CAAI Transactions on Intelligence Technology

Special Issue: Selected Papers from The 4th Asian Conference on Pattern Recognition (ACPR 2017)

Fast genre classification of web images using global and local features

ISSN 2468-2322
Received on 7th August 2018
Accepted on 8th August 2018
doi: 10.1049/trit.2018.1018
www.ietdl.org

Guo-Shuai Liu[1], Rui-Qi Wang[1,2], Fei Yin[1,2], Jean-Marc Ogier[3], Cheng-Lin Liu[1,2,4] ✉

[1]*National Laboratory of Pattern Recognition, Institute of Automation of Chinese Academy of Sciences, 95 Zhongguancun East Road, Beijing 100190, People's Republic of China*
[2]*School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, People's Republic of China*
[3]*L3i Laboratory, Faculty of Science and Technology, University of La Rochelle, 17042 La Rochelle Cedex 1, France*
[4]*Center for Excellence of Brain Science and Intelligence Technology, Beijing 100190, People's Republic of China*
✉ *E-mail: liucl@nlpr.ia.ac.cn*

Abstract: To effectively mine the contents embedded in web images, it is useful to classify the images into different types so that they can be fed to different procedures for detailed analysis. The authors herein propose a hierarchical algorithm for efficiently classifying web images into four classes. Their algorithm consists of two stages: the first stage extracts global features reflecting the distributions of color, edge and gradient, and uses a support vector machine (SVM) classifier for preliminary classification. Images assigned low confidence by the first stage classifier are processed by the second stage, which further extracts local texture features represented in the bag-of-words framework and uses another SVM classifier for final classification. In addition, they design two fusion strategies to train the second-stage classifier and generate the final prediction depending on the usage of local features in the second stage. To validate the effectiveness of proposed method, they built a database containing more than 55,000 images from various sources. On their test image set, they obtained an overall classification accuracy of 98.4% and the processing speed is over 27 fps on an Intel(R) Xeon(R) central processing unit (2.90 GHz).

1 Introduction

On the Internet and mobile network, the explosive growth of multimedia data including texts, images and videos brings us rich information and also the difficulty of efficiently mining relevant information. While the texts are explored by most web mining tools, to mine the contexts in images is also important. Particularly, the texts embedded in images provide easy understandable semantics and such images occupy a considerable proportion on web pages. A study [1] showed that 17% of the words visible on the web pages are in image form and a large proportion (76%) of text information embedded in images cannot be found anywhere in the web pages. The texts in images, however, are hard to extract by computers, though easily read by humans. For text detection and reading methods to process efficiently in the Internet environment, we need to quickly classify the images into different types of sources such that each type of images undertakes detailed analysis by a special procedure. Also, for accurate processing, different types of text images (document images) such as natural scene text images, born-digital images (BDIs), scanned and camera-captured paper documents (CPDs) are better analysed in different procedures.

In this paper, we propose a fast classification algorithm for classifying web images into four major types, namely natural scene images (NSIs), BDIs, scanned paper documents (SPDs) and CPDs. NSIs (photographs) are captured by surveillance cameras or mobile cameras and are most popular on the web. Whether they contain texts or need not to be judged using the more detailed procedure but the fast identification of this image type is helpful for the overall process of web image analysis. The other three types: BDIs, SPDs and CPDs usually contain rich texts. They also show different characteristics of image quality, e.g. BDIs usually have large areas of constant colour, and SPDs are more uniform in intensity and less distortion than CPDs. For a good tradeoff between classification accuracy and processing speed, our algorithm consists of two stages. The first stage uses global features capturing the difference of appearance between four types of images for preliminary classification with a support vector machine (SVM) classifier. Images assigned low confidence by the first-stage classifier are then processed by the second stage, which extracts local texture features encoded in the bag-of-words (BoW) framework and uses another SVM classifier for final classification. Compared to global features, local texture features are able to represent different patterns of colour transitions and properties of edges between four types of images in a more detailed way and yield higher classification accuracy. To validate the effectiveness of our proposed method, we built a large image database by collecting images from various sources such as web crawling, the camera capturing and other standard public databases. On our test image set, we obtained an overall classification accuracy of 98.4% and the processing speed is over 27 fps on a central processing unit (CPU) (2.90 GHz).

The rest of this paper is organised as follows. Section 2 briefly reviews related works; Section 3 describes the proposed method; Section 4 introduces the image database; Section 5 presents experimental results; and Section 6 makes a conclusion.

2 Related work

A large variety of feature extraction and classification methods have been proposed in the context of image classification and content-based image retrieval [2] but these existing methods are not directly applicable for our purpose of image genre classification. In the following, we outline some works related to our purpose.

Hammoud *et al.* [3] distinguished art paintings from scene photographs using colour texture signatures derived from the human visual system. The receptive field profiles and composite visual features they presented are helpful to solve our problem. Motivated by the physical image generation process, Ng *et al.* [4] proposed a

novel geometry-based model for classifying photographic images and computer graphics in the context of image forgery detection. They exploited global geometry information at different scales as well as local patch statistics to discover the distinctive physical characteristics of images such as the gamma correction of photographs and the sharp structures in graphics. Despite that the method was shown effective, the feature extraction there is very time-consuming, e.g. only global fractal geometry feature extraction takes 128.1 s on a 1280×1024 image. Athitsos *et al.* [5] presented a method for separating photographs and graphics on web pages. The graphics they considered such as corporate logos, maps and navigation buttons, are very simple even compared with our BDIs which contain both texts and graphics. Lienhart and Hartmann [6] also tried to solve this problem, and the metrics they designed depend mostly on statistics of global visual cues such as colour and edge orientation histogram. Lee *et al.* [7] tried to categorise images into art, photograph and cartoon using a neural network model. Five standard MPEG-7 visual descriptors [8] in their work were employed for extracting features such as Colour Layout, Colour Structure, Homogeneous Texture, Region Shape and Edge Histogram, which are not only redundant but also time-consuming. Pourashraf *et al.* [9] adopted an ensemble model for classifying images embedded in commercial real estate flyers into one of five genres: aerial photograph, map, inside building, outside the building and schematic drawing. However, the model was only evaluated with a small database and the processing speed was not reported.

