Dynamic Texture Video Classification Using Extreme Learning Machine

Liuyang Wang[1,2], Huaping Liu[1,2,⋆], and Fuchun Sun[1,2]

[1] Department of Computer Science and Technology, Tsinghua University, Beijing, China
[2] State Key Laboratory of Intelligent Technology and Systems, Beijing, China
hpliu@mail.tsinghua.edu.cn

Abstract. Recognition of complex dynamic texture is a challenging problem and captures the attention of the computer vision community for several decades. Essentially the dynamic texture recognition is a multi-class classification problem that has become a real challenge for computer vision and machine learning techniques. Existing classifier such as extreme learning machine cannot effectively deal with this problem, due to the reason that the dynamic textures belong to non-Euclidean manifold. In this paper, we propose a new approach to tackle the dynamic texture recognition problem. First, we utilize the affinity propagation clustering technology to design a codebook, and then construct a soft coding feature to represent the whole dynamic texture sequence. This new coding strategy preserves spatial and temporal characteristics of dynamic texture. Finally, by evaluating the proposed approach on the DynTex dataset, we show the effectiveness of the proposed strategy.

Keywords: Extreme learning machine, affinity propagation, dynamic texture.

1 Introduction

Extreme Learning Machine (ELM), which was firstly proposed by Huang [1], has become an effective learning algorithm for various classification tasks. It works on a simple structure named Single-hidden Layer Feed-forward Neural networks (SLFNs) and randomly applies computational hidden nodes. This mechanism is different from the conventional learning of SLFNs. ELM yields better performance than other conventional learning algorithms in application with higher noise. It also has an extremely fast learning speed compared with traditional gradient-based algorithms. Furthermore, ELM technique successfully overcomes the difficulty of the curse of dimensionality [2,3]. Currently, the application scope of ELMs covers face recognition [4,5], action recognition [6], video concept detection [7], and so on. Ref.[8] gives a comprehensive survey about ELM and Ref.[9] provides a deep insight into ELM. For time-series, Ref.[10] developed time-series processing of large scale remote sensing data with ELM. Ref.[11] studied the

⋆ Corresponding author.

J. Cao et al. (eds.), *Proceedings of ELM-2014 Volume 2,*
Proceedings in Adaptation, Learning and Optimization 4, DOI: 10.1007/978-3-319-14066-7_5

cross-person activity recognition using reduced kernel ELM. However, for general visual dynamic textures, which belong to the non-Euclidean manifold, how to realize effective classification using ELM is still an open problem.

Dynamic textures (DT) are video sequences of non-rigid dynamical objects that constantly change their shape and appearance over time. Some examples of dynamic textures are video sequences of fire, smoke, crowds, and traffic. These classes of video sequences are ubiquitous in our natural environment.

However, the classification of DT admits great challenging since DT include both spatial and temporal elements. To model DT, Ref.[12] developed a linear dynamic system (LDS) method. LDS can be used to model complex visual phenomena with a relatively low dimensional representation. However, the signal would rapidly decay. The work in [13] extended this work by introducing feedback control and modeled the system as a closed loop LDS. The feedback loop corrected the problem of signal decay. In [14], LDS was regarded as educational bridge to merge the gap between computer science and control engineering.

A difficult problem to use LDS for classification lies in the fact that LDS does not lie in an Euclidean space, and therefore many classification cannot be utilized. In [12], the Martin distance between LDS is adopted to compare the similarity between different DTs. Martin distance, is effective to evaluate the distance between LDS, but cannot be effectively used for video with multiple dynamic textures. Therefore, many works which utilize Martin distance are limited to investigate simple video with single DT.

Very recently, Ref.[15] proposed to categorize DT by using a novel Bag-Of-dynamic-Systems (BoS). It models each video sequence with a collection of LDSs, each one describing a small spatial-temporal patch extracted from the video. This BoS representation is analogous to the Bag-of-Words (BoW) representation for object recognition. This choice provides an effective strategy to deal with video sequences which are taken under different viewpoints or scales.

