectory of the center of a bounced ball through a period of time is shown in **(a)**. In **(b)**, the horizontal trajectory of the center of a swinging pendulum is observed from a low angle view. From (a) and (b), the repetition count can be clearly inferred. More details are shown in Sec. 3.1.

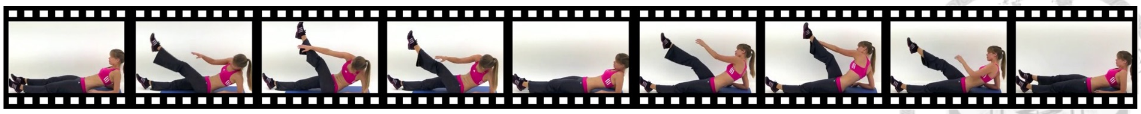
frame length of repetition is limited by this constraint. Besides, on human action repetition scenario such as workout, their method did not perform as well as other real-world tasks, see Table. 4.1. The poor performance is attributed to the complexity and low speed of the action, because the limitation constrains that one action needs to be done within the maximum frame length. However, workout movements are often slow and complicated.

Our intuition comes from the regularity of the action. Consider the examples in Fig. (1.1). Given a video with several action repetitions, we can easily count the number of repetition by observing both the horizontal trajectory and the vertical trajectory of the target object. Take Fig. 1.1(a) for example, when a ball is bounced, we can easily get the repetition count only by observing how many times the ball moves up and down. More specifically, we judge the repetition number from the vertical trajectory of the ball. Likewise, in Fig. 1.1(b), we can effortlessly tell how many times a pendulum swings by analyzing the back and forth frequency of the movement. In other words, we count the repetitions by observing the horizontal trajectory of the pendulum. Inspired by the two examples mentioned above, we address the human action repetition counting problem by observing both the horizontal trajectory and the vertical trajectory of human's body joints.

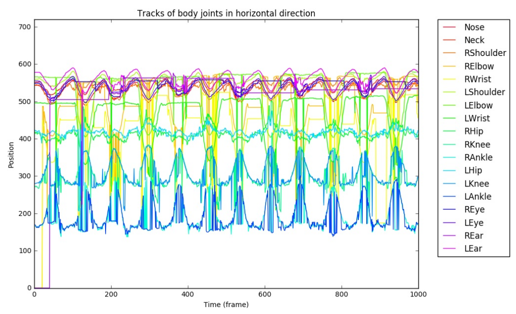
1.2 Related Work

Periodic motion detection has been a well-known issue for a long time, and many people have achieved great success on it. Previous works mostly address this problem by

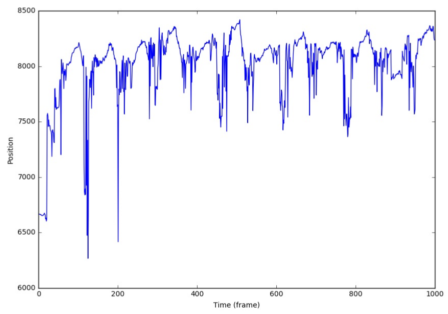
doi:10.6342/NTU201803126



(a)



(b)



(c)

Figure 1.2: **Feature extraction.** **(a)** When a woman is performing single-leg toe-touch crunch, she will bring her left leg toward her chest and touch her big toe with her right hand in the first repetition and then she will alternate sides in the following repetitions. **(b)** The trajectories of her body joints in the horizontal direction. According to (b), some body joints are at wave crests while others are at wave troughs when abrupt changes happen. **(c)** By summing up these trajectories, we can enhance the wave crests at the moment when abrupt changes happen. More discussions are shown in Sec. 3.1.

using spatial correlation and applying time-frequency analysis [2–6]. Inspired by them, we question whether we can further extract repetition counts by employing time-frequency analysis. Previous works mentioned above assume that there will exactly be a discernible peak in trajectories when abrupt changes happen, but referring to Fig. 2 in [7], clear peaks are unnecessarily displayed in spatial correlation features of periodic motions. Time-frequency feature extracted from these unclear peaks will be more complicated and more difficult to analyze. Another issue is that traditional frequency analysis needs parameter tuning such as time scale selection in different scenarios, which is inconvenient for human action repetition counting since there exists a huge variance in the frame length of each human periodic motion.