In recent years, deep neural networks, especially the convolutional neural network (CNN) [10–15] has achieved a great success in image recognition tasks including image categorisation, object detection, scene text detection and recognition [16]. The superiority of CNN is partly attributed to its ability of automatic feature exaction by learning from the large training dataset. However, the CNN suffers from the heavy computation in both training and testing, and so, is usually implemented using graphics PU (GPU) for parallel computation. This hinders its application in processing huge amount of images on the web.

Our proposed method for fast genre classification of images uses both global visual features and local texture features which consume low computation complexity and is of moderate dimensionality. The local texture features, extracted from different types of image patches and represented in the BoW framework [17, 18] are shown to be effective in differentiating photographs versus non-photograph and scanned versus CPD.

3 Proposed method

3.1 System overview

Fig. 1 shows a schematic diagram of our hierarchical classification system. The first stage extracts global features and uses an SVM for preliminary classification. In this stage, images with high confidence (over a threshold T_c) are made a decision of class directly. While the images with lower confidence are fed into the second stage, which extracts local texture features represented in BoW framework and uses another SVM for final classification. In the second stage, different types of texture descriptors are extracted from local patches and each of them is represented into a BoW histogram. In particular, we carefully design four types of local patches such as edge patch, key point patch, smooth region patch and random patch. This design is aimed to balance the computational complexity and classification accuracy for the second classifier, as the extraction of local features is much more computationally demanding than that of global features. Given a set of local features of a certain type, a two-step clustering method is adopted to generate a discriminative codebook, which is used in the following BoW framework. Finally, we concatenate four BoW histograms into the local feature vector. Depending on the usage of local features, two fusion strategies are proposed to train the second classifier and generate final prediction result.

3.1.1 Training the second classifier with global and local features: The first fusion strategy uses global and local features together to train the second classifier. Considering that global visual features alone are still not very discriminative for those 'difficult' images, which are assigned low confidence by the first classifier, we train the second classifier using both global and local features. Since the local texture features are more suitable for representing image details such as patterns of colour transitions and properties of edges, they can effectively compensate for the deficiency of their global counterparts. In particular, for each image sample, we concatenate its global visual feature calculated previously in the first stage and the new BoW features into a final feature vector and use it to train the second SVM classifier. In testing, the second classifier gives the final classification.

3.1.2 Training the second classifier with local features only: In the second fusion strategy, we take advantage of ensemble learning. Ensemble methods use multiple learning algorithms (classifiers) to obtain better predictive performance than the constituent classifiers alone. For our problem, we can train our two SVM models in different feature spaces, namely global and local features, respectively, and improve final classification performance by combining the predictions of two classifiers. Herein, we use the global features to train the first classifier, and the local features to train the second classifier. In testing, for those images that cannot be labelled with high confidence by the first classifier, local features are extracted and fed into the second classifier. After that, we fuse the predictions of two classifiers by a weighted combination of posterior probabilities to make the final decision of image class.

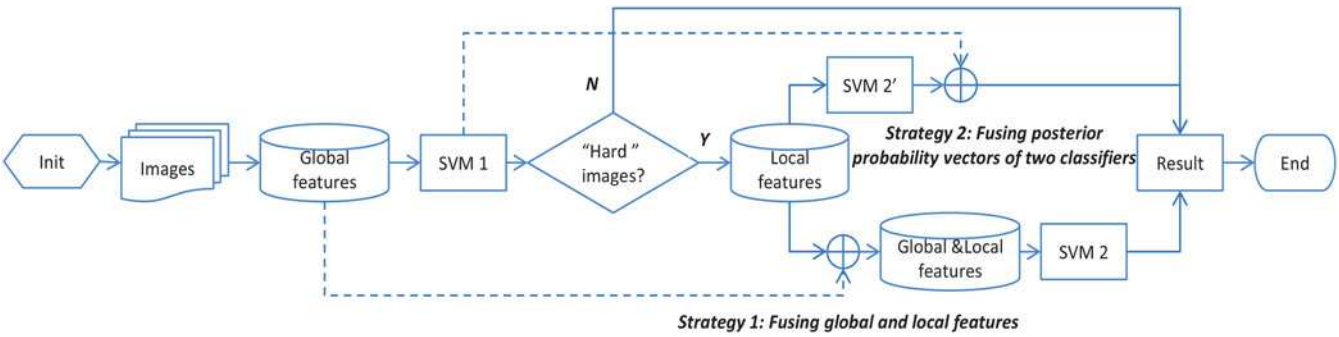

Fig. 1 *Flowchart depicting the proposed hierarchical classification framework. If the max posterior probability from SVM #1 is higher than a given threshold T_c, the image is considered 'simple' and can be made a decision with high confidence. Otherwise, fusion strategies will be introduced in the second stage to find out the final result*

CAAI Trans. Intell. Technol., 2018, Vol. 3, Iss. 3, pp. 161–168

3.2 Global features

For our first-stage classification, the global features are extracted based on the different appearances between four types of images. Compared to NSIs, BDIs tend to have fewer colours, shaper edges, larger constant colour regions and more highly saturated pixels. As for the other two types, SPDs are clearly more uniform in intensity and less distortion than CPDs. We carefully designed our global features so that the differences mentioned above could be easily captured. Meanwhile, it is also necessary to consider the computational complexity of each type of global feature. Computationally intensive feature extraction methods such as scale invariant feature transform (SIFT) or wavelet transform may be more discriminant and give higher classification performance but they also consume more CPU time and memory. By contrast, our procedures of feature extraction only involve the first-order gradient computation and some basic image processing techniques such as thresholding, colour space conversion [from red, green and blue (RGB) to hue, saturation and value (HSV)] and binary erosion.