Because BoS model is similar to BoW, it naturally inherits the disadvantages of BoW. It is well known that BoW model assign only one codebook element to a descriptor, and therefore the quantization error is large. This usually degrades the classification performance. In [16] and [17], the sparse coding and local coding methods are proposed to address such problem. In such frameworks, more than one codebook element will be assigned to a descriptor and form coding vector. Such strategy can obviously improve the classification performance since the quantization error is attenuated. Unfortunately, neither sparse coding nor local coding can be used for BoS. The intrinsic reason is that both methods depends on the linear subspace assumption and used the linear reconstruction error to design the object function.

Another problem lies in the design of codebook. Conventional k-means or k-mediod clustering method requires the user to prescribe the size of the codebook. This is an non-trivial task.

In this paper, we present a new representation of DT for ELM classifier design. We use the LDS to model the spatial-temporal patches which is sampled from the original video sequence, and adopt the affinity propagation (AP) to

design the codebook. AP algorithm is originally proposed in [18] to deal with exemplar extraction problem. The reason to adopt AP lies in two facts: (1) It can automatically determine the size of the codebook; (2) It can extract the existing DT exemplar to construct the codebook. After that, we develop a soft assignment coding method to design the DT representation. The experimental results show the advantage of the proposed method.

The rest of this paper is organized as follows: In Section 2 we give an introduction about ELM. Section 3 presents the details of the proposed coding method. Section 4 shows the experimental results. In Section 5 we give some conclusions.

2 ELM Classification

In this section, we provide a brief introduction of the ELM algorithm which will be used in dynamic texture classification. The more details can be found in [2,6].

In the case of multiple classes, the training data are denoted as $\{(\mathbf{u}_i, l_i)\}_{i=1}^{N}$, where $\mathbf{u}_i$ is the feature vector, l_i is the label and N is the number of the training samples. For each vector $\mathbf{u}_i$, l_i should be transformed to the vector $\mathbf{t}_i = [t_{i1}, ..., t_{iC}]^T$, where $t_{ik} = 1$ for the vector belonging to class k, i.e., when $l_i = k$, and $t_{ik} = -1$ otherwise.

The input weights of ELM are randomly chosen, while the output weights should be analytically calculated. Assume that the network's hidden layer consists of Q neurons and that $\mathbf{b} \in \mathbb{R}^Q$ is a vector containing the hidden layer neurons bias values. Function $G(\cdot)$ used for output calculation is denoted as $G(\mathbf{w}_j, b_j; \mathbf{u}_i)$, where $\mathbf{w}_j$ denotes the input weight. The hidden layer neurons outputs $\mathbf{G}$ can be represented as:

$$\mathbf{G} = \begin{bmatrix} G(\mathbf{w}_1, b_1; \mathbf{u}_1) & \cdots & G(\mathbf{w}_1, b_1; \mathbf{u}_N) \\ \vdots & \ddots & \vdots \\ G(\mathbf{w}_Q, b_Q; \mathbf{u}_1) & \cdots & G(\mathbf{w}_Q, b_Q; \mathbf{u}_N) \end{bmatrix} \in \mathbb{R}^{Q \times N}$$

The network's output vector corresponding to the training vector set $\{\mathbf{u}_i\}_{i=1}^{N}$ can be written in a matrix form as

$$\mathbf{O} = \mathbf{W}_o^T \mathbf{G} \tag{1}$$

where $\mathbf{W}_o \in \mathbb{R}^{C \times Q}$ is the output weight.

By assuming that the network's predicted outputs $\mathbf{O}$ are equal to the network's desired outputs $\mathbf{T} = [\mathbf{t}_1, ..., \mathbf{t}_N]$, $\mathbf{W}_o$ can be analytically calculated by

$$\mathbf{W}_o = \left(\mathbf{G}\mathbf{G}^T\right)^{-1} \mathbf{G}\mathbf{T}^T \tag{2}$$

Taking account of training errors, Ref.[2] have recently proposed an optimization based on regularized ELM algorithm formulated as follows:

$$\begin{aligned} &\text{Minimize} : L_P = \frac{1}{2}\|\mathbf{W}_o\|^2 + C\frac{1}{2}\sum_{i=1}^{N}\|\xi_i\|^2 \\ &\text{Subject to} : \mathbf{g}_i^T \mathbf{W}_o = \mathbf{o}_i^T - \xi_i^T, i = 1, ..., N \end{aligned} \tag{3}$$

where $\xi_i \in \mathbb{R}^C$ is a training error vector corresponding to training sample $\mathbf{u}_i$ and C is a parameter denoting the importance of the training error in the optimization problem. By adopting the above described optimization scheme, $\mathbf{W}_o$ can be calculated by:

$$\mathbf{W}_o = \mathbf{G}\left(\frac{1}{C}\mathbf{I} + \mathbf{G}^T\mathbf{G}\right)^{-1}\mathbf{T}^T \tag{4}$$

Finally, the test data can be introduced to the ELM network and be classified to the class corresponding to the highest networks output.

