Neurocomputing 387 (2020) 161–171

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Recurrent convolutions of binary-constraint Cellular Neural Network for texture recognition

Luping Ji*, Mingzhe Chang, Yulin Shen, Qian Zhang

School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, PR China

ARTICLE INFO

Article history:
Received 1 February 2019
Revised 6 November 2019
Accepted 23 December 2019
Available online 13 January 2020

Communicated by Jun Yu

Keywords:
Cellular Neural Network
Recurrent convolution
Local binary pattern
Texture recognition
Fully-connected classifier

ABSTRACT

Texture recognition is one of the most important branches in image research. This paper mainly aims to develop a new solution to address texture recognition using a Cellular Neural Network (CellNN). Firstly, it proposes an improved model of CellNN by the binary constraints of local receptive fields, and then designs a recurrent convolution framework of such a model to generate two types of texture feature maps, including state feature maps and output feature maps. In order to obtain low-dimensional features, state feature maps are further compressed by the mapping of rotation-invariant patterns and the merging of low-frequency occurrence patterns. By the statistics of joint-distribution patterns, state feature maps and output feature maps are fused together to generate the features of single resolution. Moreover, a multi-resolution feature combination scheme is also designed by the optimization of *softmax* & variance and concatenation of multiple features. Finally, a fully-connected neural network is trained to work as a texture recognizer. The experimental comparisons of totally 15 algorithms on five benchmark datasets show that, on the dataset whose texture-class quantity is not beyond 30, such as Brodatz, our method could always acquire the highest recognition accuracy, outperforming any other compared ones. On the big dataset with huge texture-class quantity, such as ALOT, our method could also surpass any other non-deep-learning one, such as the state-of-the-art gLBP, only slightly falling behind the best two deep-learning ones, FV-Alex and FV-VGGVD. However, in terms of time cost, our method could always outperform any deep-learning one in feature extraction stage, and also surpass any compared one except original LBP in feature matching.

© 2020 Elsevier B.V. All rights reserved.

1. Introduction

Like color and contour, texture is also a class of significant visual characteristics in images. It could often intuitively reflect the change of regional pixel colors, and even the change regularity of color distributions. In general, different objects often have different surface texture patterns. For example, tree bark is usually very different from grass or brick in terms of texture patterns. In view of this, image texture is often regarded as a quite useful reference information to be depended on in visual object recognition, therefore it has attracted much extensive concerns of image research fields. Over the past decades, there have been various texture-based applications to appear, such as the image classification [1], image retrieval [2], face recognition [3], and object detection [4].

Texture recognition, as an important branch of image-based pattern recognition problems [5,6], often contains two necessary sub-tasks. The first task is the model design for feature extraction, and the other is the classifier training for feature recognition. It is generally believed that a good or discriminating feature extraction model, once cooperated with a powerful feature classifier, could often achieve high recognition accuracy. On the contrary, a bad or unreliable model, even cooperated by a powerful classifier, would often damage recognition performance. A large number of research publications have verified that, feature extraction model could be one of the most important and most critical influence factors to further promote the performance of texture recognition [7].

In texture recognition algorithms, feature extraction model is so critical that it has attracted much research enthusiasm from all over the world. A large number of scholars have been addressing this issue in the past decades, and a great deal of research achievements have been acquired in theory and application fields. In proposed models, spatial statistics and spectral transform are often considered to be the two most prominent and most frequently utilized feature extraction paths. Spatial statistic model mainly uses the local texture information in a given pixel neighbor domain. For examples, grey-level co-occurrence matrix [8], structural metrics [9], local binary patterns (LBP) [1,2] and local ternary

* Corresponding author.
E-mail address: jiluping@uestc.edu.cn (L. Ji).

https://doi.org/10.1016/j.neucom.2019.12.119
0925-2312/© 2020 Elsevier B.V. All rights reserved.

patterns (LTP) [10] exactly belong to spatial methods. Correspondingly, some classical feature extraction methods, such as Fourier transform [11,12], wavelet transform [13], Gabor transform [14] and Gaussian derivative models [15], are usually categorized into a spectral feature group. Moreover, the Bag-of-Words model, originating from text analysis, can also be seen in texture recognition, such as the two in [16,17].

Besides the spatial and spectral paths mentioned before, neural networks are also exhibiting great potentials in texture analysis and recognition. Some famous neural networks, such as VeckerNet [18], Multilayer Neural Networks [19–22], Deep Neural Network [23], Deep Autoencoder [24] and Hierarchical Deep Word Embedding model [25], are likely to be successfully applied in texture recognition. In fact, in the past years some neural approaches, which are essentially based on the ideas of deep networks, have been developed for texture recognition. For example, we could often see convolutional neural network (CNN) methods [26,27], the deep filter banks in parallel structure [28] and the deep decomposition of circularly symmetric Gabor-Wavelet (DD-CSGW) [29]. Recently, Basu et al. even systematically analyzes the deep neural networks for texture classification in theoretical level [30].

Inspired by the successes of deep CNNs [26,27], and in order to explore a new path to texture feature representation and recognition, we propose a recurrent convolution framework by the improvement to the traditional cellular neural network (CellNN) originated by Prof. Chua [31]. In our framework, the CellNN is firstly improved by the local binary constraints to the receptive field of network. Compared with traditional version, the information captured by an improved CellNN has changed to the intensity relationships of a central pixel to its receptive-field neighbors, rather than the original pixel intensities of image. Utilizing the recurrent convolutions of improved CellNN on an image, we could always obtain a group of state feature map and output feature map in each recurrence step, and these two types of maps are exactly the important resources to generate features. Moreover, in dimensionality reduction, the feature space of state feature maps will be greatly compressed by rotation-invariant pattern mapping and low-frequency pattern merging. On these compressed maps, optimal map combination needs to be experimentally determined. Finally, a unified texture feature vector is generated by the histogram statistic of joint-distribution patterns on a group of optimized feature maps. In order to increase the robustness to image scale change, it simultaneously extracts multiple feature vectors by different sampling resolutions: radius and points. These temporary vectors are optimized by '*softmax*' and variance, and then they are concatenated together to form the final feature vector of texture image. Moreover, like most deep networks [25,26], we also pre-train a fully-connected neural classifier to execute a recognition via the feature vectors extracted from texture images.