The state-of-the-art work [1] first employs deep learning method to solve this problem by using convolutional neural network to predict cycle length. This work samples 20 frames as the input of the convolutional neural network and predicts the cycle length ranging from 1 to 10. A concept of outer system is utilized to integrate the cycle length predicted in every moment and judge the final repetition count. The network also produces

doi:10.6342/NTU201803126

entropy of the predictions, which is employed to determine the start time and the end time of periodic motions. The upper bound of the sampling rate is set to 8, which means only the repetition whose frame length is within 80 can be detected. This work provides two datasets, *YT* dataset and *YTSegments* dataset, both of which contain 100 videos. Results are evaluated on these two datasets and other benchmarks by using mean absolute errors (MAE). Inspired by [1] and the existing approaches [2–6] using time-frequency analysis, we leverage deep neural networks to accomplish the time-frequency analysis rather than addressing them manually. Additionally, we utilize a feature extraction method in order to generate trajectories with more discernible peaks.

1.3 Contribution

Toward this end, we present a brand new framework, Learning to Count Neural Networks (L2CNN), which is a single detector for arbitrary actions. By leveraging human pose detection, our framework learns to count repetitions of human action in the time-domain frequency. Our work is applicable on various human action repetitions without being limited by those constraints mentioned above. In addition, we will release a new challenging dataset called *YT_Human_Segments*, which consists of 100 videos related to workout, and we manually label the the number of repetitions frame by frame. To the best of our knowledge, this is the first work of modeling a single detector network to count the repetitions, specifically on arbitrary human actions. Our contributions are as follows.

1. We propose a single detector network with deep neural network for counting arbitrary action repetitions, which leverages the time-domain information.

2. We will release a brand new dataset called *YT_Human_Segments* for counting action repetitions. For the richness of the dataset, we create the synthetic data to enhance the variations.

3. Our proposed method has greater generalization than other methods, and is beyond the limitation of the frame length.

doi:10.6342/NTU201803126

4. We evaluate our method on the *YT_Human_Segments* dataset, and our method achieves significant improvements over other state-of-the-art methods.
5. We demonstrate that our approach can be easily used to count the periodic object motions, such as ball bouncing, without fine-tuning the network on additional data.

doi:10.6342/NTU201803126

Chapter 2

Dataset

2.1 YT_Human_Segments Dataset

Due to the fact that our repetition counter is not able to detect the start of a motion, we have to segment the collected YouTube videos to make them display the motion part only. We collect 100 periodic human motion videos from YouTube with more than 20 categories to create our own *YT_Human_Segments* Dataset (Fig. 2.1). Using our feature extraction method, features extracted at the end of $rep.n$ might be similar to the features extracted at the beginning of $rep.n + 1$. As a result, we only take the consecutive 10 frames at the beginning at $rep.n$ and label them as $n - 1$ (Fig. 3.2), e.g. for a push-up video clips which contains 10 repetitions, we would label the 10 consecutive frames at the beginning of rep. 2 as 1, the 10 consecutive frames at the beginning of rep. 3 as 2, etc. We also use the same method to label the *YTSegments* dataset which is used to evaluate our results.

2.2 Synthetic Data

Originally, we copy and concatenate an arbitrary number of a video clip which contains only one action repetition because it is hard to collect periodic motion video clips with various repetitions. Due to the fact that both segmenting video clips and extracting data from the segmented video clips are time-consuming, we propose novel strategies for synthesizing the training data. Firstly, we choose different frequencies to produce varied