Further, we use the Kernel trick to deal with complex dynamic texture videos. According to [2], we define a kernel matrix for ELM as:

$$\Omega = \mathbf{G}^T\mathbf{G}, \tag{5}$$

where the $\{i, j\}$ element in Ω is $\mathbf{g}_i^T\mathbf{g}_j = \mathcal{K}(\mathbf{u}_i, \mathbf{u}_j)$. Then, the output function of ELM classifier can be written as

$$\mathbf{o}^T = \mathbf{g}^T\mathbf{W}_o = \mathbf{g}^T\mathbf{G}\left(\frac{1}{C}\mathbf{I} + \mathbf{G}^T\mathbf{G}\right)^{-1}\mathbf{T}^T = \begin{bmatrix} \mathcal{K}(\mathbf{u}, \mathbf{u}_1) \\ \vdots \\ \mathcal{K}(\mathbf{u}, \mathbf{u}_N) \end{bmatrix}^T \left(\frac{1}{C}\mathbf{I} + \Omega\right)^{-1}\mathbf{T}^T \tag{6}$$

In this work, we adopt the Gaussian kernel which is represented as

$$K(\mathbf{u}, \mathbf{v}) = exp(-\gamma\|\mathbf{u} - \mathbf{v}\|^2) \tag{7}$$

where γ is the prescribed parameter.

The whole procedure to utilize ELM is illustrated in Fig.1. In the following sections we will give more details.

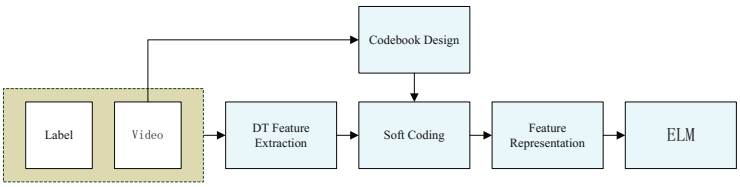

Fig. 1. The overview of the proposed method.

3 Video Representation

3.1 Dynamic Texture Modeling

This section summarizes the key concepts in dynamic texture. According to [12], an LDS model can be used to fit a small spatial-temporal patch. Assume the

short spatial-temporal patches includes τ frames with resolution $m \times n$. The dynamic texture should obeys the standard state-space equation:

$$\begin{cases} \mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{u}(k), & \mathbf{u}(k) \sim N(0, \mathbf{Q}), \quad \mathbf{x}(0) = \mathbf{x}_0 \\ \mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{w}(k), & \mathbf{w}(k) \sim N(0, \mathbf{R}) \end{cases} \tag{8}$$

where $\mathbf{x}(k) \in \mathbb{R}^{n_x}$ is the hidden state variable vector, and $\mathbf{y}(k) \in \mathbb{R}^{n_y} (n_y = m \times n)$ is the data corresponding to the sequence of images. $\mathbf{A} \in \mathbb{R}^{n_x \times n_x}$, and $\mathbf{C} \in \mathbb{R}^{n_y \times n_x}$ are the parameter matrices. $\mathbf{u}(k) \in \mathbb{R}^{n_x}$ and $\mathbf{w}(k) \in \mathbb{R}^{n_y}$ are the zero-mean normally distributed random variables, which are used to compensate the modeling error.

Since $\mathbf{w}(k)$ and $\mathbf{u}(k)$ are modeling errors, they are expected to be negligible. Therefore the problem can be formulated as: Given the observed image sequence $\mathbf{y}(1), \mathbf{y}(2), ..., \mathbf{y}(\tau) (\tau > n_x)$, estimate the values of $\mathbf{A}$, $\mathbf{C}$ and $\mathbf{x}(k)$.

