Enhanced Spatial and Multi-Scale Temporal Feature Modeling for Human Identification at a Distance

Haijun Xiong, Xiaohu Huang, Bin Feng

{xionghj, huangxiaohu, fengbin}@hust.edu.cn

1 Introduction

Unlike other biometrics such as face, fingerprint and iris, gait is a unique biometric feature that can be recognized at a distance without the cooperation of subjects and intrusion to them. Therefore, it has broad applications in crime prevention, forensic identification and social security. However, gait recognition suffers from exterior factors such as the subject's walking speed, dressing and carrying condition, and the camera's viewpoint and frame rate.
In this competition, the training set contains 500 subjects with 10 video sequences for each one. The test set contains 514 subjects which are different from the training set. In the test set, the gallery includes one sequence of each subject, and the probe set consists of the rest sequences. In addition, there are many quite low-quality images, especially for those images with extremely large or small foreground, which are mostly indistinguishable, thus would degrade recognition performance.
In our approach, we basically follow the most general network GaitSet[1], meanwhile introduce Multi-branch Diverse Region Feature Generator (**MDFG**), and Global and Micro Motion Capturing Module (**GMCM**) for discriminative feature learning and global-local temporal feature learning respectively. To deal with the low-quality images, we propose a simple image-filtering strategy to filer the low-quality images by considering the ratio of the foreground.

2 Method

2.1 Overview

In this subsection, we introduce the framework of our project as shown in Fig.1, which includes three major steps.

- **Model pre-training.** Considering the number of the training samples are limited, we use the CASIA-B [2] data set for pre-training to strengthen the generalization ability of our network.

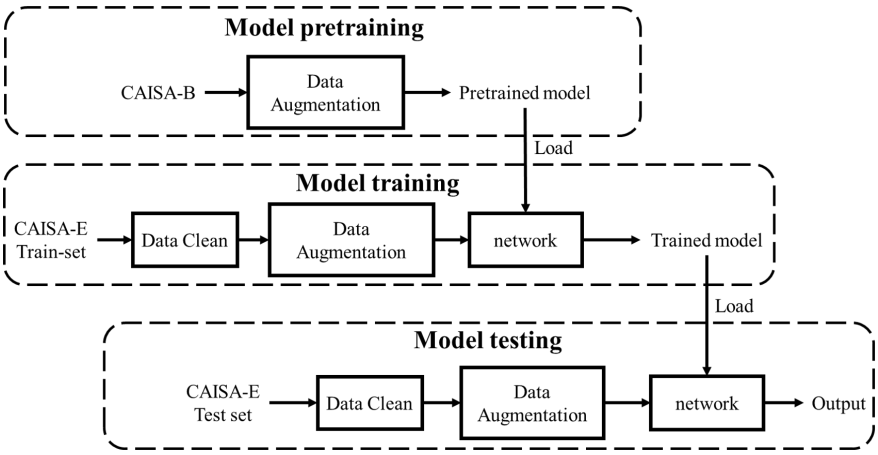


Fig. 1: The framework of our method

- **Model training.** Due to the existence of low-quality images, we clean the data set and select pictures with the middle ratio of foreground. At the same time, data augmentation is carried out during the training process to increase data diversity.
- **Model testing** Lastly, during testing, the test data set is also processed through data-augmentation.

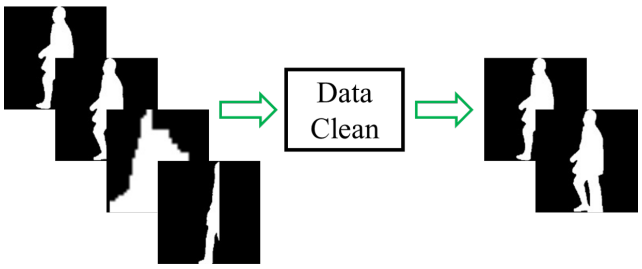


Fig. 2: DataClean

2.2 Proposed method

In this subsection, we elaborate the methods which we use in the competition, including preprocessing and details of network structure.

2.2.1 Preprocessing

Before training and testing, we firstly preprocess the dataset, which includes data cleaning and data augmentation.

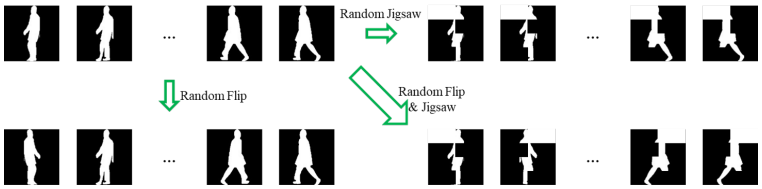


Fig. 3: DataAugmentation

- **Data cleaning** As shown in Fig.2, we observe that there are some low-quality frames in the input sequence. To deal with this, we use a simple classifier to select high quality images from an input sequence in both training and test data sets. Firstly, we count the ratio of foreground of each image in each sequence. Then, we sort the images by the ratio of foreground of each image in each sequence, and record the median ratio. Finally, we filter the images with the ratio of the foreground 15% larger or 15% smaller than the median ratio.
- **Data augmentation** We use the different data augmentation methods during training and testing. The train data augmentation methods include random horizontal flipping and random jigsaw, while the test data are only processed with random horizontal flipping. Fig.3 gives an illustration of data augmentation.