doi:10.6342/NTU201803126

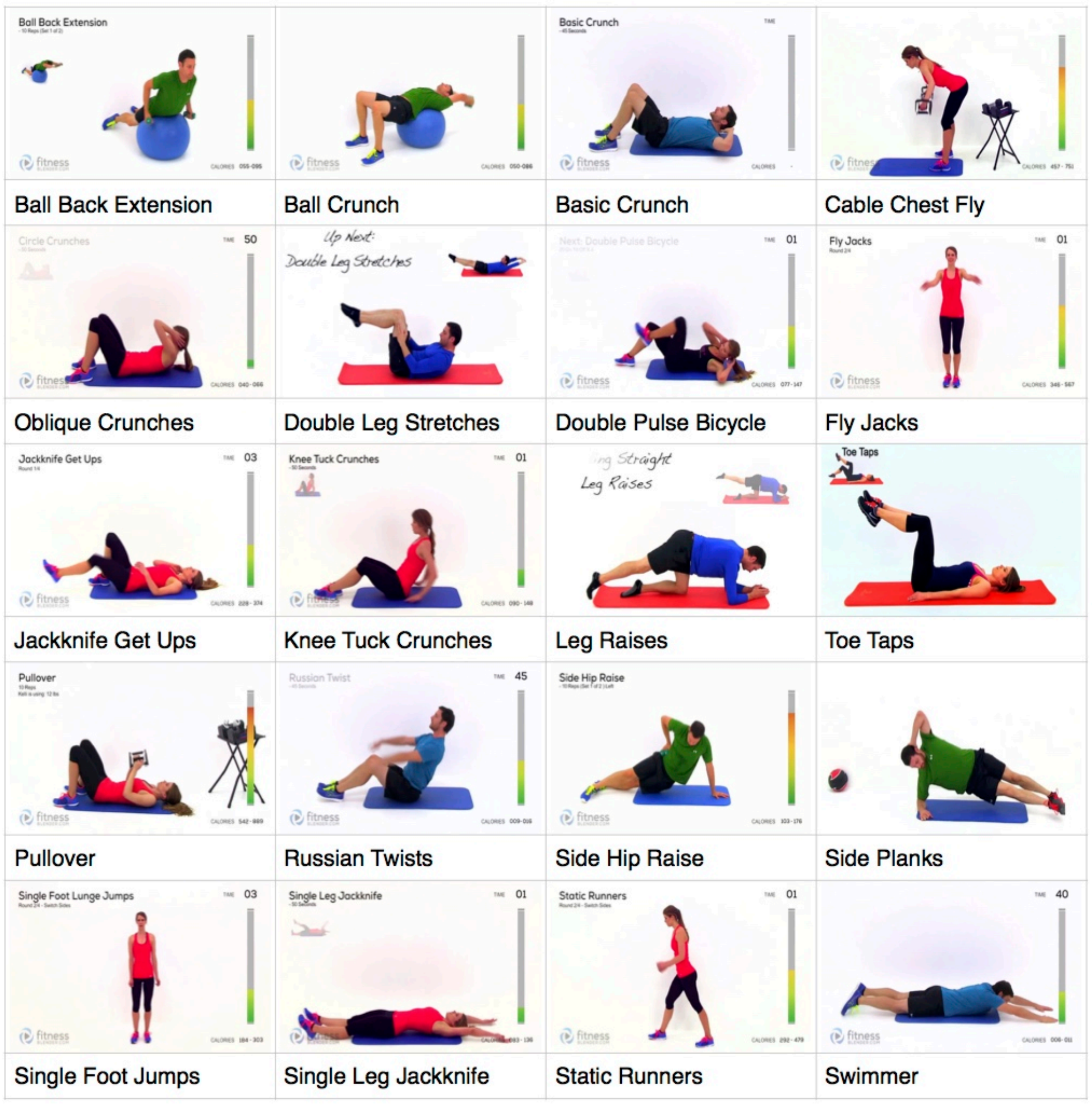


Figure 2.1: The *YT_Human_Segments* dataset contains 100 human action video clips with more than 20 categories. Part of categories are shown as above. More details are shown in Sec. 2.1.

sin signals and apply $Hann$ Window on the produced signals to increase diversity. Then, we simulate signals of different repetitions by randomly sampling part of the generated signals and repeating them within a range from 1 to 10 times. To increase more variations on our synthesizing data, we also add different waveforms with the same repetitions to make a brand new waveform. To simulate the vertical and the horizontal trajectories, a single data will contain two different curves with the same repetition count. We generate 180,000 data in total, and we also extend, normalize and obtain DFT features from the x and y trajectory curve in data respectively by using method mentioned in 3.1.

doi:10.6342/NTU201803126

Chapter 3

Human Action Repetition Counter

The Human Action Repetition Counter consists of a core system, Learning to Count Neural Networks (Sec. 3.2), which is based on a feed-forward network that produces a count number of the target video clip, followed by a outer system (Sec. 3.4) which is used to decide whether a repetition is finished or not. In the core system, the early network layers are based on a two-stream architecture used for compressing data, one for data on x-axis while the other one on y-axis (see Sec. 3.1). The entire core system is pretrained by using synthetic data and finetuned on our *YT_Human_Segments* dataset (Sec. 3.3). We then concatenate the outputs from the mentioned two-stream network and feed those outputs to some FC layers. In the end, a simple mechanism is implemented in the outer system to smooth the outputs from each time step into final prediction.