Denote $\mathbf{Y}_{1:\tau} = [\mathbf{y}(1), \mathbf{y}(2), ..., \mathbf{y}(\tau)] \in \mathbb{R}^{n_y \times \tau}$ and $\mathbf{X}_{1:\tau} = [\mathbf{x}(1), \mathbf{x}(2), ..., \mathbf{x}(\tau)] \in \mathbb{R}^{n_x \times \tau}$. It can be reformulated as

$$\mathbf{Y}_{1:\tau} \approx \mathbf{C}\mathbf{X}_{1:\tau} \tag{9}$$

SVD can be performed on the matrix $\mathbf{Y}_{1:\tau}$ and the parameters are estimated as:

$$\begin{aligned} \mathbf{Y}_{1:\tau} &\approx \mathbf{U}\Sigma\mathbf{V}^T \\ \mathbf{C} = \mathbf{U} \quad and \quad \mathbf{X}_{1:\tau} &= \Sigma\mathbf{V}^T \end{aligned} \tag{10}$$

Because $\mathbf{x}(k+1) \approx \mathbf{A}\mathbf{x}(k)$ for $k = 1, 2, ..., \tau - 1$, the estimation of $\mathbf{A}$ can be uniquely determined by solving the following least squares problem:

$$\mathbf{A} = \underset{\mathbf{A}}{\operatorname{argmin}} \sum_{k=1}^{\tau-1} ||\mathbf{x}(k+1) - \mathbf{A}\mathbf{x}(k)||_2 \tag{11}$$

Finally, we can use the tuple $\mathbf{p} = (\mathbf{A}, \mathbf{C})$ to describe A DT spatial-temporal patch.

3.2 Codebook Design

To effectively code the obtained DT spatial-temporal patches, a reasonable codebook is needed. As the whole video is complicated , we sample non-overlapped spatial-temporal patches from videos and obtain the DT models of all the patches [19]. The codebook is generated from all the patch models in training data.

In this work, we adopt Affinity Propagation (AP) clustering algorithm [18] to find prototypes $\mathbf{V} = \{\mathbf{v}_1, ..., \mathbf{v}_K\}$ as the codebook. Such a method can automatically determine the codebook size K. Let $\mathbf{P} = \{\mathbf{p}_1, ..., \mathbf{p}_M\}$ be the set of all the DT patches. AP takes as input a collection of similarities between DT patches. The details of the algorithm are described in Algorithm.1.

A DT spatial-temporal patch $\mathbf{p}_i$ can be described as $\mathbf{p}_i = (\mathbf{A}_i, \mathbf{C}_i)$ by modeling dynamic textures. The similarities of $\mathbf{p}_i$ and $\mathbf{p}_j$ should be defined for AP

Algorithm 1. AP Clustering Algorithm

Input:

$s(i,j)(1 \leq i,j \leq M)$: similarities and preference;

Output:

$\mathrm{idx}(i)(1 \leq i \leq M)$: indices of exemplars for each $\mathbf{p}_i$;

initial $a(i,j) = 0$;
repeat
  compute responsibilities $r(i,j)$ $(1 \leq i,j \leq M)$;
    $r(i,j) = s(i,j) - \max_{j' \ s.t. \ j' \neq j} \left\{ a(i,j') + s(i,j') \right\}$
  compute availabilities $a(i,j)$ $(1 \leq i,j \leq M)$;
    $a(i,j) = \min \left\{ 0, r(j,j) + \sum_{i' \ s.t. \ i' \notin \{i,j\}} \left\{ \max\{0, r(i',j)\} \right\} \right\} \quad (i \neq j)$
    $a(j,j) = \sum_{i' \ s.t. \ i' \neq j} \left\{ \max\{0, r(i',j)\} \right\}$
  Identifying exemplars;
    $\mathrm{idx}(i) = \arg\max_{1 \leqslant j \leqslant M} a(i,j) + r(i,j)$
until $\mathrm{idx}(i)$ do not change.