3.2.1 Coherence of highly saturated pixels f_1: This feature is aimed to measure different patterns of colour transitions from pixel to pixel appearing in four types of images. NSIs often depict objects of the real world and have rarely regions of uniform colour or coherent pixels of highly saturated because of the natural texture of objects, noise and diversity of illumination conditions. On the other hand, BDIs tend to have larger regions of constant colour and more blocks consisting of highly saturated pixels. Let I_{rgb}, I_{hsv} and I_{s} denote a 3-channel RGB image, its HSV version and saturation channel, respectively. A binary image I_{mask1} is obtained by thresholding I_{s} with a given threshold T_{s}. A morphological erosion operation is then performed on I_{mask1} with a 3×3 square structuring element to generate a new I_{mask2}. The number of non-zero pixels in I_{mask1} and I_{mask2} are calculated and denoted as N_1 and N_2. Finally, we define $f_1 = N_2/N_1$. To demonstrate the effectiveness of this measure visually, we randomly selected 3000 images from NSI, BDI and CPD categories in our database: 1000 samples per class, and calculated the normalised histogram of three types of images over f_1. From Fig. 2*a*, we can observe that BDIs which have more coherent and highly saturated regions tend to have higher scores than NSIs.

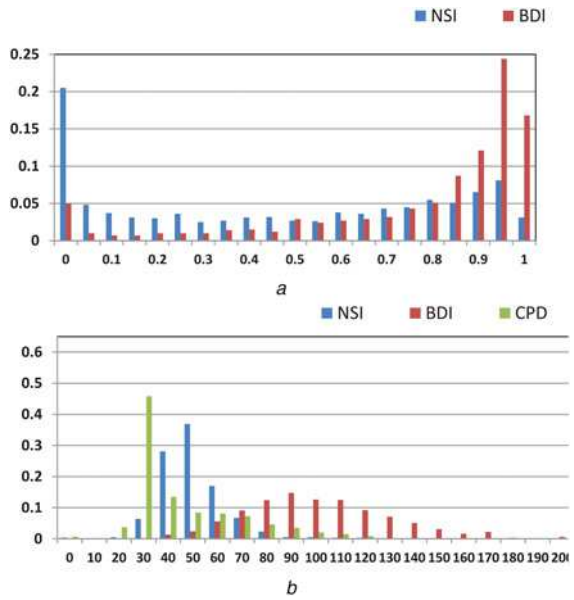

Fig. 2 *Distribution of normalised histograms of different types of images over global features.*

a and
b represent coherence of highly saturated pixels f_1 and average contrast of edge pixels f_2, respectively

3.2.2 Average contrast of edge pixels f_2: The second global feature focuses on the intensity transition between edge pixels in images, which also reflects different patterns between NSIs, BDIs, CPDs and SPDs. For example, edges in NSIs and CPDs are usually generated by occlusion, illumination and changing of surface property, while BDIs tend to have more 'colour edges' [3] resulting from adjacent uniform regions. Accordingly, sharp transitions occur more frequently in BDIs than others. Let I_{g} and M_{c} denote a grey-scale image and its Canny edge [19] map, respectively. We define the max sharpness map M_{ms}

$$M_{\text{ms}}(x, y) = \begin{cases} \max\{|I_{\text{g}}(x, y) - I_{\text{g}}(x', y')|\} & \text{if } M_{\text{c}}(x, y) > 0, \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where current pixel (x, y) and its neighbour (x', y') satisfy $\max\{|x - x'|, |y - y'|\} = D$. In our experiments, D is set to 2. Then, f_2 can be obtained by calculating the average value of M_{ms} with M_{c} as the mask. We also calculated the normalised histogram of three types of images over f_2. As expected, we can observe that BDIs tend to have sharper edges than others, as shown in Fig. 2*b*.

3.2.3 Coherence of smooth region f_3: This feature measures the spatial correlation of pixels of uniform regions in images. The large flat regions in BDIs and SPDs often have high coherence and low gradient magnitude. At first, we generate horizontal and vertical gradient maps M_{gx} and M_{gy} with the kernel $[-1\,0\,1]$ and $[-1\,0\,1]^{\text{T}}$, respectively, then an approximate gradient magnitude image M_{g} is obtained: $M_{\text{g}} = |M_{gx}| + |M_{\text{gy}}|$. Given M_{g}, a binary mask I_{mask3} indicating smooth regions and its eroded version I_{mask4} is generated with a threshold T_{g} in the same way described in Section 3.2.1. Finally, we define $f_3 = \{N_4/N_3, N_3/N_{\text{p}}\}$, where N_3 and N_4 denote the number of non-zero pixels in I_{mask3} and I_{mask4}, respectively, and N_{p} is the total number of image pixels.