3.1 Feature Extraction

Applying CMU pose estimation detection [8] on the input video shot, we get locations of 18 body joints of the target human object in each frame. Firstly, we link the X coordinate of the same joints together and the Y coordinate of the same joints together and then generate 36 curves. In the second step, we extend each curve to a fixed length by using linear interpolation method and normalize these curves to 0-1 range. Finally, we obtain the features in time domain after combining these curves by using mean pooling method. And then we extract Discrete Fourier Transform (DFT) features from the above

doi:10.6342/NTU201803126

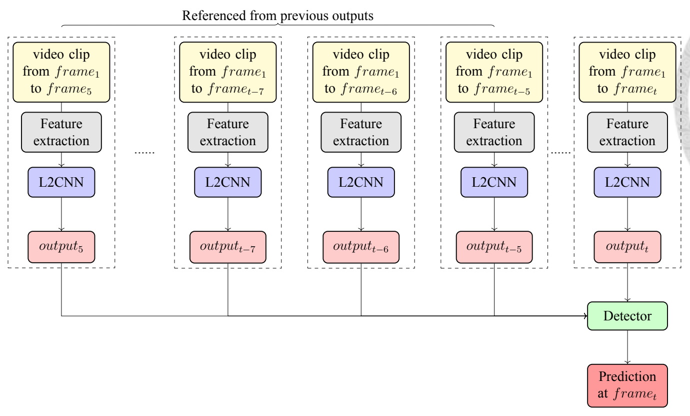

Referenced from previous outputs
video clip from $frame_1$ to $frame_5$
video clip from $frame_1$ to $frame_{t-7}$
video clip from $frame_1$ to $frame_{t-6}$
video clip from $frame_1$ to $frame_{t-5}$
video clip from $frame_1$ to $frame_t$
Feature extraction
L2CNN
$output_5$
$output_{t-7}$
$output_{t-6}$
$output_{t-5}$
$output_t$
......
Detector
Prediction at $frame_t$

Figure 3.1: **Framework.** We define the output of video clip from $frame_1$ to $frame_t$ as $output_t$. Due to the limit of our feature extraction method, we start producing outputs after 4 frames in the input video clip, which means $output_5$ will be the first output. When the length of input video is T frames, the detector will judge the accurate repetition count from $output_t$ and outputs in the past, e.g. $output_{t-1}$, $output_{t-2}$, $output_{t-3}$, etc. And we denote the detector's output at $frame_t$ as $prediction_t$. More discussions are shown in Sec. 3.2 and Sec. 3.4.

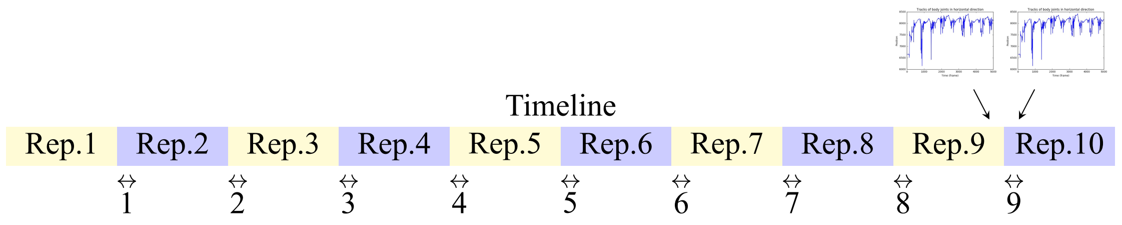


Figure 3.2: **Label Rule.** Due to the fact that features would look similar when they are at the very end of $rep.n$ and the very beginning of $rep.n + 1$, we only take the features of the first consecutive 10 frames at the beginning of each repetition as our training data. More details are shown in Sec. 3.1.

time domain features. Furthermore, frequency domain features are generated by applying 0-1 normalization on the extracted DFT features. We extend our curves to the same length and extract DFT features because repetition count would be proportional to the frequency in this scenario.