The main motivation we propose an improvement of CellNN to construct a recognition algorithm is explained as follows. CellNN is usually a recurrent network, consisting of an array of neurons, and each neuron of CellNN simultaneously receives external inputs and feedback inputs from neighbors, so as to accumulate dynamic states to generate neuron response [32]. The recurrence steps of CellNN will continue until all neurons converge to the stable status of $\{-1, +1\}$ after finite-time recurrences. Such a type of characteristic is so suitable for image processing that CellNN has been intensively researched and applied, such as in image segmentation [33,34], image classification [35] and image recognition [36,37]. Furthermore, the recurrent convolutions of CellNet could have good potentials in the generation of feature maps. CellNN could simultaneously generate a state feature map and an output feature map in a recurrence time. However, at present how to make full use of these maps to extract low-dimensional and robust features has not yet been paid enough attention to. Because of

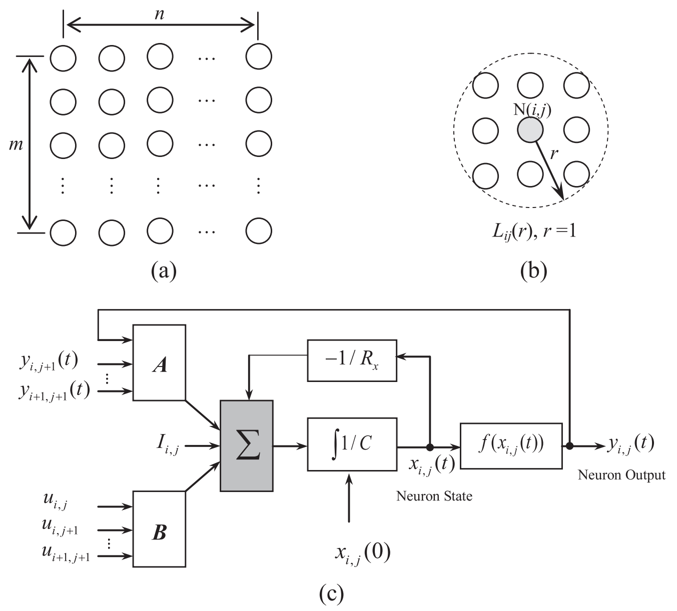


Fig. 1. Conventional CellNN: (a) 2-dimensional structure, (b) locally-connected field and (c) neuron input and output.

this, we consider CellNet as a novel and potential neural-network solution to the feature extraction of textures.

In our CellNN-based texture recognition, feature extractor makes full use of the local-neighbor constraints, recurrent convolutions and neuron outputs. Different from deep convolution neural networks (DCNNs) [30], in improved CellNN, each neuron receives binary external-inputs from neighbors, instead of original pixel gray-scale values. It can conveniently implement the backward reuse of feature information by receiving feedback from receptive-field neurons, rather than only forward reuse. Moreover, it executes recurrent convolutions to generate two separate types of feature maps. It usually compresses feature maps by pattern statistics, rather than by the pooling operation as in deep networks. Besides, its feature dimensionality is generally independent of image size, and multiple features could be further optimized and fused together to promote the robustness of texture features.

To briefly sum up, the primary contributions of this paper include three points. Firstly, it designs an improved CellNN with local-neighbor constraints to capture binary texture feature information. Secondly, it originally proposes a feature extraction framework utilizing the state/output feature maps generated by the recurrent convolutions of improved CellNN. To the best of my knowledge, no other publication has reported these two ideas by now. Finally, in this paper we have designed a set of schemes for feature map compression, fusion and multi-resolution *softmax* optimization, so as to obtain low-dimensional and robust texture feature representation. The biggest characteristic of our CellNet solution is that, it only depends on lower-dimensional features to acquire higher accuracy and faster speed than compared deep-learning algorithms, especially on the datasets with small texture-class quantity.

2. Related work

2.1. Cellular Neural Network (CellNN)

The conventional CellNN, originally proposed by Chua and Yang [31], is a two-dimensional locally-connected network, and it contains a neuron array of m rows $\times$ n columns, see Fig. 1(a). Moreover, any central neuron is only connected with its local neighbor neurons, as shown in Fig. 1(b), and the equivalent block diagram of a continuous-time neuron is briefly exhibited in Fig. 1(c).

In Fig. 1, every small circle represents a neuron, and $'N(i, j)'$ indicates the central neuron at the ith row and jth column in neural network. The symbol r is the locally-connected radius of $N(i, j)$, and $L_r(i, j)$ represents the locally-connected neuron set of $N(i, j)$ with linking radius r. In Fig. 1(c), $x_{i,j}(t)$ and $y_{i,j}(t)$ are the state value of $N(i, j)$ and the output value of $N(i, j)$ at the recurrent time t, respectively. Moreover, the two irrelevant items to time, $u_{i,j}$ and $I_{i,j}$ are the external input to $N(i, j)$ and the fixed input to $N(i, j)$, respectively. Both R_x and C are the constant parameters existing in CellNN. In model, the universal activation function shared by all CellNN neurons is mathematically expressed as $f(.)$. Besides, symbols **A** and **B** are exactly the two convolution kernels corresponding to the feedback inputs (**Y**) and the external inputs (**U**) to neuron $N(i, j)$, respectively.

Just as described in [31], original CellNN is exactly a continuous-time neural network. However, in most image applications, a discrete model could often be more suitable than continuous ones [33,34]. Correspondingly, the discrete state equation of CellNN could be mathematically expressed by

$$x_{i,j}(t+1) = x_{i,j}(t) + \frac{1}{C}\Bigg[-\frac{1}{R_x}x_{i,j}(t) + \sum_{k,l\in L_{i,j}(r)} A_{k,l}y_{k,l}(t) + \sum_{k,l\in L_{i,j}(r)} B_{k,l}u_{k,l} + I_{i,j}\Bigg] \tag{1}$$

where the locally-connected neuron set $L_{i,j}(r) = \left\{C(k,l)|\sqrt{(i-k)^2+(j-l)^2} \le r\right\}$. The constraint conditions for convergence [31,38] include: $|u_{i,j}| \le 1$, $|x_{i,j}(0)| \le 1$, constant $C > 0$ and $R_x > 0$. Coordinates $'k, l'$ represent neuron's row-column location, and they are also the element coordinates in two convolution kernels. Furthermore, the activation function $f(x_{i,j}(t))$ to specially manipulate neuron output is concretely defined as

$$y_{i,j}(t) = f\big(x_{ij}(t)\big) = \frac{1}{2}\left(\left|x_{ij}(t)+1\right| - \left|x_{ij}(t)-1\right|\right) = \begin{cases} 1, & x_{ij}(t) \ge 1; \\ x_{ij}(t), & |\, x_{ij}(t)\,| < 1; \\ -1, & x_{ij}(t) \le -1. \end{cases} \tag{2}$$

Under certain constraint conditions, after finite recurrences all neurons could often generate stable output values, *i.e*, $y_{i,j}(t=t^*) \in \{+1,-1\}$ for any neuron $N(i, j)$. At this time, it implies that CellNN has converged to a stable status [33]. Sometimes, CellNN could work as an image filter, projecting the pixel intensity space of $\{0, 1, \ldots, 255\}^{m\times n}$ to the two-valued pattern space of $\{-1, +1\}^{m\times n}$. It is just this characteristic which makes CellNN receive intensive applications in image processing fields [34–36].

