GAIT RECOGNITION BASED ON CONVOLUTIONAL NEURAL NETWORKS

A. Sokolova[a], A.Konushin[a, b]

[a] National Research University Higher School of Economics, Moscow, Russia - ale4kasokolova@gmail.com, akonushin@hse.ru
[b] Lomonosov Moscow State University, Moscow, Russia anton.konushin@graphics.cs.msu.ru

Commission II, WG II/10

KEY WORDS: Gait Recognition, Biometrics, Convolutional Neural Networks, Optical Flow

ABSTRACT:

In this work we investigate the problem of people recognition by their gait. For this task, we implement deep learning approach using the optical flow as the main source of motion information and combine neural feature extraction with the additional embedding of descriptors for representation improvement. In order to find the best heuristics, we compare several deep neural network architectures, learning and classification strategies. The experiments were made on two popular datasets for gait recognition, so we investigate their advantages and disadvantages and the transferability of considered methods.

1. INTRODUCTION

Gait recognition has always been a challenging problem. It is shown in medical and physiological studies that human gait is a unique identifier. Unlike other biometrical indices such as fingerprints, face, or iris recognition, it has a very important advantage: it can be measured at a large distance without any interaction with the human. This feature makes gait recognition applicable to intelligent video surveillance problems used, for example, in the security field. However, there are many factors affecting gait performance such as viewpoints, various carrying conditions, clothing, physical and medical conditions, etc. Due to these factors, gait recognition became a difficult problem of computer vision.

Formally the problem statement is similar to face or fingerprint recognition, but as the input, we use video sequences where person's full height walk is recorded. This problem can also be considered as action recognition, but actually, the human walking styles are usually much closer to each other than different activities that are included in popular datasets. It is worth noting that unlike many other problems of computer vision this one is difficult even for a human because the walking styles of people often look very similar to each other and the difference is hardly noticed.

The main goal of this work is not only to construct a classifier that can solve the problem for certain set of people but to build a feature extractor to be able to recognize new people not included in the initial dataset. In addition, since all the datasets available for gait recognition are not very large, the problem of transfer learning becomes really important.

Most of the approaches to gait recognition proposed in recent years are based on a model of motion using silhouette masks or pose estimations. Unlike these methods, we extract gait features from motion itself training convolutional neural network (CNN). CNN models are very successful in many computer vision problems, but their first application to gait recognition was made not long ago in (Castro et al., 2016). And we are going to push the performance further.

Note that most of the characteristics of appearance (such as the color or contrast) do not influence the gait properties, only the movements of human body points matter. So, we suppose that optical flow contains all the information about motion needed for gait recognition and therefore we use only the maps of optical flow to solve this problem.

2. RELATED WORK

The overwhelming majority of the state-of-the-art approaches to gait recognition are based on hand-crafted features of body motion: pose estimations or just silhouettes. One of the most popular descriptors based on silhouette is Gait Energy Image (GEI) (Han and Bhanu, 2006), the average image of the binary silhouette masks of the subject over the gait cycle. After the GEI was invented many authors began proposing computation various popular descriptors from these images, for example, HOG descriptor in (Liu et al., 2012). In (Chen and Liu, 2014) the modification of GEI is proposed – frame difference energy image (FDEI). Instead of averaging all normalized silhouettes over the gait cycle, they take the difference between every pair of consecutive frames and combine it with "denoised" GEI. Due to their report, such modification improves the metrics of quality. Zheng in (Zheng et al., 2011) applies a special feature selection algorithm called Partial Least Square regression to GEI in order to learn optimal vectors representing gait. One more approach is based on the inner model of gait computed using various measurements of human body and motion. (Yang et al., 2016) proposes to measure relative distances between joints, their mean and standard deviation, make a feature selection and then train a classifier based on selected descriptors.

Another approach to gait recognition is based on deep learning and does not use any handcrafted features. All features are trained inside the neural network on their own. Convolutional neural networks are now very popular in different problems concerned with video recognition and achieve the highest results. In (Simonyan and Zisserman, 2014) it is proposed to construct a two-stream convolutional neural network for action recognition. The first stream, similarly to many other approaches, is fed by distinct frames to get the spatial presentation of video, and the second stream gets blocks of optical flow maps as input and extracts the temporal information from the video. A lot of researchers continue the investigation of such multistream approach and consider

different methods of stream fusion and feature extraction. For example, in (Feichtenhofer et al., 2016) authors propose various functions for stream fusion and locations of fusing layers in net architecture. Ng in (Ng et al., 2015) uses blocks of raw frames as well as optical flow maps and deals with them as equals. Additionally to common feature pooling recurrent neural network is proposed to aggregate gait descriptors. Unlike all these methods (Wang et al., 2015) proposes to get temporal information not from the optical flow, but from the trajectories computed from the video. Combining these trajectories with spatial features extracted from neural network gives enough information for gait recognition.