DFT Features. Given a 1-D array feature of which the length is 5000, we extract its amplitude by using Discrete Fourier Transform. Also sample frequencies are generated with $time\_step = 1/5000$. After sorting the amplitude according to its sample frequencies, we will get a symmetric amplitude. We only take the right part with positive sample frequencies of which the length is 2499 to be our DFT features.

doi:10.6342/NTU201803126

3.2 Learning to Count Neural Networks (L2CNN)

We train our L2CNN to analyze the input DFT features and get an action count prediction of a video clip. The entire architecture is shown in Fig. 3.3. We get used to calculating human action repetitions by observing the horizontal trajectory and the vertical trajectory of the target human at the same time, e.g. while a person is doing squat, we can induce squat count by observing how many times a person moves up and down, which is denoted as the vertical trajectory; while a person is doing cable chest fly, we can induce the repetition count by observing how many times a person moves his/her arms back and forth, which is denoted as the horizontal trajectory. Accordingly, we design our L2CNN by using a two-stream structure in the beginning to reduce dimensions of the input DFT features. One stream is for DFT from x trajectory while the other one is for DFT from y trajectory. Each stream consists of three FC layers with 5000, 1200, 400 nodes respectively and the Rectified Linear Unit (ReLU) is applied afterwards. These FC layers are all followed by batch normalization and dropout. After these two streams, we concatenate the outputs and get an aggregated feature of which the dimension is 800. Additionally, we feed this feature into two-subnetworks (SN1 and SN2) with different structures. SN1 consists of six FC layers with 700, 560, 450, 300, 150, 1 nodes, respectively. The top five FC layers are with $relu$ activation function and are followed by batch normalization and dropout; SN2 consists of two FC layers with 50 and 1 nodes, respectively. The first layer is with $relu$ activation function and are followed by batch normalization and dropout. In the end, we concatenate the outputs of SN1 and SN2 and feed them into the last FC layer which has one node and output the final prediction.

3.3 Training

Due to the constraint of our label method, only 7736 training data reach our criteria; however, the amount is too small to train a model. Consequently, we first use the synthesizing data to pretrain our L2CNN and then finetune it on our training data. Since we define our problem as a regression problem, we use mean-squared-error as our loss func-

doi:10.6342/NTU201803126

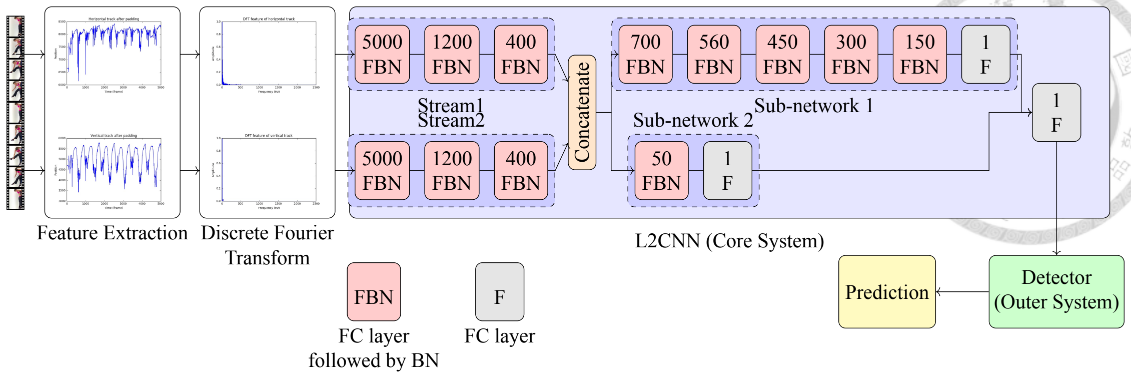


Figure 3.3: **L2CNN.** Our human repetition counter consists of two parts, one is a core system, L2CNN, and the other one is a outer system which integrates outputs of L2CNN in a period of time, predicting an accurate repetition count. The L2CNN mainly consists of FC layers, some of which are followed by a batch normalization layer while the others of which are not. Two streams are used to reduce the dimension of inputs in the beginning, and then the outputs of these two streams are concatenated, fed into two sub-networks. After that we will concatenate the outputs of sub-networks and vote for a final output of the entire L2CNN by using a FC layer with one node. More details are shown in Sec. 3.2.

tion during model training and pretraining. We set the dropout ratio of all dropout layers to 0.5 during pretraining and finetuning stage. When we pretrain our L2CNN with synthetic data, we employ keras adam optimizer with learning rate $= 0.0001, beta_1 = 0.9, beta_2 = 0.999, epsilon = 10^{-8}$ and $decay = 0.0$. After 300 epochs, the error of the pretrained model has converged to 0.0867. We finetune our L2CNN on our *YT_Human_Segments* dataset by using the same keras adam optimizer with learning rate $= 0.00008, beta_1 = 0.9, beta_2 = 0.999, epsilon = 10^{-8}$ and $decay = 0.0$. After 350 epochs, the model has reached the convergence and its error is 0.2723.