3.2.4 Colour histogram f_4: This feature is designed based on the assumption that certain colours occur more frequently in a certain type of images. For example, BDIs come from business websites as ad images tend to be filled with highly saturated red or yellow blocks to grab people's attention. By contrast, most SPDs and CPDs are relatively monochrome and their colour boxes mainly consist of colours of papers and notes, which have very limited fashions. The colour histogram should be effective to represent this special characteristic. Instead of directly calculating the histogram in original RGB colour space, we here choose hue channel I_{h} for speed in practise. The dimensionality of the histogram vector in our implementation is 180 and the histograms are normalised to 0–1 range. We also provide a scatter plot in Fig. 3*a* to visualise the discriminability of this feature. Similarly, 100 images per class are selected randomly and grouped together as a small visualisation dataset. Considering the high dimensionality of this feature, we adopt a dimensionality reduction algorithm, the *t*-stochastic neighbor embedding (SNE) algorithm [20], to map the feature vectors from 180-dimensional (180D) to 2D. From Fig. 3*a*, we can see that most of the SPDs and CPDs points are clustered and form two very distinguishable curves on the reduced map. However, it is also worth noting that there are large overlaps between the BDIs and NSIs points, which means that the colour histogram feature alone cannot differentiate between BDIs and NSIs. Fortunately, features f_1 and f_2 calculated above complement very well. For the large overlaps, one reasonable explanation is that the BDIs samples contain certain small NSI patches, which we will discuss in Section 4.

3.2.5 Gradient magnitude histogram f_5: The distribution of gradient magnitude values also reflects the style of images. We calculated an equal interval histogram of M_{g} as f_5. The gradient value, in a range of [0, 510], is quantised into 200 bins. We also show a scatter plot of gradient magnitude histogram by dimensionality reduction in 2D in Fig. 3*b*.

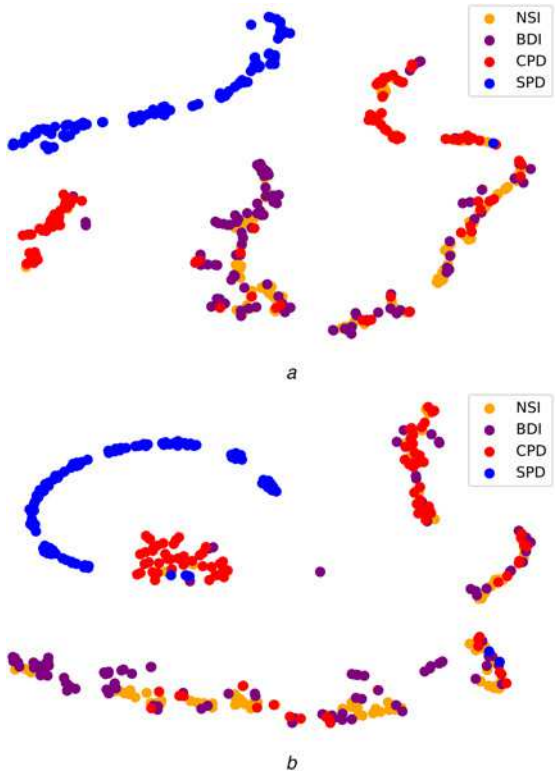

Fig. 3 *Scatter plots of four types of images in the reduced space based on different global features.*

a,b Colour histogram f_4 and gradient magnitude histogram f_5, respectively

3.3 Local features and BoW coding

Although capturing the different characteristics of appearance in common Web images successfully, global features proposed in the above section are not sufficient to discriminate 'difficult' images. While global features are aimed to quickly classify relatively 'easy' images, local features are aimed to discriminate the difficult images at costs of higher computation. We introduce local texture features based on the observation that different types of images show distinct local texture patterns, e.g. BDIs and SPDs often have large constant regions, and CPDs show different texture patterns from SPDs due to the non-uniform illumination in photographing. In addition, some objects possessing certain typical texture patterns such as sky, trees or walls occur frequently in NSIs. To extract local texture features, we adopt local feature aggregation methods [21, 22], which have been widely used for image classification or retrieval in recent years. We exploit four types of local patches and organise their corresponding descriptors in BoW framework [17, 18], which represents an image as a histogram of certain key descriptors and has been demonstrated very effective in image categorisation tasks. For computational simplicity and efficiency, the local patch types we adopt in this paper are edge patch, key point patch, smooth region patch and random patch. The details of different types of patches and feature vectors construction are as follows.

3.3.1 Local patches and descriptors: Four types of local patches are designed in this paper, i.e. edge patch, key point patch, smooth region patch and random patch. The local binary pattern [23] descriptors are used for the first three types of patches, and reduced colour index histogram for the last. The number of each type of patch we sampled from each test image is N_{lp} and all patches have the same size: $S_{\text{lp}} \times S_{\text{lp}}$.

Edge patch: Inspired by the concept of 'intensity edge' and 'colour edge' [3], we randomly select N_{lp} local patches whose centres are exactly located at Canny edge point and then build an edge patch collection for each image. Combined with the BoW framework, the differences of texture in the vicinity of an edge between four types of images are reflected in the local features.

Key point patch: Key point detectors and descriptors have been widely used in image analysis and categorisation. Considering that certain specific objects occur frequently in particular types of images, extracting key pointers can be useful for image genre classification. We adopt the features from accelerated segment test (FAST) corner detection algorithm [24] to locate key points for speeded processing. Moreover similarly, N_{lp} key points are randomly selected as the centres of corresponding patches.

Smooth region patch: Another distinctive texture comes from smooth regions, e.g. sky, lawn and water surface in NSIs, constant colour regions in BDIs and SPDs. Pixels coming from these regions usually have a low gradient magnitude in images. Therefore, we randomly select patches that have high overlap area with I_{mask3}. To make sure smooth pixels are able to occupy sufficient areas in the patches, the overlap ratio threshold is set as 0.7.