Our approach is based on the idea from (Simonyan and Zisserman, 2014). As the appearance is not important in the gait recognition problem we get rid of the spatial stream and investigate only temporal one.

3. PROPOSED METHOD

Let us describe the approach we use to classify the gait video sequence. As we have already mentioned it is based on CNN that is used to compute motion features of gait. The algorithm consists of three main parts:

1. Preparing the data to be used as input to neural network;
2. Extracting neural features;
3. Classification of found features.

Let us observe these three steps in details.

3.1 Preparing the input data

Since the appearance (color, brightness, etc.) is not important when we consider the gait, we use the maps of optical flow for human classification. In order to use not only instantaneous but continuous motion, we concatenate several consecutive maps of optical flow and use such blocks as network input. Specifically, we cut a square containing the human figure from the whole optical flow block and feed the network with it. Let $\{I_t\}_{t=1}^{T}$ be the video sequence such that there is a human figure on each frame. Firstly we compute the sequence of optical flow (OF) maps for every consecutive pair of frames: $\{(F_t^V, F_t^G)\}_{t=1}^{T-1}$, where F_t^V and F_t^G are vertical and horizontal components of the optical flow between I_t and I_{t+1} frames, respectively. We then decrease the resolution of the maps to 88×66 not to make the network input too large and make a linear transformation of all maps, all the values of OF to lie in the interval $[0, 255]$ similarly to common RGB frames.

In addition we find the bounding boxes $\{B_t\}_{t=1}^{T}$ containing the figure for every frame. It is convenient to encode each box by the coordinates of its upper left and lower right vertices $B_t = (x_t^l, y_t^u, x_t^r, y_t^l)$. Having the bounding box for each frame we construct the outer box that contains all the boxes in block inside. So, for block that will consist of the OF maps for frames $I_k, I_{k+1}, \ldots, I_n$, the box coordinates are $x^l = \min_{t=k,\ldots,n}(x_t^l)$, $y^u = \min_{t=k,\ldots,n}(y_t^u)$, $x^r = \max_{t=k,\ldots,n}(x_t^r)$, $y^l = \max_{t=k,\ldots,n}(y_t^l)$.

These two computational steps are made beforehand. Immediately before putting the data into the network the certain number

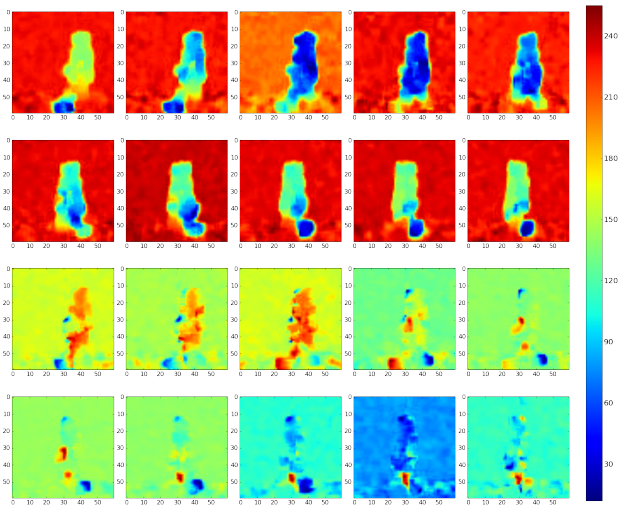

Figure 1. Input block. The square box cropped from 10 consecutive frames of horizontal (the first and the second rows) and vertical (the third and the fourth rows) components of OF maps linearly transformed to the interval $[0, 255]$.

L of consecutive horizontal and vertical components of OF maps are concatenated to one block and the square 60×60 containing the outer bounding box is cropped from the block. So, we get a block of size $60 \times 60 \times 2L$.

We considered various values of L and arrangement of the blocks but the best results were received having $L = 10$ and 5 frame overlap of the blocks. So, k-th block consists of the maps $\{(F_t^V, F_t^G)\}_{t=5k-4}^{5k+5}$, $k \geqslant 1$. An example of the frames in one cropped block is shown in Figure 1.