3.4 Detector: Smoothing Outputs

Since we only use the first 10 frames at the beginning of a repetition to train our model, our core system would be more sensitive to the start of a repetition. After referring to the detector design in [1], we also has a counter R which stores the current repetition count and holds the estimated repetition count from the beginning of the target video clip. A simple mechanism is employed to integrate the outputs over time. When a number M consecutively appears for N times and M is larger than the value of the current repetition count R, then we update R's value as M, e.g. the current repetition count stored in R is

doi:10.6342/NTU201803126

5, and N is set to 8. Now a sequence of outputs 3, 4, 1, 2, 6, 6, 6, 6, 6, 6, 6, 6 is generated by our core system and it is apparently that 6 consecutively appears for 8 times. Thus we update R's value to 6.

doi:10.6342/NTU201803126

Chapter 4

Experiment

4.1 Evaluation Method

We evaluate our human action repetition counter on both *YTSegments* dataset [1] and our *YT_Human_Segments* testing dataset. Due to the limit of human action video, only 61 out of 100 video clips of the *YTSegmens* dataset are used to evaluate our proposed method. The frame length of a single action repetition in these 61 video clips is around 80 while that in our *YT_Human_Segments* testing dataset is ranging from 90 to 400. Accordingly, *YTSegments* dataset can be used to evaluate the performance of our repetition counter on repetitions with shorter frame length and the other one can be used to evaluate repetitions with longer frame length, respectively. Each video clip contains n repetitions, and we evaluate our counter by detecting whether any output of our repetition counter is corresponding to the ground truth within a margin N of the end of a repetition (see Fig. 4.1), e.g. if $repetition_8$ ends at $frame_{1000}$, then we will see if any output in $prediction_{1000-N}$, $prediction_{1000-N+1}$, $prediction_{1000-N+2}$, ...$prediciton_{1000+N}$ corresponds to 8. If it does, then we will say the repetition counter hits in $repetition_8$, and vice versa. Mean absolute error (MAE) is also used for evaluation. Noticeably, [1] define absolute error as $\frac{|GT-Prediction|}{GT}$ while we define absolute error in our paper as $|GT - Prediction|$ because we think penalty on wrong prediction with smaller ground truth is unfair. Both kinds of MAE are utilized for evaluation. When we evaluate our model using MAE [1], there is no extra annotated labels for the *YTSegments* dataset. Besides, we pick 4 video clips from

doi:10.6342/NTU201803126

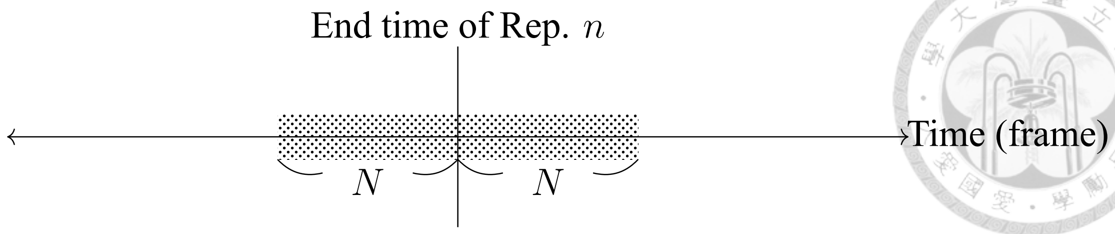


Figure 4.1: **Evaluation rule.** When it comes to evaluation, we take a look at the N frames before the end of rep. n and the N frames after the end of rep. n, seeing if any prediction of these $2N$ frames is exactly n. Even if only one of these $2N$ predictions is n, we still say that we predict the correct answer. More details are shown in Sec. 4.1

our training dataset with repetition average frame length shorter than 80 but contain much more complex repetitions such as switch-foot kicks, lunge jumps to test if [1] can handle them, and the MAE [1] turns out to be 34%. We can infer from the result that [1] can not deal with complicated periodic motions perfectly.