2.2.2 Network

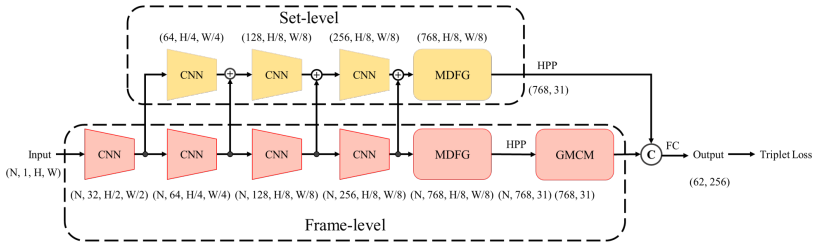


Fig. 4: Overview of our network

As shown in Fig.4, we use the GaitSet[1], MDFG and GMCM modules in our network.

GaitSet Observing the original GaitSet network, we found the CNN part only has three layers, the number of channels is 32, 64, 128. To get more detailed information, we add one CNN layer, which including two convolution modules, so the CNN part of the network has the four layers, and the number of channels became 32, 64, 128, 256. Generally, the deeper the network structure, the more information we can obtain and extract.

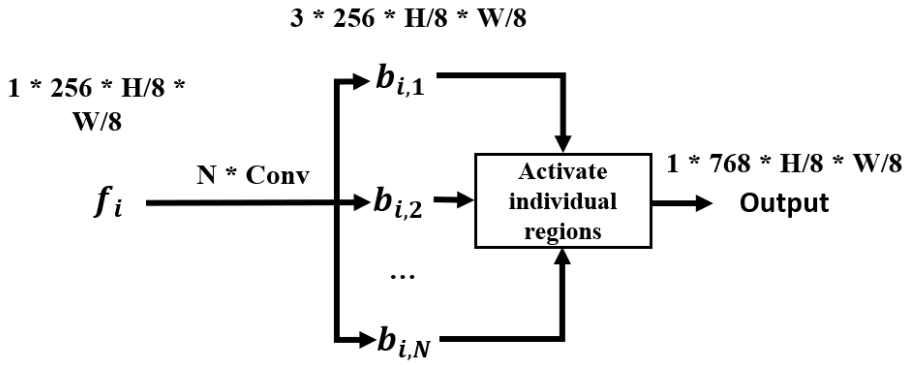


Fig. 5: Details of MDFG. We take the features of i-th frame as illustration. N independent 1×1 2D convolutions are applied on feature f_i for multi-branch feature generation. The we supervise branches to produce individual activated regions.

MDFG As mentioned in Sec.1, MDFG is proposed to generate visual clues in diverse regions for fine-grained feature learning. As shown in Fig.5, we apply N independent 1×1 2D convolutions on each frame $f_i \in \mathbb{R}^{1 \times C \times H \times W}$, then produce N branches output $b_{i,.}, b_{i,.} = \{b_{i,j} | j = 1, 2, \ldots, N\}$. We denote $B = (b_{i,j})_{S \times N}$ as the overall multi-branch output features. Firstly, we employ GAP and GMP on $b_{i,j}$ along channel dimension, then obtain channel-compressed feature $H_{i,j} \in \mathbb{R}^{1 \times H \times W}$. For each branch, the k-th maximum value of $H_{i,j}$ is denoted as $\sigma_{i,j}$, then we combine $H_{i,j}$ with $\sigma_{i,j}$ to obtain a focal mask $A_{i,j}$ by a Sigmoid form:

$$A_{i,j}(x, y) = \frac{1}{\exp(-H_{i,j}(x, y)) - \sigma_{i,j})} \tag{1}$$

where (x, y) denotes any spatial location in $H_{i,j}$ and $\sigma_{i,j}$ represents the threshold of Sigmoid function. Thus, locations with values lager than $\sigma_{i,j}$ are highlighted, and locations with values smaller than $\sigma_{i,j}$ are suppressed. To generate different activated regions in different branches, we apply OAP loss on $A_{i,j}$ for supervising. OAP loss aims to punish overlapped activated regions, here is the definition:

$$L_{oap}^{i} = \frac{1}{N} \sum_{x,y} (A_{i,1} \odot A_{i,2} \odot \cdots \odot A_{i,N}) \tag{2}$$

where L_{oap}^{i} represents OAP loss for i-th frame, $\odot$ denots element-wise multiplication, the overall L_{oap} is defined as:

$$L_{oap} = \frac{1}{S} \sum_{i=1}^{S} L_{oap}^{i} \tag{3}$$

where S denotes the number of the input frames.