3.2 Extracting neural features

We consider few architectures of convolutional neural networks and compare them. Before describing these architectures let us explain the training and testing process. During the training process, we construct the blocks by random cropping a square containing the bounding box of the figure. If both sides of the box are less than 60 pixels we uniformly choose the left and upper sides (x and y, respectively) of the square so that x is between the left border of the frame and the left side of the box and $x + 60$ is between the right side of the box and the right border of the frame. The upper bound y of the square is chosen similarly. If any of the box sides is greater than 60 pixels we crop the square with the side equal to the maximum of height and width of the box and then resize it to 60. In addition, we flip all frames of the block simultaneously with probability 50% to get the mirror sequence. These two transformations help us to increase the amount of the training data and to reduce the overfitting of the model. Besides, before inputting the blocks into the net we subtract the mean values of horizontal and vertical components of optical flow over all training dataset.

When the net is trained it is used as a feature extractor. We remove its dense layers and use the outputs of the last convolutional layers as gait descriptors. To get the descriptor corresponding to certain block we do not flip any frames and make a crop choosing the mean values of the domains of left and upper bounds.

CNN architectures and training methods

Several CNN architectures were considered. We started from the architecture used in (Castro et al., 2016) that is based on the CNN-M architecture from (Chatfield et al., 2014) and used model based on this architecture as a baseline. The only modification we made is using Batch Normalization layer (Ioffe and Szegedy, 2015) instead of Local Response Normalization used initially. The architecture is described in the following table.

C1	C2	C3	C4	F5	F6	SM
7x7	5x5	3x3	2x2			
96	192	512	4096	4096	2048	soft-
Norm	stride 2	stride 2	-	d/o	d/o	max
pool 2	pool 2	pool 2	-			

Table 1. The baseline architecture

The first four columns correspond to convolutional layers. The first one has filters 7×7, the second one 5×5, the third one has 3×3 filters and the last layer has the filters of size 2×2. The next row shows the size of each layer (the number of filters) and the last two rows correspond to additional transformations such as normalization (batch normalization after the first convolution), max-pooling and stride. The last three layers are fully-connected. The fifth and sixth layers consist of 4096 and 2048 units, respectively, and they both are followed by dropout layers with parameter $p = 0.5$. The first six layers are followed by ReLU non-linearity. The last one is a predictional layer with the number of units equal to the number of subjects in the training set and softmax non-linearity. The model with such architecture was trained by Nesterov Momentum gradient descent method with starting learning rate equal to 0.1 reducing by a factor of 10 each time the error became constant.

Despite not very large size of the whole net, it was trained in four steps. Starting with the fourth, fifth and sixth layers of size 512, 512 and 256, respectively, we doubled the sizes of these layers when the loss function stopped decreasing. We considered two ways of widening the layers: adding random values of new parameters and net2net method (Chen et al., 2016). The latter method really does not lose any information, and the validation quality of the net before widening and after it is the same. However, it turns out, that such a saving of information got from previous training steps increases the overfitting and worsens the quality. And when we add randomly initialized new weights the random component works as a regularization and improves the classification accuracy.

The first modification of the CNN architecture is a slow fusion of several branches of the same net (similar to the idea from (Karpathy et al., 2014)). Instead of one big block taken as an input, we use 4 consecutive blocks of the smaller size. In more detail, we get 4 blocks of OF maps, each one consisting of 5 pairs of maps, and pass them through distinct branches of convolutional layers with shared weights. After the fourth convolutional layer, the outputs of the first two branches and the last two ones are concatenated so that we get two streams instead of four. These two outputs are passed through the fully-connected fifth layer after which two outputs are concatenated to one and continue passing through the net. This approach allows us to take a larger period of walking into account without losing instantaneous features. This net was trained stage-by-stage, as well.

The third architecture we have experimented with is similar to the architectures from VGG family (Simonyan and Zisserman, 2015). Because of the small sizes of the databases available for gait recognition, deep networks, such as VGG-16 are difficult to train without overfitting, so, we removed the last block of convolutions from VGG-16 architecture and trained such network. The details are described in Table 2.