Random patch: As the name suggests, patches of this type are cropped randomly from the image and mostly play a complementary role to other types of patches. We use the histogram of reduced colour index map of the raw image to describe these types of local regions for speed. Given the original 256^3 colour space, a uniform quantisation is performed and generates a 64-level (4^3) one: each axis is divided into four equal-sized segments. We then convert the quantised 3-channel image to a 1-channel colour index map by replacing the original triple value (r, g, b) with $r \times 4^2 + g \times 4^1 + b \times 4^0$ pixel by pixel. Finally, a 64D histogram based on the reduced colour index map is calculated and used as random patches' descriptors. The reduced colour index histogram proposed in this paper, despite its simplicity, is efficient for image patch representation, in respect of the relatively lower cost and complexity of applying the reduced colour index histograms as local descriptors.

3.3.2 Concatenated BoW representation: After local feature extraction, each image is abstracted by several local descriptor vectors. Since the traditional 'hard' coding methods in BoW framework fails in capturing spatial layout of descriptors of local patches, we herein adopt the locality-constrained linear coding (LLC) [18] algorithm to organise local descriptors. An approximate version is used for speed to incorporate locality constraint by reconstructing each descriptor with a few closest K entries in the codebook. All the reconstructing vectors are then averaged to generate a final histogram vector. To achieve a more discriminative codebook, we also adopt a two-step clustering method: at first, for each image in training set, N_{c1} sub-centres are selected with the K-means clustering algorithm, and all the sub-centres are gathered and then clustered again to generate a codebook containing N_{c2} entries. Finally, we build codebooks for each type of patches, generate corresponding histogram vectors with LLC coding and concatenate them into a $4N_{\text{c2}}$D vector as the final local feature representation.

4 Database

To validate the effectiveness of the proposed method, we have built a large database of four types of images, i.e. NSI, BDI, CPD and SPD. Depending on the degree of difficulty of labelling images, we divide our database into two sets: the single-label (SL) and the double-label (DL). The first set consists of such images that are easily classified by their appearances and tagged with only one label. Roughly more than 90% images in our database belong to the SL. However, there are a small fraction of 'complex' samples such as photorealistic images produced by cutting edge computer graphics effects, images spliced or embedded by other different types of smaller ones, computer graphics that are displayed on liquid-crystal display monitors and then recaptured by a camera, and so on. With such confusing appearances, they cannot be classified clearly into one type. Hence, for these images, we carefully selected two proper labels as their ground truth labels in order to describe them as accurately as possible. Given the complexity and variety of images on the web, the DL of 3693

CAAI Trans. Intell. Technol., 2018, Vol. 3, Iss. 3, pp. 161–168

Table 1 Details of our image database

	SL	DL				Remarks
		NSI	BDI	CPD	SPD	
NSI	26,410	—	—	—	—	SUN397:5175
BDI	12,153	1792	—	—	—	—
CPD	6805	1484	282	—	—	—
SPD	6124	12	72	51	—	MHW:6036
total	51,492	3,693				55,185

There are 5175 NSIs in our database coming from the public database SUN397, and most of the SPDs (6036) used here come from the multilingual HW dataset.

Fig. 4 *Some SL images in the SL dataset*

Fig. 5 *Some DL images in DL dataset*

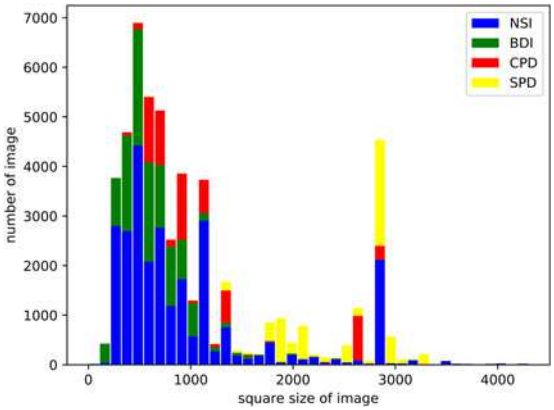

Fig. 6 *Stacked histograms of square sizes of four types of images in our database*

images is a salutary supplement to the SL database and makes it more proper and scalable. Totally, we collected 55,185 images from various sources such as web crawling, manually camera capturing and other public databases including SUN397 [25] and the multilingual hand written (HW) dataset [26]. More details regarding the distribution of different types of images are listed in Table 1 and some samples are shown in Figs. 4 and 5. In addition, we also calculate the distributions of image square size for each type of images, as shown in Fig. 6.

5 Experimental results and discussion

In this section, we first describe the experimental setting and implementation details including the selection of classification models and feature parameters. Then we present our experimental results for the proposed method with global and local features and compared the proposed method with the popular CNN models.

5.1 Experimental setup

In our experiments, we adopt the radial basis function (RBF) kernel SVM as our learning algorithm, and all the classification experiments were implemented with the library for SVM [27] package. Note that for the first-stage classification, though using a linear SVM can largely improve the classification speed, its accuracy is evidently lower than an RBF-kernel SVM. So, we use RBF-kernel SVM in both stages. Since the global feature vector in the first stage has low dimensionality, the speed of non-linear SVM is still acceptable. As for CNN models, we implemented all the models using the Pytorch [28] platform, which is a popular Python package and widely used by researchers in the field of deep learning in recent years. The maximum image size allowed by the system is 1000×1000 for processing speed. If the original image is larger than that, a 1000×1000 sub-region will be randomly cropped and then alternatively tested. About 70% images from each class are selected randomly for training classifiers, and the rest is used for testing. Although our hierarchical classification algorithm involves several parameters, the ranges of their values are relatively broad. For convenience, we divide all key parameters into two parts: feature parameters and system parameters. The first set contains parameters related to feature extraction such as T_s, T_g, N_{lp}, S_{lp}, N_{c1} and N_{c2}. Moreover, the second consists of the confidence threshold T_c (default 0.95) and weighting coefficient w (default 0.80) of local feature classifier in the second fusion strategy. We observed in experiments that the feature parameters influence the final results only slightly. Thus, we only present herein their ranges: $T_s \in [150, 230]$, $T_g \in [0.2, 2.5]$, $N_{lp} \in [100, 300]$, $S_{lp} \in [5, 30]$, $N_{c1} \in [5, 20]$ and $N_{c2} \in [50, 200]$. As for the two system parameters, we will give a detailed analysis in the following section.