clustering algorithm. One family of distances between two models is based on principal angles between specific subspaces derived from the models, namely the observability subspaces [15]. The observability subspaces is the range of the extended observability denoted by $O_\infty(\mathbf{p}_i) = \left[\mathbf{C}_i{}^T, (\mathbf{C}_i\mathbf{A}_i)^T, (\mathbf{C}_i\mathbf{A}_i{}^2)^T, ... \right]^T \in \mathbb{R}^{\infty \times n_x}$. Let θ_a be the a-th principal angles between the spaces. The Martin distance between $\mathbf{p}_i$ and $\mathbf{p}_j$ is defined as

$$d_M(\mathbf{p}_i, \mathbf{p}_j) = -\ln \prod_{a=1}^{n_x} \cos^2 \theta_a \tag{12}$$

After we obtain the distance matrix $\mathbf{D} \in R^{M \times M}$ and its element $\mathbf{D}(i,j)$ is the Martin distance between $\mathbf{p}_i$ and $\mathbf{p}_j$, we set $s(i,j) = -\mathbf{D}(i,j)^2$ and $s(i,i) = \min_{j=1,...,M} \left(-\mathbf{D}(i,j)^2 \right)$ for AP clustering algorithm. AP clustering provides two advantages: (1) the size of the codebook is automatically determined; (2) the result of the clustering is deterministic and do not depend on the initialization.

3.3 Soft Coding Feature Design

A popular method for coding is vector quantization which searches the nearest neighbor to represent the descriptor. Such a representation is usually called BoW. BoW quantizes a video into discrete "visual words", and then computes a histogram representation. One disadvantage of BoW is that it introduces significant quantization errors since only one element of the codebook is selected to represent the descriptor. To reduce the quantization error, we develop a soft coding approach to solve this problem.

Given a spatial-temporal patch which is denoted as $\mathbf{p}_i$, the corresponding feature vector is constructed as $\{\mu_{ik}\}_{k=1}^{K}$. In this representation, μ_{ik} is the membership value of sample $\mathbf{p}_i$ to the cluster identified by the center $\mathbf{v}_k^*$. The memberships can be obtained by

$$\mu_{ik} = \frac{e^{(-\beta \cdot d_M(\mathbf{p}_i, \mathbf{v}_k^*))}}{\sum_{k'=1}^{K} e^{\left(-\beta \cdot d_M(\mathbf{p}_i, \mathbf{v}_{k'}^*)\right)}} \tag{13}$$

Please note that $\{\mu_{ik}\}_{k=1}^{K}$ is normalized to satisfy $\sum_{k=1}^{K} \mu_{ik} = 1$.

The coding vector for the spatial-temporal patch $\mathbf{p_i}$ is then obtained as $\mathbf{c_i} = [\mu_{i1}, \mu_{i2}, ..., \mu_{iK}]^T \in \mathbb{R}^K$. For a single video sequence, if we extract N local DT descriptors, then we can get the codes $\mathbf{C} = [\mathbf{c}_1, ..., \mathbf{c}_N] \in \mathbb{R}^{K \times N}$ .Then we need an operator to pool all codes in a video into a single vector $\mathbf{u_i} \in \mathbb{R}^K$. This pooling operation is defined as

$$\mathbf{u_i} = \mathbb{P}(\mathbf{C_i}) \tag{14}$$

where the pooling function $\mathbb{P}$ is defined for each column of $\mathbf{C}$. Each column of $\mathbf{u_i}$ corresponds to the responses of all the local descriptors in the specific video. Therefore, different pooling functions construct different image statistics. In this study, we select the average operator. Such strategy results in

$$\mathbf{u}_i(k) = \frac{\sum_{j=1}^{N} |\mathbf{C}(k,j)|}{\sum_{k'=1}^{K} \sum_{j=1}^{N} |\mathbf{C}(k',j)|} \tag{15}$$

where $\mathbf{u}_i(k)$ is the k-th element of $\mathbf{u}_i(k)$ and $\mathbf{C}(k,j)$ is the matrix element in the k-th row and j-th column of $\mathbf{C}$.

4 Experimental Results

In this section, we present experimental results that validate the proposed algorithm. As indicated by [20], some existing DT data sets have a number of drawbacks such as the resolution is quite low; there is only a single occurrence per class and not enough classes are available for practical classification purposes. To tackle this problem, Ref.[20] developed the DynTex data set, which aims to serve as a reference database for dynamic texture research by providing a large and diverse database of high-quality dynamic textures. Dyntex provides 3 data sets for classification. In our experiment, we adopt the Gamma dataset for classification evaluation. It includes 10 classes such as sea, calm water, grass, etc. We randomly choose half of the videos in each class as the training data. The remaining videos are used for test purpose.