2.2. Local binary pattern

Local binary pattern (LBP) is a classic feature modeling scheme specially for image textures. It was firstly proposed by T. Ojala in 2002 [39]. Fig. 2 briefly exhibits the basic principle of LBP modeling scheme.

In Fig. 2, the small black ball represents the central pixel (labelled by C) to be encoded by LBP rule, and the gray ball P_i, where $i = 0, 1, \ldots, 7$, indicates the pth sampling point that evenly distributes on the circle of radius r exactly around the central pixel C.

In general, the LBP code of every pixel in image is often computed by comparing a given central pixel with its p sampling points. Eq. (3) shows the computational formula to encode a pixel

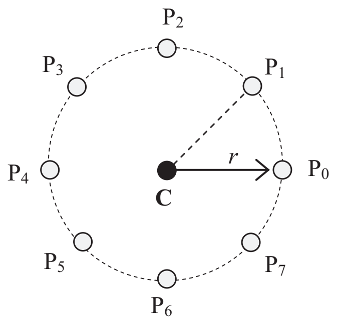


Fig. 2. LBP modeling scheme with 8-neighbor sampling, $(r = 1, p = 8)$.

by p sampling points.

$$LBP_{r,p}(i,j) = \sum_{k=0}^{p-1} s\big(g_k - g(i,j)\big) \times 2^k, s(x) = \begin{cases} 1, & x \ge 0 \\ 0, & x < 0 \end{cases} \tag{3}$$

where (i, j) represent the coordinates of a central pixel. Meanwhile, g_k and $g(i, j)$ represent the equivalent pixel intensity by interpolating on the pth sampling position and the original intensity of central pixel, respectively. Moreover, function $s(.)$ is a binary function whose response is strictly limited to the range of {0, 1}.

By LBP encoding, any original image can be transformed into the LBP code map, in which any code belongs to the pattern range of $\{0, 1, \ldots, 2^p - 1\}$. For example, a map will contain 256 different LBP pattern codes when $p = 8$. Therefore, on LBP map the distribution histogram of 256 patterns could be generated by statistics. The histogram may also be regarded as the feature vector extracted from an original image. However, in order to reduce feature dimensionality, some further map compression schemes, such as the rotation-invariance pattern mapping, named *riu*2 and initially proposed by Ojala et al. [39], need often to be executed on an original LBP map before the extraction of feature histograms begins. From original LBP model, a large number of improved versions have been developed in the past decades, such as completed LBP (CLBP) [40], DLBP [41,42], SSLBP [43], MRELBP [44], BRINT [7], gradient LBP (gLBP) [45] and Median LTP (MLTP) [10]. These variants of LBP have already acquired extensive applications in texture recognition. Specially, Dr. Liu has given a systematic review and finished an experimental study on texture recognition via original LBP, LBP variants and deep-learning methods in [46].

3. Proposed feature extraction model

3.1. Basic framework

In order to obtain discriminating features from original texture images, in this section we propose a feature extraction framework utilizing the CellNN model shown in Fig. 1. Different from traditional CellNNs [31,34,38], in this framework we design a local-constraint rule for the external inputs to the neurons of receptive field. By recurrent convolutions, CellNN produces a series of feature map pairs, and then these maps are transformed into texture feature vector by post-processing. This framework, named CellNet, is briefly shown by Fig. 3.

In Fig. 3, r is a radius of neuron receptive field, identical to the radius r in Fig. 2, and p is the quantity of neighbor neurons in the receptive field of a central neuron. In order to effectively capture the intensity relationship of a pixel and its neighbors, in our framework the local binary constraints, designed specially for neuron's external inputs, *i.e*., the $u_{k,l}$ in Eq. (1), are defined by

$$u_{k,l} = s\big(g_{k,l} - g(i,j)\big) = \begin{cases} 1, & g_{k,l} - g(i,j) \ge \delta \\ 0, & g_{k,l} - g(i,j) < \delta \end{cases} \tag{4}$$

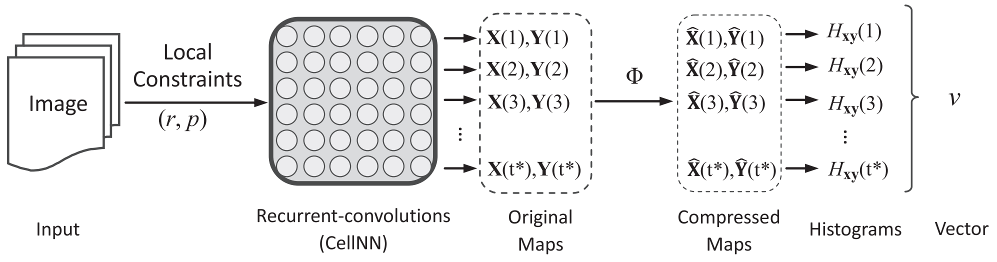


Image
Local Constraints
(r, p)
$\mathbf{X}(1),\mathbf{Y}(1)$
$\mathbf{X}(2),\mathbf{Y}(2)$
$\mathbf{X}(3),\mathbf{Y}(3)$
$\mathbf{X}(t^*),\mathbf{Y}(t^*)$
Φ
$\widehat{\mathbf{X}}(1),\widehat{\mathbf{Y}}(1)$
$\widehat{\mathbf{X}}(2),\widehat{\mathbf{Y}}(2)$
$\widehat{\mathbf{X}}(3),\widehat{\mathbf{Y}}(3)$
$\widehat{\mathbf{X}}(t^*),\widehat{\mathbf{Y}}(t^*)$
$H_{\mathbf{xy}}(1)$
$H_{\mathbf{xy}}(2)$
$H_{\mathbf{xy}}(3)$
$H_{\mathbf{xy}}(t^*)$
v
Input
Recurrent-convolutions (CellNN)
Original Maps
Compressed Maps
Histograms
Vector

Fig. 3. CellNet, a texture feature extraction framework by the recurrent convolutions of CellNN.

where function $s(.)$ is identical to Eq. (3), $'k, l'$ are still the row-column coordinates of neuron, and δ is a turnable threshold. Hereby, it's easy to see that the external input to neuron $N(i, j)$ is no longer original pixel intensity, but the pixel intensity comparison of central neuron $N(i, j)$ with its receptive-field neighbor neuron $N(k, l)$.