4.2 Comparison with Benchmarks

We evaluate our HARC on two benchmarks and compare the result with the baseline method Live Rep. [1] and Segment Rep. [1]. Live Rep. [1] can detect when periodic motions begin and stop while Segment Rep. [1] can not. The two benchmarks mentioned above are the *YTSegments* dataset and the *YT_Human_Segments* dataset. Since the first dataset containing repetitions of which the frame length is shorter, we adjust the parameter N mentioned in Sec. 3.4 to 7. Whereas we adjust the parameter to 20 when we evaluate performance on the second dataset. Due to the fact that our training data having labels mostly ranging from 1 to 11, we also evaluation on video clips containing less than 12 repetitions. Significantly, our HARC performs better than the state-of-the-art method. We not only have smaller mean absolute error but also obtain double hit rate compared to baseline's performance on both datasets, see Table. 4.1.

doi:10.6342/NTU201803126

Table 4.1: **Comparison between our method and the baseline method.** The hit rate and mean absolute error (MAE) of *YTSegments* benchmark and our *YT_Human_Segments* benchmark for our method and Live Repetition Count method [1]. More details are shown in Sec. 4.2.

		Full		Rep.# < 12	
		YTSegments	YT_Human_Segments	YTSegments	YT_Human_Segments
Hit Rate	Live Rep. [1]	31%	5.3%	29%	5.6%
	HARC	**68%**	**65%**	**77%**	**70%**
MAE	Live Rep. [1]	1.74	4.50	1.74	4.49
	HARC	**1.32**	**1.30**	**0.67**	**1.19**
MAE [1]	Segment Rep. [1]	6.5	213.8		
	HARC	**29.4**	**17.0**		

Table 4.2: **Comparison between the finetuned model and the unfinetuned model.** The hit rate and mean absolute error (MAE) of *YTSegments* benchmark and our *YT_Human_Segments* benchmark for our finetuned model and unfinetuned model. More discussions are shown in Sec. 4.3 and Sec. 4.1.

		Full		Rep.# < 12	
		YTSegments	YT_Human_Segments	*YTSegments*	YT_Human_Segments
Hit Rate	Finetuned	68%	65%	77%	70%
	No Finetuned	30%	53%	29%	56%
MAE	Finetuned	1.32	1.30	0.67	1.19
	No Finetuned	1.85	1.22	1.01	1.09

4.3 Pretrained Model

Since our training dataset is not enough to cover all kinds of curve patterns, synthetic data is employed to pretrain our L2CNN. Fine-tuning on the pretrained model using our *YT_Human_Segments* dataset achieves better performance than simply training our L2CNN with the *YT_Human_Segments* dataset. To better understand the role of fine-tuning on the pretrained model, we evaluate on two models to compare the difference, one is with fine-tuning procedure while the other one is not. As shown in Table 4.2, the removal of fine-tuning procedure produces a large drop in performance. The hit rate of *YTSegments* dataset even drop from about 70% to less than 30%.

4.4 Robustness

In order to test the robustness of our HARC, we randomly drop n joints' locations of every input frame while testing our model on both the *YTSegments* dataset and our

doi:10.6342/NTU201803126

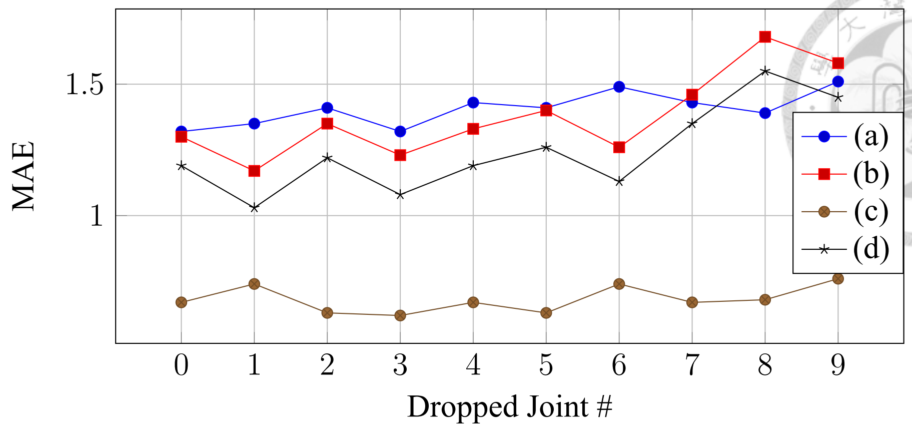


Figure 4.2: **Robustness of our method.** The mean absolute error of *YTSegments* benchmark and our *YT_Human_Segments* benchmark for our method with randomly dropping body joints. (a) to (d) are Full *YTSegments*, Full *YT_Human_Segments*, *YTSegments* with Rep.# $<$ 12 and *YT_Human_Segments* with Rep.# $<$ 12 recursively. More details are shown in Sec. 4.4.