Table 1. Accuracy Comparisons

Method	Accuracy	Method	Accuracy
BoW-ELM	75.00%	*1-NN*	52.34%
BoW-SVM	75.78%	*3-NN*	49.22%
soft coding-ELM	88.28%	*5-NN*	40.62%
soft coding-SVM	82.81%		

In this work, the size of the codebook is never prescribed by the designer. After we run the AP procedure, we get a codebook for Gamma dataset. The obtained codebook size is 184.

As to the coding methods, we compare two methods: soft coding and hard coding. The hard coding method assigns only one codebook element to a local DT descriptor. It is equivalent to the usual BoW method. As to the classifiers, we compare three classifiers: ELM, SVM, and k-Nearest Neighbors(NN). Both ELM and SVM can be combined with two coding methods. For k-NN, we use a single DT model to model the video and use the Martin distance to measure the difference between videos. Such a method serves the role of baseline. In our implementation, we set $k = 1, 3$, and 5.

Table.1 lists the performance of all of the above methods. k-NN always obtains the worse results. This is not surprising because only a single DT is used to describe the video and therefore the modeling error will be significant. If we use BoW feature, both ELM and SVM obtain better results than k-NN, while ELM is a little worse than SVM. However, if we use soft coding method, the results of ELM will be dramatically increased and is superior to SVM. In Fig.2, we list the confusion matrice of ELM. From this figure we see the performance of ELM is rather good.

In the above comparison, the parameters of ELM and SVM are carefully tuned to get the best results. To show the influence of the parameters γ and C, we change the values of γ from 2^{-18} to 2^{18}, and the values of C from 2^{-18} to 2^{18}. The obtained performance results are listed in Figs.2. From this figure we see that the performance change of ELM is very smooth, while SVM is rather susceptible to the parameters.

Finally, to show the advantages of the obtained codebook, we also used the K-Medoids algorithm to design a codebook with the same size. Therefore we actually compare the following four methods: (1) AP-ELM(using AP and ELM); (2) AP-SVM(using AP and SVM); (3) KM-ELM(using K-Medoids and ELM); (4) KM-SVM(using K-Medoids and SVM).

We study the influence of the parameter β which plays important roles in the soft coding stage. In Fig.2 we list the accuracy versus the values of β. In addition to β, the other parameters in the algorithms are carefully-tuned to get the best results. From this figure we see that AP-ELM performs better than other methods and this shows that the AP codebook indeed helps ELM to get better results.

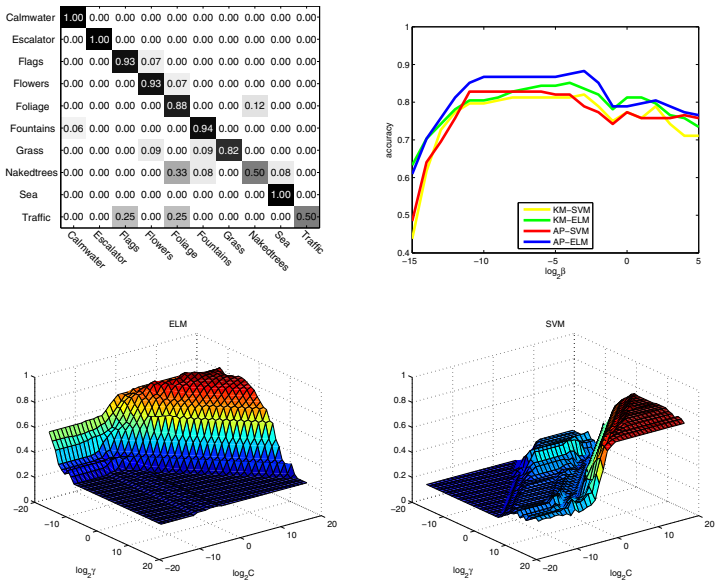

Fig. 2. 1st row: Confusion Matrix and Classification performance v.s. the parameter β; 2nd row: Classification performance v.s. the parameter γ and C

5 Conclusions

In this paper, the ELM classifier is developed to tackle the dynamic texture classification problem. Since the dynamic texture lies in the non-Euclidean space, we design a soft coding BoS representation for it. Such a representation can be used for ELM classifiers and obtains satisfactory performance on public datasets.