As we could see, the two dynamic equations presented by Eqs. (1) and (2) are in scalar forms. They define neuron state recurrence and output modulation rules for CellNN, respectively. For the convenience of description, we need to further rewrite Eqs. (1) and (2) into the matrix forms as follows:

$$\begin{cases} X(t+1) = X(t) + \frac{1}{C}\Big[-\frac{1}{R_x}X(t) + A\otimes Y(t) + B\otimes U + I\Big]; \\ Y(t) = f\big(X(t)\big) = \frac{1}{2}\Big(\big|X(t)+1\big| - \big|X(t)-1\big|\Big); \end{cases} \quad (5)$$

where symbol $'\otimes'$ represents a standard convolution operation between a kernel and an image matrix. The two recurrent convolution kernels of CellNN, A and B have a same kernel size of $(2r+1)\times(2r+1)$. In our CellNet, $X(t)$ and $Y(t)$ are named SFM (State Feature Map) and OFM (Output Feature Map) of neurons at the recurrence time t, respectively. Moreover, in order to ensure the finite recurrence of feature extraction framework, the stability status of CellNN is defined by $|y_{i,j}(t^*)| = 1$ for any neuron $N(i, j)$ at time t^*. It means that only if the output of every neuron reaches to $'+1'$ or $'-1'$ when $t = t^*$, the CellNet framework is deemed to have gone into its stable status (*i.e.*, convergence), therefore it will stop recurrent convolutions immediately.

For simplicity and convenience, in our framework we set parameters $C = R_x = 1$ and $I(i, j) = 0$ for any neuron. Under local sampling condition (r, p), two convolution kernels could often be empirically set as

$$A = \begin{pmatrix} 1 & 2 & 4 \\ 2^{p-1} & -2^p+1 & 8 \\ 64 & 32 & 16 \end{pmatrix} \text{ and } B = \begin{pmatrix} 1 & 2 & 4 \\ 2^{p-1} & 0 & 8 \\ 64 & 32 & 16 \end{pmatrix}$$

Moreover, from the perspective of local patterns [39], kernel B could usually be treated as the position weight matrix of locally-connected neighbor neurons, therefore the edge element of B is usually set to 2^{p-1} $(p = 1, 2, \ldots)$, shown as above. Kernel A is designed for the feedback convolutions of CellNet, and it could be separately optimized by supervised learning. The loss function used to optimize A is: $L(A) = \sum_{q=1}^{Q} \|T_q - Y_q(t*)\|^2$, where Q is the total number of image samples for learning, T_q is the qth objective OFM and $Y_q(t^*)$ is the qth final OFM by the recurrent convolutions of CellNN. Finally, feedback kernel A is optimized by minimizing $L(A)$.

The framework presented in Fig. 3 is initialized, and then it begins recurrent convolutions. At $t = 1$ it could generate the first group of feature maps, $\{X(1), Y(1)\}$. Recurrently, it will also continue to generate more map groups, such as $\{X(2), Y(2)\}$ at $t = 2$, $\{X(3), Y(3)\}$ at $t = 3,\ldots$, and $\{X(t^*), Y(t^*)\}$ at $t = t^*$, where t^* still indicates the recurrence time of convolutions when CellNN converges to a stable status. Moreover, because $X(t)$ and $Y(t)$ are generated by recurrent convolutions, they are likely to fall the real range $\Re^{m\times n}$, too wide to unsuitably extract feature extraction. Therefore, $X(t) \in \Re^{m\times n}$ needs to be normalized to $X(t) \in \{0, 1, \ldots, 2^p - 1\}^{m\times n}$, and similarly $Y(t) \in \Re^{m\times n}$ needs to be normalized to $Y(t) \in \{-1, 0, +1\}^{m\times n}$.

For better intuitiveness, in Fig. 4 we present a numerical example of feature maps using the image sample '007401.*ras*' of dataset Outex_TC_00012. Fig. 4(a) is only a small part of original image, and (b) is the state feature map generated by CellNN at the first recurrence time, t=1. Fig. 4(c) is the normalized map of $X(1)$, in which feature patterns have been normalized to the range {0,...,255}. Fig. 4(d) represents the output feature map generated by CellNN convolutions at the first time $t = 1$, and (e) is the normalized map of $Y(1)$, in which feature patterns have been normalized to {-1,0,1}. Correspondingly, Fig. 4(f), (g) and (h) are the state feature map, the normalized state map and the output feature map of CellNN at the final recurrence, $t = t^* = 9$, respectively.

3.2. Dimensionality compression of feature maps

In deep networks, feature vectors are often obtained by pooling and serializing maps [47], therefore they are usually with very high dimensionality (sometimes even larger than 10^4 [48]), and inevitably accompanied by a large-number requirement for image training samples. Considering small-number samples, and in order to avoid extracting high-dimensional texture feature vectors, in our CellNet framework we have given up the pooling operation intensively adopted in deep networks, and then switched to a different solution path. In our method, it is designed to compress pattern dimensionality as much as possible by the rotation-invariant pattern mapping and low-frequency pattern merging on feature maps. In fact, this solution essentially depends on pattern distribution statistics, and it could take full advantage of the high-level information implied in texture image pixels.

In Fig. 3, the critical pattern mapping from original feature maps to compressed ones, $\Phi : X(t) \mapsto \widehat{X}(t)$, is defined by a rotation-invariant pattern mapping (named Φ_1) and a low-frequency pattern merging (named Φ_2) [39,40]. In our CellNet, the first mapping operation, Φ_1 is specifically defined as

$$\Phi_1 : X(i, j) \mapsto \widehat{X}(i, j) = \min_{q=0,1,\ldots,p-1} \left\{ \sum_{k=0}^{p-1} 2^k \times u_{r,p}(i, j)\big[q, k\big] \right\}, \quad (6)$$

where $u_{r,p}(i, j)$ is the binary external input to central neuron $N(i, j)$, therefore it could be seen as a binary sequence and totally consists of p bits. Moreover in $u_{r,p}(i, j)[q, k]$, q indicates the shift times of circular left-shift operation of the p-bit sequence $u_{r,p}(i, j)$, and meanwhile k represents the bit location from right to left in the binary sequence $u_{r,p}(i, j)$. Therefore, $u_{r,p}(i, j)[q, k]$ just indicates the k^{th} bit after executing q-bit circular left shift on original $u_{r,p}(i, j)$. For example, if $u_{1,8}(i, j) = 11010010$, $u_{1,8}(i, j)[0, 1] = 1$ (corresponding to the bit underlined in $110100\underline{1}0$), and similarly, $u_{1,8}(i, j)[7, 0] = 1$ (corresponding to the bit underlined in $0110100\underline{1}$).