GMCM As shown in Fig.6, the GMCM contains two parts: the Micro-motion Template Builder (MTB) and Global-motion Template Builder (GTB). The MTB is borrowed from GaitPart [3], which aims to map the frame-level part-informed feature vectors into the micro-motion feature vectors and the GTB is designed to map the frame-level global-informed feature vectors into the feature vectors.

MTB is a short-range temporal feature aggregation module in [3], the details are shown in Fig.7(a), where TempFunc is MaxPooling function.

GTB As shown in Fig.7(b), the GTB is to aggregate the inputs. Let $S_{in} = \{b_i | i = 1, 2, \cdots, t\}$ be a row of the PR-Matrix [3], which is a two-dimensional tensor with the sequence and channel dimension. The GTB is designed to map the sequence of global-level feature vectors S_{in} into the sequence of feature vectors S_{out}, formulated as

$$S_{out}^{GTB} = \mathbf{GTB}(S_{in}). \tag{4}$$

Then the ouput can be formulate as

$$S_m^i = S_{in}^i \cdot \text{Sigmoid}(\text{MLP}(S_{in}^i)) \tag{5}$$

$$S_{out}^{GTB} = \frac{1}{t} \sum_{i=1}^{t} S_m^i \tag{6}$$

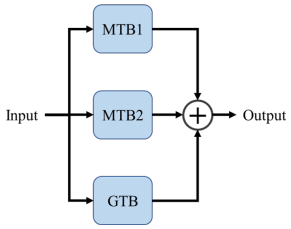
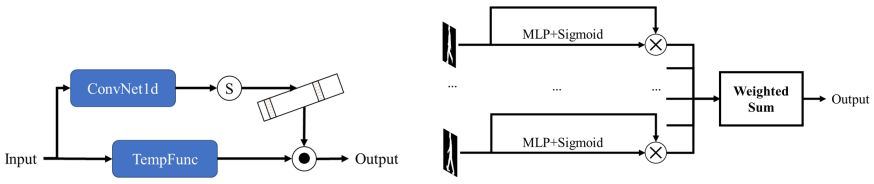


Fig. 6: Details of GMCM



(a) MTB (b) GTB

Fig. 7: Details of MTB and GTB

3 Datasets and Training Details

module	layer	kernel size	stride
module1	conv1-1	5×5	1
	conv1-2	3×3	1
	pooling1	2×2	2
module2	conv2-1	3×3	1
	conv2-2	3×3	1
	pooling2	2×2	2
module3	conv3-1	3×3	1
	conv3-2	3×3	1
	pooling3	2×2	2
module4	conv4-1	3×3	1
	conv4-2	3×3	1

Table 1: The convolution network settings

Module	MTB1|MTB2			
Layer	Conv1d_1	Conv1d_2	Avgpool1d	Maxpool1d
In_C	$C\|C$	$C/s\|C/s$	$\times$	$\times$
Out_C	$C/s\|C/s$	$C\|C$	$\times$	$\times$
Kernel	3|3	1|3	3|5	3|5
Pad	1|1	0|1	1|2	1|2

Table 2: The exact structure of MTB1 and MTB2. In_C, Out_ C, Kernel and Pad represent the input channels, output channels, kernel size and padding of the 1-D convolution layer, respectively. In particular, C and s represent the channels of input feature map and the squeeze ratio, respectively. Note that the values around '|' represent the setting of MTB1 and MTB2, respectively.

CASIA-B CASIA-B[2] contains 124 subjects, and each subject owns 110 sequences with 11 different camera views. Under each camera view, each subject contains three walking conditions, i.e., normal (NM) (6 sequences), walking with bag (BG) (2 sequences) and walking with coat (CL) (2 sequences). For the pre-training, all of them are used as train set.

Training Details The image of each frame is normalized to the size 128×88. The network parameters are shown in Table 1 and Table 2. The batch size parameters P and K are set to 16 and 4 in the network. During the training stage, the length of input gait sequences of the CASIA-B and competition datasets are all set to 30, respectively. During the test stage, if the sequence length is less than 300, the whole gait sequences are put into the proposed model to extract gait features, otherwise the input sequence length is set to 300. All experiments take Adam as the optimizer. For the CASIA-B dataset, the iteration number is set to 100K, and the learning rate is 1e-4. For the competition dataset, the

iteration number is set to 160K, and the learning rate is first set to 1e-4 and set to 1e-5 after 100K.

4 Result

The final accuracy on the competition dataset is 71.9%.

References

[1] Hanqing Chao, Yiwei He, Junping Zhang, and Jianfeng Feng. GaitSet: Regarding gait as a set for cross-view gait recognition. In *AAAI*, 2019.

[2] Shiqi Yu, Daoliang Tan, and Tieniu Tan. A framework for evaluating the effect of view angle, clothing and carrying condition on gait recognition. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 4, pages 441–444, 2006.

[3] Chao Fan, Yunjie Peng, Chunshui Cao, Xu Liu, Saihui Hou, Jiannan Chi, Yongzhen Huang, Qing Li, and Zhiqiang He. Gaitpart: Temporal part-based model for gait recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.