Acknowledgements. This work was supported in part by the National Key Project for Basic Research of China under Grant 2013CB329403; in part by the National Natural Science Foundation of China under Grant 91120011 and Grant 61210013; in part by the Tsinghua Self-innovation Project under Grant 20111081111; and in part by the Tsinghua University Initiative Scientific Research Program under Grant 20131089295.

References

1. Huang, G.B., Zhu, Q.Y., Siew, C.K.: Extreme learning machine: theory and applications. Neurocomputing 70(1), 489–501 (2006)
2. Huang, G.B., Zhou, H., Ding, X., Zhang, R.: Extreme learning machine for regression and multiclass classification. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 42(2), 513–529 (2012)

3. Savitha, R., Suresh, S., Kim, H.: A meta-cognitive learning algorithm for an extreme learning machine classifier. Cognitive Computation, 1–11 (2013)
4. He, B., Xu, D., Nian, R., van Heeswijk, M., Yu, Q., Miche, Y., Lendasse, A.: Fast face recognition via sparse coding and extreme learning machine. Cognitive Computation, 1–14 (2013)
5. Zong, W., Huang, G.B.: Face recognition based on extreme learning machine. Neurocomputing 74(16), 2541–2551 (2011)
6. Iosifidis, A., Tefas, A., Pitas, I.: Dynamic action recognition based on dynemes and extreme learning machine. Pattern Recognition Letters 34(15), 1890–1898 (2013)
7. Lu, B., Wang, G., Yuan, Y., Han, D.: Semantic concept detection for video based on extreme learning machine. Neurocomputing 102, 176–183 (2013)
8. Huang, G.B., Wang, D.H., Lan, Y.: Extreme learning machines: a survey. International Journal of Machine Learning and Cybernetics 2(2), 107–122 (2011)
9. Huang, G.B.: An insight into extreme learning machines: random neurons, random features and kernels. Cognitive Computation, 1–15 (2014)
10. Chen, J., Zheng, G., Fang, C., Zhang, N., Chen, H., Wu, Z.: Time-series processing of large scale remote sensing data with extreme learning machine. Neurocomputing 128, 199–206 (2014)
11. Deng, W.Y., Zheng, Q.H., Wang, Z.M.: Cross-person activity recognition using reduced kernel extreme learning machine. Neural Networks 53, 1–7 (2014)
12. Doretto, G., Chiuso, A., Wu, Y.N., Soatto, S.: Dynamic textures. International Journal of Computer Vision 51(2), 91–109 (2003)
13. Yuan, L., Wen, F., Liu, C., Shum, H.-Y.: Synthesizing dynamic texture with closed-loop linear dynamic system. In: Pajdla, T., Matas, J(G.) (eds.) ECCV 2004. LNCS, vol. 3022, pp. 603–616. Springer, Heidelberg (2004)
14. Liu, H., Xiao, W., Zhao, H., Sun, F.: Learning and understanding system stability using illustrative dynamic texture examples. IEEE Transactions on Education 57(1), 4–11 (2014)
15. Ravichandran, A., Chaudhry, R., Vidal, R.: Categorizing dynamic textures using a bag of dynamical systems. IEEE Transactions on Pattern Analysis and Machine Intelligence 35(2), 342–353 (2013)
16. Yang, J., Yu, K., Gong, Y., Huang, T.: Linear spatial pyramid matching using sparse coding for image classification. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1794–1801. IEEE (2009)
17. Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., Gong, Y.: Locality-constrained linear coding for image classification. In: 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3360–3367. IEEE (2010)
18. Frey, B.J., Dueck, D.: Clustering by passing messages between data points. Science 315(5814), 972–976 (2007)
19. Ravichandran, A., Chaudhry, R., Vidal, R.: Dynamic texture toolbox (2011), `http://www.vision.jhu.edu`
20. Péteri, R., Fazekas, S., Huiskes, M.J.: Dyntex: A comprehensive database of dynamic textures. Pattern Recognition Letters 31(12), 1627–1632 (2010)