Table 2 Results of proposed features on SL dataset (fps: abbreviation of 'frames per second')

Features	ACC, %	Speed, fps
global	93.97	28–30
global + local	98.56	16–18

5.2 Experimental results and discussion

5.2.1 Effectiveness of proposed features: We first extract global and local features from images of SL and use an SVM to train and test them directly. Results are reported in Table 2. As we expected, both global and local features are discriminative for different types of web images. Our *ad hoc* global features can achieve 93.97% classification accuracy at the speed of 28 fps. Furthermore, compared to using global features alone, introducing local texture features evidently increases the final classification accuracy by around 5% but at the sacrifice of processing speed. Such results also justify the rationality of our hierarchical classification algorithm, which quickly filters most 'simple' images using global features and extracts time-consuming local features only for those small number of 'difficult' images (DL) to achieve further fine classification.

5.2.2 Performance of the proposed hierarchical classification method: As we know, deep learning models, especially deep CNNs have achieved a huge success in many computer vision tasks. To further validate the effectiveness of our classification method, we also compared our method with several most popular CNN models such as AlexNet [11], VGGNet [12], ResNet [13], DenseNet [14] and SqueezeNet [15] on both SL and DL datasets. We briefly introduce the key part of each network model and preserve their original structures provided in Pytorch by following the default configuration. All the models we used are pre-trained using the ImageNet [29] database, and we then use a transfer learning strategy to fine-tune the models (number of output nodes changed) on our training dataset.

As mentioned above, there are five types of CNN models adopted in our comparison experiments. The first is AlexNet [11], which is one of the classic CNN architectures for image classification. It first demonstrated superior performance on the large-scale ImageNet task [29].Typical AlexNet consists of five convolutional layers, five max-pooling layers and three fully connected layers. Some regularisation techniques such as dropout and batch normalisation are also used for reducing overfitting and accelerating model training. The second network model is VGGNet [12], which is a family of neural networks sharing the same three-layer fully connected classifier. We perform experiments on VGG-11, VGG-13, VGG-16 and VGG-19, which differ from each other only on the numbers of maps of convolutional feature extractor. The third model is ResNet [13], which has shortcut connections between different layers so as to better fit the difference between input and expected output (residual) other than fitting the output directly. The submodule that fits the residual between expected output and input is named block. Configurations of blocks are either two convolutional layers with size 3×3, stride = 1 and padding = 1 followed by batch normalisation or three convolutional layers with stride = 1 and padding = 1. All ResNets start with a convolutional layer with filter size 7×7, stride = 2 and padding = 3 followed by a batch normalisation layer, and the last layer is always a softmax classifier. The DenseNet [14] has direction connections between each layer with all the layers before it. This makes it able to alleviate the vanishing-gradient problem, strengthen feature propagation, encourage feature reuse and substantially reduce the number of parameters. We experiment its four variants named DenseNet-*x*, *x* here denotes the depth of the models. The last one we chose is a small and energy-efficient deep neural network (DNN) named SqueezeNet [15]. With fewer parameters, SqueezeNet can more easily fit into computer memory and can more easily be transmitted over a computer network.

Table 3 shows the experimental results of different models on the SL dataset. We give the classification accuracy, speed, number of parameters and GPU memory usage for each classifier. Compared to direct classification (Table 1), our hierarchical algorithm with two fusion strategies can achieve a comparable accuracy but at much faster speed (comparable to the speed of global feature only). This is because most 'simple' samples have been confidently classified and filtered by the first-stage classifier using global features. As to the comparison with CNN models, it is shown that our proposed method yields comparable accuracy with some typical CNN models such as AlexNet, VGG-11, VGG-13, ResNet-18, ResNet-34 and two SqueezeNet models. Some deeper architectures such as ResNet-152 and DenseNet-201 obtain higher (over 99%) accuracy but their classification speed has fallen to 0.2 fps on CPU, much slower than the proposed method. It is also worth noting that deeper models have to learn much more parameters and occupy much more memory during the training and validation phases. By contrast, our hierarchical classification algorithm achieves a good tradeoff between classification accuracy and processing speed. Although CNN can run very fast on GPUs, this limits its application in occasions that GPU is not available, and GPUs are also much more energy consuming. On the other hand, implementing our proposed global + local feature-based classification on GPU can also obtain a hundred times of speedup.

Table 3 Performance of different models on SL dataset

Methods	ACC, %	Speed, fps (*C*)/(*G*)	Number of Paras (10^6)	GPU memory, MB
AlexNet	98.39	2.46/157.0	43.02	796
VGG-11	98.19	0.15/328.9	128.78	1073
VGG-13	98.24	0.14/306.9	128.86	1073
VGG-16	98.76	0.12/280.68	134.27	1173
VGG-19	98.84	0.08/297.67	139.58	1633
ResNet-18	97.62	1.03/104.63	11.19	507
ResNet-34	97.89	0.56/85.89	21.32	669
ResNet-50	98.90	0.45/84.09	23.58	1485
ResNet-101	98.94	0.29/83.25	42.63	1571
ResNet-152	99.17	0.17/80.13	58.33	1641
DenseNet-121	98.86	0.27/344.94	7.07	1335
DenseNet-161	99.24	0.10/270.93	26.73	2007
DenseNet-169	99.05	0.15/339.73	12.69	2045
DenseNet-201	99.20	0.12/308.12	18.38	2131
SqueezeNet-10	98.10	1.96/287.94	0.77	589
SqueezeNet-11	97.05	2.15/239.37	0.72	957
our (strategy 1)	98.43	27.2/—	5.58	—
our (strategy 2)	98.21	28.0/—	3.89	—

Paras number of SVM mostly depends on the number of support vectors [CPU (C): Intel(r) Xeon(r) 2.90 GHz; GPU (G) (batch size: 64): NVIDIA Titan, 12G].