B1	B2	B3	B4	F5	F6	SM
3x3,64	3x3,128	3x3,256	3x3,512			
3x3,64	3x3,128	3x3,256	3x3,512	4096	4096	soft-
		3x3,256	3x3,512	d/o	d/o	max
pool 2	pool 2	pool 2	pool 2			

Table 2. The VGG-like architecture

The first four columns correspond to blocks of convolutional layers. All the convolutions have 3×3 filter, the first block consists of two layers of size 64, the second one consists of two layers with 128 filters each, and the third and the fourth blocks contain three layers with 256 and 512 filters, respectively. All the convolutional blocks are followed by a max-pooling layer. After convolutional blocks, there are three dense layers. Two of them consist of 4096 units and are followed by dropout with parameter $p = 0.5$, and the last one is softmax layer. All layers except the last one are followed by ReLU non-linearity. This net was also learned step by step, starting from the size of two fully connected layers equal 1024 and doubling it when the validation error stopped decreasing.

In all the architectures we added L_2 regularization to loss function in order to avoid too large weights and overfitting. The regularization parameter increased step by step from 0.02 to 0.05.

3.3 Classification methods

After the network is trained we use the last hidden layer outputs as gait descriptors and construct a classifier on these descriptors. We consider several ways of doing it.

The most trivial case is when the subjects from the training set are the same as the subjects from the testing one. In this case, the softmax classifier is trained simultaneously with the features and the outputs of the network can be used as the vectors of the probability distribution over all the subjects. Another way to classify the descriptors extracted from the net is the Nearest Neighbour (NN) classifier. This method, one of the simplest in machine learning, gives better results and can be generalized for the case when subjects from the training and testing sets differ from each other. In this case, the network is used to extract features and the classifier is fitted on the part of videos for testing subjects and tested on the rest of videos. This way of classification is much more general because unlike the previous one, we do not have to retrain or fine-tune the net if we get new subjects that were not included in the training set. We use new labeled data only to fit a classifier based on neural features which is much quicker than retraining.

Additionally to the classical NN approach, when we measure the Euclidian distance between vectors, we considered NN classifier based on Manhattan distance. It turns out that this metrics is more stable and appropriate for measuring the similarity of gait descriptors and shows a better result in the majority of experiments.

Construction of NN classifier on network outputs is a baseline which was modified in several ways to improve the quality of the classification.

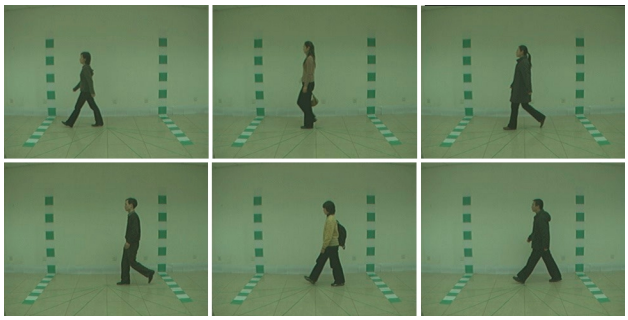

Figure 2. CASIA Gait Dataset B. Frames are captured from the side view under the angle 90°. Different conditions are shown in each column: normal walk in the first one, carrying bag in the second one, and wearing outdoor clothing in the third column.

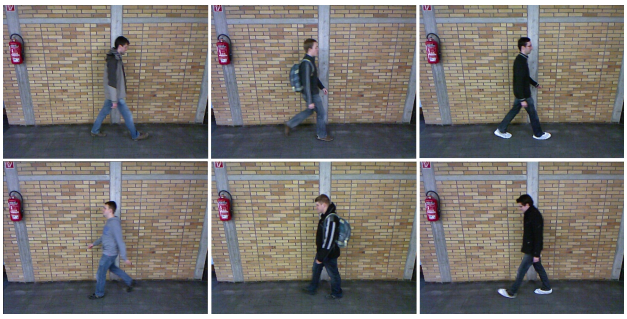

Figure 3. TUM-GAID Dataset. People are walking from left to right (top row) and from right to left (bottom row) with different conditions: normal walk (the first column), carrying a bag (the second column), and wearing coating shoes (the third column).

First of all, principal components analysis (PCA) algorithm can be made to reduce the dimensionality of feature vectors and get rid of any noise they contain. Besides, shortening of vectors accelerates the fitting of any classifier.