111	111	121	121	115
130	127	129	127	112
134	128	132	128	119
134	122	129	140	141
135	125	126	137	145

(a) original pixels

-123	-96	-96	-126	-62
-49	0.004	-64	-124	-128
-17	0.004	-64	-125	-128
-9	-1	0.004	-121	-126
-12	-13	0.004	-73	-122

(b) feature map $\boldsymbol{X}(t=1)$

9	63	63	3	131
157	255	127	7	0
221	255	127	5	0
237	253	255	13	3
231	229	255	109	11

(c) nomalized $\boldsymbol{X}(t=1)$

-1	-1	-1	-1	-1
-1	0.004	-1	-1	-1
-1	0.004	-1	-1	-1
-1	-1	0.004	-1	-1
-1	-1	0.004	-1	-1

(d) feature map $\boldsymbol{Y}(t=1)$

-1	-1	-1	-1	-1
-1	0	-1	-1	-1
-1	0	-1	-1	-1
-1	-1	0	-1	-1
-1	-1	0	-1	-1

(e) nomalized $\boldsymbol{Y}(t=1)$

-366	-344	-348	-378	-314
-228	470	408	352	348
-244	302	-538	-601	-604
-204	-177	214	351	350
-159	-133	290	-551	-598

(f) feature map $\boldsymbol{X}(t^*=9)$

60	65	64	57	71
90	246	232	220	219
87	209	21	7	7
96	102	189	220	219
106	112	206	18	8

(g) nomalized $\boldsymbol{X}(t^*=9)$

-1	-1	-1	-1	-1
-1	1	1	1	1
-1	1	-1	-1	-1
-1	-1	1	1	1
-1	-1	1	-1	-1

(h) feature map $\boldsymbol{Y}(t^*=9)$

Fig. 4. The numerical explanation for the recurrent convolutions of CellNet with $(r = 1, p = 8)$, on image '007401.*ras*'.

9	63	63	3	131
157	255	127	7	0
221	255	127	5	0
237	253	255	13	3
231	229	255	109	11

(a) nomalized map, $\boldsymbol{X}(t=1)$

9	6	6	2	3
9	8	7	3	0
9	8	7	9	0
9	7	8	9	2
6	9	8	9	9

(b) compressed map, $\widehat{\boldsymbol{X}}(t=1)$

60	65	64	57	71
90	246	232	220	219
87	209	21	7	7
96	102	189	220	219
106	112	206	18	8

(c) nomalized map, $\boldsymbol{X}(t^*=9)$

4	9	1	9	9
9	9	9	9	9
9	9	9	3	3
2	9	9	9	9
9	3	9	9	1

(d) compressed map, $\widehat{\boldsymbol{X}}(t^*=9)$

Fig. 5. The numerical explanation for the feature map compression, on image '007401.*ras*', dataset: *Outex_TC*_00012.

Furthermore, aiming to merge all the low-frequency patterns in feature maps, the second mapping operation, Φ_2 is specifically expressed by

$$\Phi_2 : \widehat{X}(i,j) \mapsto \begin{cases} \widehat{X}(i,j), & \text{if } trans\left[\widehat{X}(i,j)_b\right] \le d; \\ p+1, & \text{otherwise.} \end{cases} \tag{7}$$

where $trans[\widehat{X}(i,j)_b]$ indicates computing the transition times between bit '0' and '1' in binary $\widehat{X}(i,j)_b$. For example, if $\widehat{X}(i,j) = 01110000$, then $trans[01110000] = 2$ (the underlined bits in $\underline{01}1\underline{10}000$). Integer d is an adjustable threshold, and p is still the total number of neighbor neurons (*i.e.*, sampling points).

In essence, the role played by Φ_1 is that it could compress one pixel pattern and its $p-1$ possible rotation variants into only one shared pattern. For example, $X(i,j) =' 3'$, $X(i,j) =' 6'$ and $X(i,j) =' 12'$ will always be mapped into pattern '3' by Φ_1. Similarly, we could see that Φ_2 always keeps the high-frequency patterns of maps unchanged, and merges all low-frequency patterns into one. Therefore, the total number of patterns in maps $X(t)$ could be significantly reduced by Φ_1 and Φ_2 at the same time, realizing $X(t) \in \{0 \sim 2^p - 1\}^{m\times n} \mapsto \widehat{X}(t) \in \{0 \sim p+1\}^{m\times n}$, where p is exactly the total number of sampling points. Obviously, it always has $p+1 \ll 2^p$ for any $p \ge 8$, so our method could greatly compress feature maps from dimensionality 2^p to $p+2$ [39,45,46]. This map compression makes it possible to obtain low-dimensional features. Our simulation also shows that any state feature map $X(t) \in \{0, ..., 255\}^{m \times n}$ could be compressed into $\widehat{X}(t) \in \{0, ..., 9\}^{m\times n}$, reduced almost by 96%.

In order to show the idea of dimensionality compression very intuitively, a part of numerical results on a real texture image are also given in Fig. 5. In the figure, (a) is the normalized state feature map $X(1)$ at $t = 1$, the same as Fig. 4(c), and (b) is the compressed state feature map of $X(1)$ by the pattern mapping rules: both Φ_1 and Φ_2. Correspondingly, Fig. 5(c) and (d) are the normalized state map $X(9)$ at the final recurrence $t^* = 9$ and the compressed map of $X(9)$ also by both Φ_1 and Φ_2, respectively.

3.3. Joint-distribution feature pattern fusion

In the CellNet of Fig. 3, the final feature vector v, is not separately extracted from compressed $\widehat{X}(t)$ and $\widehat{Y}(t)$. In order to generate robust features, we compute the joint-distribution frequency of feature patterns on the map group of $\{\widehat{X}, \widehat{Y}\}$, and then distribution histogram is taken as the feature vector, v of objective texture.

Since $\widehat{X}(t) \in \{0 \sim p+1\}^{m\times n}$ and $\widehat{Y}(t) \in \{-1, 0, +1\}^{m\times n}$, we define $\widehat{x} \in \{0 \sim p+1\}$ and $\widehat{y} \in \{-1, 0, +1\}$, where $\widehat{x}$ and $\widehat{y}$ represent the matrix elements of maps $\widehat{X}$ and $\widehat{Y}$, respectively. Therefore, the joint-distribution frequency on two compressed feature maps $\widehat{X}(t)$ and $\widehat{Y}(t)$ could be computed by

$$H(\widehat{x}, \widehat{y}) = \sum_{i=0}^{m} \sum_{j=0}^{n} \left\{ \widehat{X}(i,j) == \widehat{x} \cap \widehat{Y}(i,j) == \widehat{y} \right\}, \tag{8}$$

where '$\cap$' indicates a standard logic AND operator, and '$==$' represents a comparison logic which judges whether its two sides are equal. This comparison logic outputs 1 if its two sides are equal, otherwise it outputs 0.

On the feature map group $\{\widehat{X}(t), \widehat{Y}(t)\}$, where $t = 1, 2, ..., t^*$-1, Eq. (8) could always generate a joint-distribution feature histogram with $3 \times (p+2)$ bins. However when recurrent time $t = t^*$, CellNN has converged to the output range $\{-1, +1\}$, therefore the joint-distribution histogram $H_{xy}(t^*)$, generated from $\{\widehat{X}(t^*), \widehat{Y}(t^*)\}$, only contains $2 \times (p+2)$ bins. In our CellNet framework, it is also permissible to extract a joint-distribution histogram on any map group such as $\{\widehat{X}(t_1), \widehat{Y}(t_2)\}$. The available histograms coming from different feature map groups could even be optimally concatenated together to form a longer feature vector for texture recognition.