5.2.3 Performance on DL dataset: In testing the images in the DL dataset, when the classification decision is identical to one of the ground truth labels, it is considered as correct. In training, the label of the DL image is randomly selected from the label set. The results are listed in Table 4. In the setting 'SL + DL', the test performance of the proposed method is inferior to those of CNN models ResNet-152 and DenseNet-201 but is comparable with those of other CNN models. While in the setting 'DL→DL', the proposed method performs comparably well with the CNN models. The proposed

Table 4 Results on DL dataset

Methods	ACC (SL + DL), %	ACC (SL→DL), %
AlexNet	95.11	94.20
VGG-19	97.85	96.01
ResNet-152	99.01	95.02
DenseNet-201	98.59	96.17
SqueezeNet-10	95.54	92.15
our (strategy 1)	96.41	96.11
our (strategy 2)	96.01	95.33

'SL + DL' indicates the training set contains both SL images and DL images: 'SL->DL' means that the classifier is trained on SL dataset but tested on DL dataset.

CAAI Trans. Intell. Technol., 2018, Vol. 3, Iss. 3, pp. 161–168

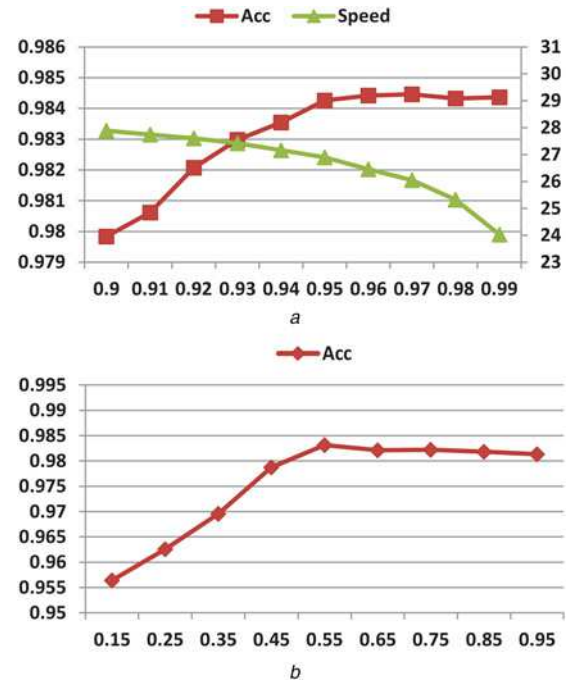

Fig. 7 *Effects of hyperparameters*

a Threshold T_c on classification performance
b Fusion weight w on classification performance

Fig. 8 *Some images misclassified by our classification method*

a NSIs misclassified into BDI
b BDIs misclassified into NSI
c CPDs misclassified into NSI

method has little loss of performance when training without DL images. In contrast, the loss of performance for CNN models from 'SL + DL' to 'SL→DL' is considerable. This is because deep neural networks largely rely on the training set to guarantee the generalisation performance.

5.2.4 Effects of parameters T_c and w on classification performance: Fig. 7 shows the effects of two major hyper-parameters in testing on the SL dataset. In particular, the parameter - T_c controls the number of images sent into the second-stage classifier. When it increases, more images are selected for sending to the second-stage classifier, thus the classification accuracy increases but the speed slows down. As the weighting coefficient of the local feature classifier in the second fusion strategy w is influential to the accuracy of fused classification. It is seen that when w increases from a small value, the accuracy increases gradually and gets saturated at a larger value of w (from 0.5 to 0.95) because the local feature information has played a sufficient role.

5.2.5 Analysis of misclassified images: Fig. 8 shows some images misclassified by our method. We can see there are many confusions between NSIs and BDIs. Specifically, we show examples of three cases: NSIs misclassified into BDI, BDIs misclassified into NSI and CPDs misclassified into NSI. As we can see, most misclassified NSIs in Fig. 8*a* have large flat regions and highly saturated pixels, which violate the previous assumptions we proposed above, thus are categorised into BDI. The BDIs and CPDs containing a large proportion of scene photographs in Figs. 8*b* and *c* are more likely to be classified as NSI. In the future work, we will try to introduce more elaborated features to fix them.

6 Conclusion

In this paper, we proposed a fast two-stage classification method for categorising web images into one of four categories, i.e. NSIs, BDIs, CPDs and SPDs. The first-stage classifier uses global features which have low dimensionality and low computation for guaranteeing high speed. The second-stage classifier extracts local texture features and represents them in BoW framework. Our experimental results show that the proposed method yields high classification accuracy and high speed. Even comparing with the popular CNNs (convolutional networks), the proposed method still provides competitive and its computational speed on CPU is much higher than CNN models. For practical application, a next work is to further differentiate between NSIs with text and NSIs without texts or to detect texts in NSIs.

7 Acknowledgments

This work has been supported by the National Natural Science Foundation of China (NSFC) Grant nos. 61721004 and 61411136002.