The further recognition improvement can be made by Triplet Probability Embedding (TPE) (Sankaranarayanan et al., 2016). It is low-dimensional discriminative embedding learned using triplet probability. This method uses PCA transformation as starting data and solves an optimization problem that improves the discriminative ability of data. This optimization problem can be formulated as follows: we define a function S_W of similarity between two vectors, so that for each triplet $t = (v_i, v_j, v_k)$, where v_i and v_j belong to the same class, and v_k is from a different class

$$S_W(v_i, v_j) > S_W(v_i, v_k) \tag{1}$$

The function S_W can be parametrized by the embedding matrix W: $S_w(v, u) = (Wv)^T(Wu)$. Hence, the problem of finding S_W reduces to finding W using Stochastic Gradient Descent. We changed this approach a bit using the Euclidian distance instead of the scalar product and the distance function so that the distance between the embeddings of vectors from the same class is less than from the different classes.

Additionally to improved performance, this embedding maintains the advantages of dimensional reduction such as memory and time effectiveness.

Let us now describe how we find the final result for requested video. Having the video sequence separated into blocks of OF maps, we get a descriptor for each of these blocks. We fit a classifier based on distinct blocks, so, we get a vector of distribution over all the subjects for every block. To get the distribution for a whole video we compute an average of all these vectors. We considered the voting method, as well, but averaging gives higher accuracy of classification, thus we have chosen this method.

4. DATA AND EXPERIMENTS

4.1 Datasets

We have evaluated the described methods on two most popular and challenging gait databases: CASIA Gait Dataset B (CASIA gait database., 2001) and "TUM Gait from Audio, Image and Depth" (TUM-GAID) dataset (Hofmann et al., 2014).

CASIA Dataset is a large database consisting of videos for 124 captured from 11 different viewing angles (from 0° to 180° with the step 18°). The videos have length 3-5 seconds, the frame rate about 25 fps, and the resolution 320×240 pixels. All the trajectories are straight and recorded indoor in the same room for all people. Besides the variety of the viewpoints, different clothing and carrying conditions are presented: wearing a coat and carrying a bag. In total, we have 10 video sequences for each person captured from each view: 6 normal walks without extra conditions, two walks in an outdoor clothing and two walks with a bag. Some examples of the frames are shown in Figure 1. CASIA dataset is very popular due to the fact that it is multiview. Nevertheless, there are too few subjects to train even small neural network to recognize gait from all views without overfitting. In the most of the experiments made with this database by other researchers (for example, (Chen and Liu, 2014)) the subjects were not split in training and testing parts, only videos for each person were. In this case, we have 124 classes and training and testing sets are equal. Experiments with such split show high classification accuracy, but they are not general and can not be used in real life without regular retraining. In order to get general results comparable with other datasets, we have renounced almost all viewpoints and used only 90° angle recordings. We used videos for 60 subjects for training the net and the rest 64 subjects for evaluating.

The second database we evaluated all the methods on is TUM-GAID dataset. It contains videos for 305 subjects going from left to right and from right to left captured from the side view. The frame rate of videos is about 30 fps and the resolution of each frame is 320×240. Similarly to the first database, TUM-GAID contains videos with different walking conditions: 6 normal walks, 2 walks carrying a backpack, and 2 walks wearing coating shoes for each person. Although there are recordings only from one viewpoint, the number of subjects allows us to take around a half of the people as training set and another half for evaluation. The authors of the database have provided the split into training, validation, and testing sets, which we used uniting training and validation sets into one so that the net was trained using the data for 150 subjects and tested with the other 155. Hence, no randomness influenced the result.

In our experiments with both datasets, we split the testing set into fitting and evaluating parts. The fitting part consists of the first 4 videos of normal walks for each person and is used for fitting the classifier and training TPE, and the evaluating part consists of the rest 6 videos and is used only to compute the accuracy of

classification. It is worth noting that although the NN classifier is fitted using only fitting videos of testing subjects, the TPE is trained much quicker and better if we train it by both training and fitting sequences (all the information that is supposed to be known in the algorithm).

4.2 Performance evaluation

All the algorithms have a classifier on their top that returns the vector of the distribution of the requested video sequence over all testing subjects. We use *Rank-k* metrics equaled the ratio of samples for which the correct label belongs to the set of k most probable answers to evaluate the results. *Rank-1* metric is accuracy, that defines the ratio of the correctly classified videos. Besides the accuracy, we measure *Rank-5* metrics.

4.3 Experiments and results

We evaluated all the methods described above on two considered datasets. All the experiments can be divided into three parts.