4. Multi-resolution recognition method

By the feature extraction framework, CellNet in Fig. 3, a distribution feature could be extracted by pattern statistics from a texture image. In order to promote the scale-invariance of feature model, inspired by the multi-resolution methods [7,39,40], we further design a multi-resolution optimization & concatenation scheme to generate more robust and more discriminative features. Such a scheme and its classier are briefly exhibited in Fig. 6. The primary characteristics and procedures of our recognition profile are described as follows.

Firstly, it experimentally chooses multiple sampling parameter couples (r_k, p_k), where $k = 1, 2, \ldots, K$, for the local receptive-field

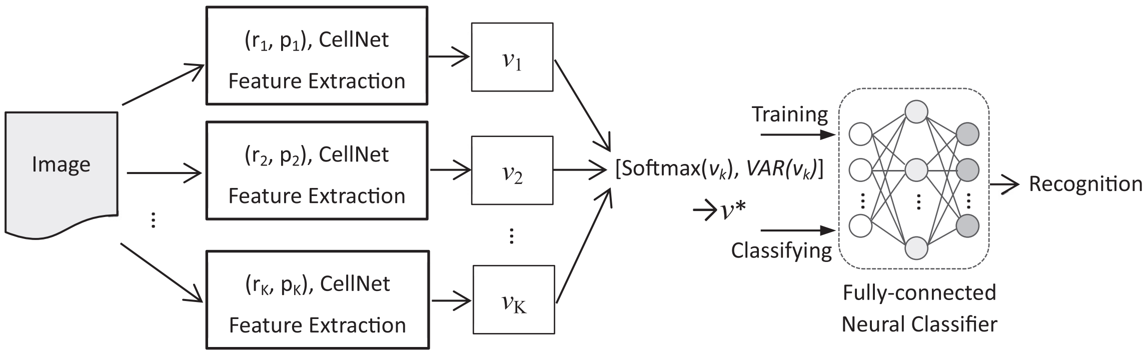


Image
(r_1, p_1), CellNet Feature Extraction
(r_2, p_2), CellNet Feature Extraction
(r_K, p_K), CellNet Feature Extraction
v_1
v_2
v_K
[Softmax(v_k), VAR(v_k)]
→v^*
Training
Classifying
Fully-connected Neural Classifier
Recognition

Fig. 6. The texture recognition scheme by multi-resolution feature optimization and neural classifier.

definition of CellNN, so as to generate k separate feature vectors, $v_1, v_2, \ldots, v_K$. Because the further concatenation of feature vectors will inevitably largen the dimensionality of final texture vector v^*, k usually cannot be a big number. In general, that $K \in \{3, 5, 7\}$ is often appropriate to vector combination [39]. Secondly, an objective texture image is fed into the CellNN-based feature extractor (*i.e.*, CellNet) shown in Fig. 3, to separately extract the K joint-distribution feature vectors (histograms), $v_1, v_2, \ldots, v_K$, which are exactly corresponding to the K couples of sampling parameters, $(r_1, p_1), (r_2, p_2), \ldots, (r_K, p_K)$, respectively.

Thirdly, in Fig. 6, before vector concatenation is done, K feature vectors are separately optimized by the *softmax* function *softmax*(.), which is defined by

$$\widetilde{v}_k = softmax(v_k) = e^{v_k(q)} \Bigg/ \sum_{q=1}^{Q} e^{v_k(q)}, \tag{9}$$

where q is the component index of vector v_k, and Q represents the dimensionality (*i.e.*, vector length) of v_k. Moreover, $VAR(v_k)$ represents the calculation of the standard variance of vector v_k. Therefore, a K-dimensional variance vector v_{var} can be created by vectorizing the K variances of K vectors. In our recognition scheme, the final feature vector of a texture image is generated by concatenating all the $K+1$ vectors together, *i.e.*, $v^* = [v_1, v_2, \ldots, v_K, v_{var}]$. In applications, in order to keep feature dimensionality from growing too much high, not all vectors will be finally used for concatenation. Both too many or too few vectors could be unsuitable for promoting recognition performance. Therefore, in K choosing it needs to reach a tradeoff between dimensionality and performance.

Furthermore, like the usual deep convolutional network in [47], in our scheme we also arrange a fully-connected forward neural network to work as a feature recognizer. Such a recognizer, containing input layer, hidden layer and output layer, receives v^* and then outputs recognition results. As usual, this neural recognizer could be trained by some supervised learning algorithms such as the *Levenberg–Marquardt* (LM) in [49,50]. On objective texture dataset, all the feature vectors $\{v^*\}$, separately extracted from texture samples, are used to train this recognizer, and then the rest of this dataset are taken for testing the recognition performance of our method.

5. Experiments

The primary contribution of this paper includes the improved CellNN with local binary constraints, and the feature extraction and recognition of CellNN-based multi-resolution recurrent convolutions for texture images. In order to evaluate our method and compare it with others, we have conducted a large number of test & comparison experiments on some publicly-available benchmark texture datasets, such as Outex_TC_00012 and Brodatz.

5.1. Experiment platform and datasets

In Section 5.2, our method is simulated by Matlab R2011a on the computer with Pentium Dual-core CPU (3.2 GHz), 4GB RAM and 500GB disk, so as to fulfil the low-cost test of our CellNet method as quickly as possible. However in Section 5.3, our method needs to be compared with some other recognition algorithms, such as the non-deep MRELBP proposed in [44] and the deep FV-AlexNet proposed in [48]. Considering the fairness of experimental comparisons, we have to turn to a high-performance deep learning platform for all these compared algorithms. The important configures of this platform include: Tensorflow v1.23, Intel CORE i7 6850K CPU, two GeForce GTX 1080 GPUs, 32GB RAM and 2TB disk.

In our experiments, we totally uses five public benchmark datasets of texture images. The simple profile introductions to them are presented in Table 1, and more image details are listed as follows.

(1) **Outex_TC_00012**
Outex datasets contain many sub-datasets. In experiments we only choose the most classic one, Outex_TC_00012 (OTC12). This dataset contains 24 texture classes of three illuminations and 9 angles (20 samples per angle). The 480 images of illumination *'inca'* and angle 0, like in some well-konwn publications such as [46], are also chosen for training a neural classifier.

(2) **Brodatz**
There are totally 112 texture images in original Brodatz album. In our experiments, like in the publication [7], we only choose 20 original images of size 640 × 640 to generate extra texture samples. Their image labels are D1, D4, D16, D19, D21, D24, D28, D32, D53, D57, D65, D68, D77, D82, D84, D92, D95, D98, D101 and D102. Eache original image is firstly divided into 16 non-overlapped sub-images, and then every sub-image is ulteriorly rotated by 10 angles: 0, 15, 30, 45, 60, 75, 90, 105, 130 and 145, so as to acquire more texture samples. In training, all the image samples of five angles per class are used to train classifier.