YT_Human_Segments testing dataset, e.g. if we set n to 3, we may drop locations of neck, left eye and right wrist at $frame_t$, and those of left shoulder, nose and right knee at $frame_{t+1}$. We address these missing locations issue by using the previous location of that joint. If the location of a joint is lost at the very beginning of a video clip, then we will set the location of that joint to $(0, 0)$. In the example we mentioned above, we set the locations of neck, left eye and right wrist to where they are at $frame_{t-1}$, and set the locations of left shoulder, nose and right knee to where they are at $frame_t$. With the increase of dropped body joint number, the mean absolute error is still robust and lower than which of Live Repetition Count method (referring to Table. 4.1), see Fig. 4.2.

4.5 Action Repetition Counting on Objects

HARC could also be employed on object action repetition counting as long as we are able to track the location of the target object in video clips. We use object tracking tools to draw bounding box on our target object to obtain its locations of every frame in video clips. We utilize two object action video clips to test our repetition counter. The first video, Bounced Ball, shows that a ball is bounced for 14 times (repetitions), and the second one, Swinging Pendulum, shows a pendulum swings back and forth for 10 times (repetitions). Both of them have repetitions which have a shorter frame length within 10. Therefore, we

doi:10.6342/NTU201803126

adjust the parameter N mentioned in Sec. 3.4 to 3. We aim at counting how many times the ball has been bounced and how many times the pendulum has been swung. According to our experimental result, the mean absolute error of bounced ball repetition counting and swinging pendulum repetition counting is 0.72 and 0.66, respectively. Therefore we also achieve great success on object action repetition counting with the same L2CNN, which means additional training procedure is unnecessary in this scenario.

doi:10.6342/NTU201803126

Chapter 5

Conclusion

In this paper, we propose a new approach for human action repetition counting. To deal with time-frequency features extracted from complicate trajectories, we leverage deep neural network to analyze them. Our main contribution is that, we present a novel framework, HARC, which is the first framework that could use a single detector for arbitrary actions. In addition to this, we provide a new benchmark covers periodic motions containing repetitions with longer frame length. This dataset, *YT_Human_Segments* dataset, will be publicly available to benefit future researches. Furthermore, we demonstrate that our HARC could count other repeating actions, rather than human actions, without additional training procedure and labeled data. As for future work, we would like to enable our repetition counter to precisely count periodic motions with more than 11 repetitions as well as to detect the start point and the end point of a repetition.

doi:10.6342/NTU201803126

Reference

[1] Ofir Levy and Lior Wolf. Live repetition counting. In *International Conference on Computer Vision (ICCV)*, 2015.

[2] Ousman Azy and Narendra Ahuja. Segmentation of periodically moving objects. In *2008 19th International Conference on Pattern Recognition, ICPR 2008*, 2008.

[3] Ping-Sing Tsai, Mubarak Shah, Katharine Keiter, and Takis Kasparis. Cyclic motion detection for motion based recognition. 27:1591–1603, 12 1994.

[4] Alexia Briassouli and Narendra Ahuja. Extraction and analysis of multiple periodic motions in video sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(7):1244–1261, 7 2007.

[5] A. Thangali and S. Sclaroff. Periodic motion detection and estimation via space-time sampling. In *Application of Computer Vision, 2005. WACV/MOTIONS '05 Volume 1. Seventh IEEE Workshops on*, volume 2, pages 176–182, Jan 2005.

[6] R. Cutler and L. S. Davis. Robust real-time periodic motion detection, analysis, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8): 781–796, Aug 2000.

[7] Scott Satkin and Martial Hebert. Modeling the temporal extent of actions. In *European Conference on Computer Vision*, September 2010.

[8] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. *CoRR*, abs/1611.08050, 2016.

doi:10.6342/NTU201803126