1. Distinct learning and evaluating on each of the datasets;
2. Learning on one dataset and evaluating on both of them;
3. Joint learning on two datasets and evaluating on both of them.

These experiments aim to estimate the influence of different methods on the quality of classification and to learn how general all these methods are. Ideally, the algorithm should not depend on the background conditions, the illumination, or anything other than the walking style of a person. So, it is worth evaluating how the algorithm learned on one dataset works on the other one.

The first set of the experiments is made to evaluate the proposed approach. We trained a net from scratch with the top layer of size equal the number of training subject. Then we trained a classifier on the learned descriptors using training and fitting data. The results of the experiments are shown in Table 3.

Method	Evaluation	
Architecture and Metrics	Rank1	Rank5
CNN (PCA 1100), L_1	93,22	98,06
CNN (PCA 1100), L_2	92,79	98,06
CNN (PCA 600), L_1	93,54	98,38
CNN+TPE (PCA 450), L_2	94,51	98,70
fusion CNN (PCA 160), L_1	93,97	98,06
fusion CNN (PCA 160), L_2	94,40	98,06
fusion CNN +TPE (PCA 160), L_1	94,07	98,27
fusion CNN +TPE (PCA 160), L_2	95,04	98,06
VGG (PCA 1024), L_1	97,20	99,78
VGG (PCA 1024), L_2	96,34	99,67
VGG+TPE (PCA 800), L_1	**97,52**	**99,89**
VGG+TPE (PCA 800), L_2	96,55	99,78
(Castro et al., 2016) CNN+SVM, L_2	98,00	99,60

Table 3. Results on TUM-GAID dataset

Although the accuracy of our method does not exceed Castro's Rank-1, the second metrics is higher in our method. So, the correct label is more often among the most probable ones.

The experiments show that training TPE having fewer dimensions always increases the accuracy as compared to common NN classifier with L_2 metrics. Nevertheless, L_1 metrics often turns out to be more successful. Despite the fact that TPE was trained as optimization of Euclidean similarity, the best result is achieved on the NN classifier based on L_1 distance between the embeddings of the descriptors. Let us show the Table 4, where it is written in details what quality we get with different conditions.

	Normal		Backpack		Shoes		Avg	
Method	R1	R5	R1	R5	R1	R5	R1	R5
VGG-like, L_1	99,7	100	96,5	99,7	96,5	100	97,5	99,8
CNN+SVM, L_2 (Castro et al., 2016)	99,7	100	97,1	99,4	97,1	99,4	98,0	99,6

Table 4. Results for different conditions on TUM-GAID dataset

Knowing that VGG-like architecture gives the best result, we evaluated it on CASIA dataset. The accuracy on this dataset turned out to be much lower: $71,80\%$ of correct answers using L_2 distance and a bit more – $74,93\%$ when L_1 is used.

Having the nets trained on each of the datasets we made the second set of the experiments investigating the generality of the algorithm – the experiments on transfer learning. TUM-GAID videos and the side view of CASIA look similar, so, we checked if one algorithm can be used for both datasets. Despite good results of the algorithms on TUM-GAID dataset, it turns out that the accuracy of the same algorithm with the same parameters on CASIA dataset is very low. The same happens on the other side when the net trained on CASIA is used for extracting features for videos from TUM. These experiments were made using the best of considered architectures – cropped VGG and the NN classifier based on L_1 metric without any extra embedding. Table 5 shows the accuracy of such transfer classification.

Training Set \ Testing Set	CASIA	TUM
CASIA	74,93	67,41
TUM	58,20	97,20

Table 5. Results of transfer learning

All the results on CASIA are lower than on TUM-GAID and transfer of the network weights noticeably worsens the initial quality.

The third part of experiments aims to investigate if the union of all available data improves the recognition. All subjects in two databases are distinct, so, we can unite them to one dataset and train the net on this union. The evaluation of the algorithm was made individually for each dataset.

	Result on TUM		Result on CASIA	
Method	Rank1	Rank5	Rank1	Rank5
VGG + NN, L_1 balancing learning	96,45	99,24	72,06	84,07
VGG + NN, L_2 balancing learning	96,02	99,35	70,75	83,02

Table 6. Results of join learning

Because of the difference in sizes of two datasets we use, we had to balance the input batches while training the net. Combining the batch of OF blocks we use the same number of blocks from both datasets, so, the net sees samples from the small dataset more frequently than from the large one. Despite such joint learning, the quality of the algorithm is lower than the quality of distinct

algorithms for each dataset. It means the overfitting in latter ones and the presence of dropout and regularization does not prevent it.