(3) **KTH-TIPS2-b**
KTH-TIPS2-b consists of 11 material texture classes, and each class contains 4 basic images, 3 imaging poses, 4 illuminations and 9 different imaging scales (totally 432 samples per class). In experiments, 4 images and their variants (including 2 poses, 3 illuminations and 6 scales) are taken for training.

(4) **UIUC**
UIUC contains 25 texture classes, and every class consists of 40 images with size 640 × 480. In experiments, every original image is divided into 16 non-overlapped sub-images of

Table 1
The Profile introductions of five benchmark datasets used in experiments.

Texture datasets	Class quantity	Image size	Samples per class	Image quantity	Training per class	Testing samples	Ratio of training to testing
Outex_TC_00012	24	128×128	360	8640	20	8640	5.56%
Brodatz	20	160×160	160	3200	80	1600	50%
KTH-TIPS2-b	11	200×200	432	4752	144	3168	33.33%
UIUC	25	160×120	640	16,000	320	8000	50%
ALOT	250	192×128	400	100,000	200	50,000	50%

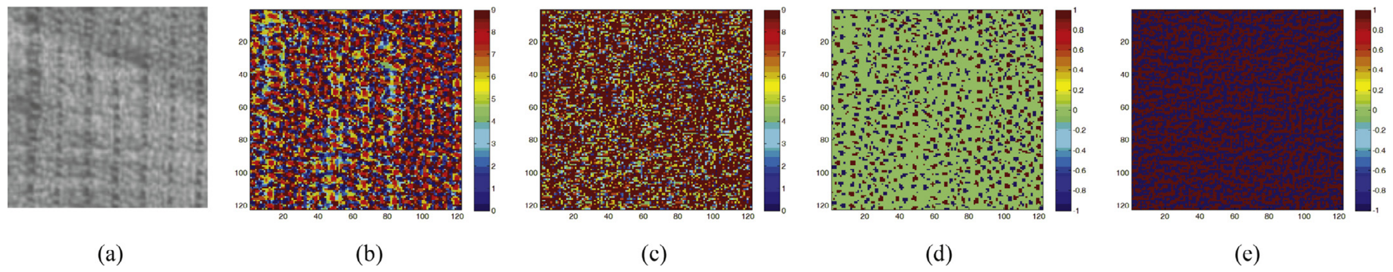


Fig. 7. The recurrent convolutions on sample '000000.ras' by our CellNN with sampling parameters ($r = 3, p = 8$): (a) original texture image, (b) compressed map $\widehat{X}(t = 1)$, (c) compressed map $\widehat{X}(t^* = 5)$, (d) output feature map $Y(t = 1)$ and (e) output feature map $Y(t^* = 5)$.

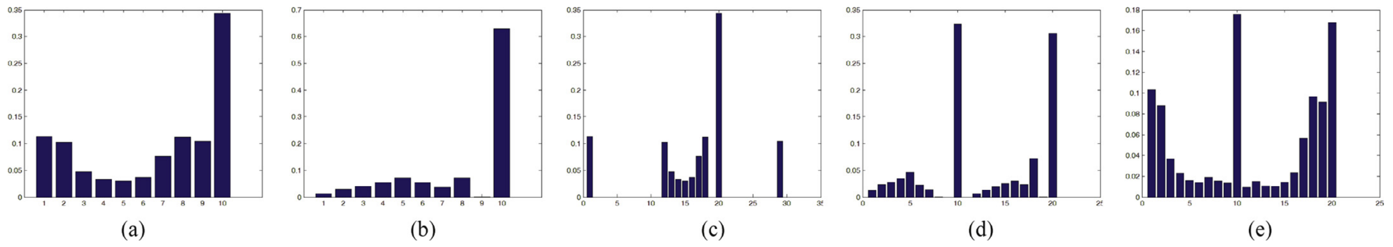


Fig. 8. Feature pattern histograms and three joint-distribution fusions: (a) pattern distribution histogram on $\widehat{X}(1)$, (b) pattern distribution histogram on $\widehat{X}(5)$, (c) pattern fusion on $\{\widehat{X}(1), Y(1)\}$, (d) pattern fusion on $\{\widehat{X}(5), Y(5)\}$ and (e) pattern fusion on $\{\widehat{X}(1), Y(5)\}$.

size 160 × 120 to generate 640 samples. Like the experiment set in publication [46], one half of every class are used to train a feature classifier.

(5) **ALOT**
ALOT is a big dataset, containing 250 classes and 25,000 original images of size 384 × 256 in total. To obtain more samples, we divide every image into 4 sub-images of size 192 × 128. In our experiments, one half of expanded dataset are used to train neural classifier, and the remainder are taken for testing.

5.2. Performance testing

In order to visually show processing details of our method, we have saved a part of important intermediate results in feature extraction stage, and these results are graphically presented in Fig. 7. They mainly include the state feature maps and the output feature maps, which are generated by the recurrent convolutions of CellNN on the random texture image '000000.*ras*' in dataset OTC12.

On sample '000000.*ras*', by the binary constraints of Eq. (4), CellNN obtains original input signals from image, and then it begins to run recurrent convolutions in accordance with Eq. (5). At $t = 1$, the state feature map of CellNN, $X(t = 1)$ is mapped by both Eqs. (6) and (7), so the compressed map $\widehat{X}(1) \in \{0, \ldots, 9\}^{128\times128}$ is acquired. Moreover, by Eq. (5), CellNN releases the output feature map $Y(1) \in \{-1, 0, +1\}^{128\times128}$, and then at $t = 5$, CellNN converges to a stable status, releasing the output feature map $Y(t^* = 5) \in \{-1, +1\}^{128\times128}$. In the meanwhile, the compressed map $\widehat{X}(5) \in \{0, \ldots, 9\}^{128\times128}$ is obtained from original state feature map $X(5)$. For clearance, in Fig. 7, the pixel feature patterns achieved by our method have been represented by different color points.

Fig. 8(a) and (b) exhibits two feature histograms, separately extracted from $\widehat{X}(1)$ and $\widehat{X}(5)$. Fig. 8(c), (d) and (e) are three joint-distribution histograms, fused by Eq. (8) on $\{\widehat{X}(1), Y(1)\}$, $\{\widehat{X}(5), Y(5)\}$ and $\{\widehat{X}(1), Y(5)\}$, respectively. It could be seen that the feature histograms generated by CellNN are low dimensional, only 10 bins in (a) and 20 bins in (e). And in terms of sparsity, the pattern fusion of (e) looks better than the other two, (c) and (d).

In order to numerically present the influences of sampling parameters and map fusions on recognition performance, and determine a relatively-optimal map fusion scheme, we have tested several typical fusions of recurrent map groups by three groups of different sampling parameters. The recognition accuracy comparisons, achieved by our CellNet on five benchmark datasets, are listed in Table 2.