8 References

[1] Antonacopoulos, A., Karatzas, D., Lopez, J.O.: 'Accessing textual information embedded in Internet images'. Proc. SPIE Internet Imaging II, San Jose, USA, 2001, pp. 24–26
[2] Liu, Y., Zhang, D., Lu, G., *et al.*: 'A survey of content-based image retrieval with high-level semantics', *Pattern Recognit.*, 2007, **40**, (1), pp. 262–282
[3] Hammoud, R.: 'Color texture signatures for art-paintings vs. scene-photographs based on human visual system'. Proc. 17th ICPR, Cambridge, UK, 2004, vol. 2, pp. 525–528
[4] Ng, T.-T., Chang, S.-F., Hsu, J., *et al.*: 'Physics-motivated features for distinguishing photographic images and computer graphics'. Proc. 13th ACM Multimedia, New York, NY, USA, 2005, pp. 239–248
[5] Athitsos, V., Swain, M.J., Frankel, C.: 'Distinguishing photographs and graphics on the world wide web'. Proc. IEEE Workshop on Content-Based Access of Image and Video Libraries, San Juan, Puerto Rico, 1997, pp. 10–17
[6] Lienhart, R., Hartmann, A.: 'Classifying images on the web automatically', *J. Electron. Imaging*, 2002, **11**, (4), pp. 445–454

[7] Lee, J.H., Baik, S.W., Kim, K., *et al.*: 'IGC: an image genre classification system'. Proc. Third Artificial Intelligence and Computational Intelligence, Taiyuan, China, 2011, pp. 360–367
[8] Sikora, T.: 'The MPEG-7 visual standard for content description – an overview', *IEEE Trans. Circuits Syst. Video Technol.*, 2001, **11**, (6), pp. 696–702
[9] Pourashraf, P., Tomuro, N., Apostolova, E.: 'Genre-based image classification using ensemble learning for online flyers'. Proc. Seventh ICDIP SPIE, Los Angeles, CA, USA, 2015, p. 96310Z
[10] LeCun, Y., Bottou, L., Bengio, Y., *et al.*: 'Gradient-based learning applied to document recognition', *Intell. Signal Process.*, 1998, **86**, (11), pp. 2278–2324
[11] Krizhevsky, A., Sutskever, I., Hinton, G.E.: 'ImageNet classification with deep convolutional neural networks'. Proc. 25th NIPS, USA, 2012, pp. 1097–1105
[12] Simonyan, K., Zisserman, A.: 'Very deep convolutional networks for large-scale image recognition', *CoRR*, 2014, abs/1409.1556
[13] He, K., Zhang, X., Ren, S., *et al.*: 'Deep residual learning for image recognition'. Proc. IEEE Conf. Computer Vision and Pattern Recognition, Las Vegas, USA, 2016, pp. 770–778
[14] Huang, G., Liu, Z., van der Maaten, L., *et al.*: 'Densely connected convolutional networks'. Proc. IEEE Conf. Computer Vision and Pattern Recognition, Honolulu, HI, USA, 2017
[15] Iandola, F.N., Moskewicz, M.W., Ashraf, K., *et al.*: 'SqueezeNet: AlexNet-level accuracy with 50 × fewer parameters and <1 MB model size', *CoRR*, 2016, abs/1602.07360
[16] Jaderberg, M., Vedaldi, A., Zisserman, A.: 'Deep features for text spotting'. Proc. 13th ECCV, Zurich, Switzerland, 2014, pp. 512–528
[17] Csurka, G., Dance, C., Fan, L., *et al.*: 'Visual categorization with bags of key points'. Proc. ECCV Workshop on Statistical Learning in Computer Vision, Czech Republic, 2004, vol. 1, pp. 1–22
[18] Wang, J., Yang, J., Yu, K., *et al.*: 'Locality-constrained linear coding for image classification'. IEEE Proc. CVPR 2010, San Francisco, CA, USA, 2010, pp. 3360–3367
[19] Canny, J.: 'A computational approach to edge detection', *IEEE Trans. Pattern Anal. Mach. Intell.*, 1986, **8**, (6), pp. 679–698
[20] van der Maaten, L., Hinton, G.: 'Visualizing high-dimensional data using *t*-SNE', *J. Mach. Learn. Res.*, 2008, **9**, pp. 2579–2605
[21] Jurie, F., Triggs, B.: 'Creating efficient codebooks for visual recognition'. Proc. Tenth Int. Conf. Computer Vision (ICCV'05), Washington, DC, USA, 2005, pp. 604–610
[22] Nowak, E., Jurie, F., Triggs, B.: 'Sampling strategies for bag-of-features image classification'. Proc. Ninth European Conf. Computer Vision, Berlin, Heidelberg, 2006, pp. 490–503
[23] Ojala, T., Pietikainen, M., Maenpaa, T.: 'Multiresolution gray-scale and rotation invariant texture classification with local binary patterns', *IEEE Trans. Pattern Anal. Mach. Intell.*, 2002, **24**, (7), pp. 971–987
[24] Rosten, E., Drummond, T.: 'Machine learning for high-speed corner detection'. Proc. Ninth ECCV, Berlin, Heidelberg, 2006, pp. 430–443
[25] Xiao, J., Hays, J., Ehinger, K.A., *et al.*: 'Sun database: large scale scene recognition from abbey to zoo'. IEEE Proc. CVPR 2010, San Francisco, CA, USA, 2010, pp. 3485–3492
[26] 'Handwritten language and writer id dataset', University of Maryland, Laboratory for Language and Media Processing (LAMP), 2016. Available at http://lamp.cfar.umd.edu, accessed 2016
[27] Chang, C.-C., Lin, C.-J.: 'LIBSVM: a library for support vector machines', *ACM Trans. Intell. Syst. Technol.*, 2011, **2**, (3), p. 27
[28] Paszke, A., Gross, S., Chintala, S., *et al.*: 'Automatic differentiation in Pytorch', 2017
[29] Deng, J., Dong, W., Socher, R., *et al.*: 'ImageNet: a large-scale hierarchical image database'. IEEE Conf. Computer Vision and Pattern Recognition (CVPR'09), Miami, FL, USA, 2009