5. IMPLEMENTATION DETAILS

We prepare the data using OpenCV library. The silhouette masks and bounding boxes were found by background subtraction, and the maps of optical flow were computed using Farneback (Farneback, 2003) method implemented in OpenCV. For CNN training Lasagne and Theano libraries were used. All the experiments were performed on a computer with 12 cores, 32 GB of RAM and a GPU Nvidia GTX 1070. The most successful of considered architectures, the VGG-like one, was trained from scratch on TUM-GAID dataset in about 16 hours. When the net is trained it takes approximately 0.43 seconds to calculate all block descriptors for one four-second video and to classify it (including 0.14 seconds of preprocessing).

6. CONCLUSIONS AND FURTHER WORK

The experiments show that available datasets are too small and not enough for training general stable algorithm even for one certain viewpoint. Having 155 subjects in the testing set and 6 testing videos for each of them we get only 930 testing samples (and even fewer for a smaller database). It means that one correct answer changes the accuracy by about 0.1%, so the few percent difference may be not significant.

Therefore the most important step for further improvements is to collect new large gait dataset in order to have enough data for training complex deep models. Having appropriate database it is worth considering new CNN architectures and training methods.

REFERENCES

CASIA gait database., 2001. Online available: http://www.cbsr.ia.ac.cn/english/Gait%20Databases.asp.

Castro, F., Marin-Jimenez, M. and Guil, N.and Perez de la Blanca, N., 2016. Automatic learning of gait signatures for people identification. http://arxiv.org/abs/1603.01006.

Chatfield, K., Simonyan, K., Vedaldi, A. and Zisserman, A., 2014. Return of the devil in the details: Delving deep into convolutional nets. In: *Proc. BMVC*.

Chen, J. and Liu, J., 2014. Average gait differential image based human recognition. *Scientific World Journal*.

Chen, T., Goodfellow, I. and Shlens, J., 2016. Net2net: Accelerating learning via knowledge transfer. in international conference on learning representation. In: *ICLR*.

Farneback, G., 2003. Two-frame motion estimation based on polynomial expansion. In: *Proc. of Scandinavian Conf. on Image Analysis*, Vol. 2749, p. 363370.

Feichtenhofer, C., Pinz, A. and Zisserman, A., 2016. Convolutional two-stream network fusion for video action recognition. In: *CVPR*.

Han, J. and Bhanu, B., 2006. Individual recognition using gait energy image. In: *IEEE PAMI*, Vol. 28(2), p. 316322.

Hofmann, M., Geiger, J., Bachmann, S., Schuller, B. and Rigoll, G., 2014. The tum gait from audio, image and depth (gaid) database: Multimodal recognition of subjects and traits. *J. of Visual Com. and Image Repres.* 25(1), pp. 195 – 206.

Ioffe, S. and Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: *ICML*.

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R. and Fei-Fei, L., 2014. Large-scale video classification with convolutional neural networks. In: *CVPR*.

Liu, Y., Zhang, J., Wang, C. and Wang, L., 2012. Multiple hog templates for gait recognition. In: *Proc. ICPR*, pp. 2930–2933.

Ng, J., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R. and Toderici, G., 2015. Beyond short snippets: Deep networks for video classification. In: *CVPR*.

Sankaranarayanan, S., Alavi, A., Castillo, C. and R., C., 2016. Triplet probabilistic embedding for face verification and clustering. arXiv preprint arXiv:1604.05417.

Simonyan, K. and Zisserman, A., 2014. Two-stream convolutional networks for action recognition in videos. In: *NIPS*, p. 568576.

Simonyan, K. and Zisserman, A., 2015. Very deep convolutional networks for large-scale image recognition. In: *ICLR*.

Wang, L., Qiao, Y. and Tang, X., 2015. Action recognition with trajectory-pooled deep-convolutional descriptors. In: *CVPR*, p. 43054314.

Yang, K., Dou, Y., Lv, S., Zhang, F. and Lv, Q., 2016. Relative distance features for gait recognition with Kinect. https://arxiv.org/abs/1605.05415.

Zheng, S., Zhang, J., Huang, K., He, R. and Tan, T., 2011. Robust view transformation model for gait recognition. In: *Proc. IEEE International Conference on Image Processing (ICIP)*, pp. 2073–2076.