From Table 2, it could be easy to see the three important conclusions: *i*) different (r, p) couples could have different feature dimensionality and recognition performance, *ii*) map fusion could often raise recognition accuracy, and *iii*) the feature map fusion of $\{\widehat{X}(1), Y(t^*)\}$ looks better than the other two, $\{\widehat{X}(1), Y(1)\}$ and $\{\widehat{X}(t^*), Y(t^*)\}$.

Although feature map fusion could inevitably increase the dimensionality of feature vector, the accuracy with fusions could often be greatly raised. For example, the non-fused feature map $\widehat{X}(1)$ of (1,8) sampling only obtains an accuracy of 68.45% on OTC12, however the fused feature map $\{\widehat{X}(1), Y(t^*)\}$ obtains 84.31%, nearly 15.86% higher than the former. Furthermore, we could also see that $\{\widehat{X}(1), Y(t^*)\}$ acquires an accuracy of 78.29% on dataset UIUC when sampling parameters are (3,12), while $\{\widehat{X}(t^*), Y(t^*)\}$ and $\{\widehat{X}(1), Y(t^*)\}$ only obtain 67.32% and 65.23%, respectively. It's obvious that $\{\widehat{X}(1), Y(t^*)\}$ could greatly outperform both $\{\widehat{X}(t^*), Y(t^*)\}$ and $\{\widehat{X}(1), Y(t^*)\}$ under same conditions. One possible explanation

Table 2
The experimental comparisons of different sampling parameters and fusions on five datasets.

Feature fusion groups	(*r*, *p*)	Feature length of v^*	OTC12 (%)	Brodatz (%)	KTH-TIPS2-b (%)	UIUC (%)	ALOT (%)
$\{\widehat{X}(1)\}$	(1,8)	10	68.45	58.82	52.06	61.22	65.39
	(3,12)	14	67.26	58.05	53.44	61.98	64.02
	(5,16)	18	67.14	58.78	52.82	60.25	64.87
$\{\widehat{X}(1)\}, Y(1)\}$	(1,8)	30	84.31	63.92	56.91	66.17	74.55
	(3,12)	42	83.89	63.64	56.26	67.32	72.82
	(5,16)	54	83.58	63.89	56.88	66.45	73.17
$\{\widehat{X}(t^*)\}$	(1,8)	10	66.26	57.73	50.94	59.33	63.03
	(3,12)	14	66.52	56.94	51.28	59.51	62.56
	(5,16)	18	66.55	57.66	51.69	59.05	62.14
$\{\widehat{X}(t^*), Y(t^*)\}$	(1,8)	20	81.08	62.82	54.32	64.18	71.45
	(3,12)	28	80.26	62.74	53.86	65.23	70.36
	(5,16)	36	80.84	62.61	54.23	64.09	71.82
$\{\widehat{X}(1), Y(t^*)\}$	(1,8)	20	89.53	68.52	62.09	79.56	82.49
	(3,12)	28	88.21	69.78	63.34	78.29	82.96
	(5,16)	36	88.52	68.44	63.01	78.82	81.37

Table 3
The performance comparisons of our multi-resolution combination scheme, using the feature map fusion of $\{\widehat{X}(1), Y(t^*)\}$.

Multi-resolution sampling (*r*, *p*) and feature concatenation groups	Feature length of final v^*	OTC12 (%)	Brodatz (%)	KTH-TIPS2-b (%)	UIUC (%)	ALOT (%)
(1,8)	20	89.53	68.52	62.09	79.56	82.49
	21/softmax	89.97	68.94	63.23	80.29	82.98
(1,8)+(2,16)	56	92.69	74.36	68.35	86.82	88.54
	58/softmax	92.95	75.28	68.90	87.46	89.12
(1,8)+(2,16)+(3,24)	108	94.98	86.82	81.74	94.38	92.46
	111/softmax	95.02	87.21	82.01	94.54	92.68
(1,8)+(2,16)+(3,24)+(3,8)	128	95.17	93.54	84.26	95.25	94.04
	132/softmax	95.35	93.82	84.55	95.48	94.21
(1,8)+(2,16)+(3,24)+(3,8)+(5,12)	156	98.26	95.31	87.52	96.07	97.18
	161/softmax	98.42	95.56	87.81	96.25	97.33
(1,8)+(2,16)+(3,24)+(3,8)+(5,12)+(7,16)	192	99.62	95.98	89.35	97.84	98.63
	198/softmax	99.76	96.32	89.49	98.11	98.92

is that $\widehat{X}(1)$ contains the most abundant local pattern features, and $Y(t^*)$ is exactly corresponding to the convergence output of CellNN convolutions.

For a further assessment on multi-resolution recognition performance, we have tested six typical multi-resolution feature optimization and concatenation schemes in all. The recognition results, obtained respectively by six groups of different multi-resolution schemes, are comparatively exhibited in Table 3. It could be clear to see that the feature dimensionality of texture vector v^* will always increase with the total number of concatenated multi-resolution feature vectors. It is only 10 when only the feature vector of $'(r = 1, p = 8)'$ is chosen, however it gradually reaches 192 when the six feature vectors of different sampling resolutions are chosen to concatenate together. Moreover, the recognition accuracy of textures also increases with the total number of multi-resolution feature vectors. For example, on Brodatz our scheme only obtains an accuracy of 74.36% when the two-resolution combination, corresponding to $'(1, 8) + (2, 16)'$, is directly concatenated without a *softmax* and variance optimization, however it obtains 93.54% at a four-resolution concatenation, and 95.98% at a six-resolution concatenation.

By further analysis, it could also be shown that, although multi-resolution scheme is efficient in promoting recognition performance, the increment obtained by different multi-resolution combinations is inhomogeneous. It achieves a big increase of 13.39% from $'(1, 8) + (2, 16)'$ to $'(1, 8) + (2, 16) + (3, 24)'$ on KTH-TIPS2-b. However, it only obtains 2.52% increase from $'(1, 8) + (2, 16) + (3, 24)'$ to $'(1, 8) + (2, 16) + (3, 24) + (3, 8)'$. In addition to the above, Table 3 also corroborates that the optimization measure of feature vectors, *i.e.*, [$softmax(v_k)$, $VAR(v_k)$], is effective to raise recognition performance. For example, for two-resolution group $'(1, 8) + (2, 16)'$, the recognition accuracy, acquired by our scheme without optimization measure, is only 88.54% on ALOT, however it increases to 89.12% when the optimization is carried out before multi-resolution concatenation.

5.3. Comparisons with other methods

In order to highlight the advantages of our method, we have compared our 6-resolution combination scheme with some deep-learning ones, whose names have been marked by **bold font**, such as the state-of-the-art DD-CSGW in [29], as well as some non-deep-learning ones such as gLBP [45] and MLTP [10]. Because of involving deep-learning networks, our experiments have been conducted on the GPU platform introduced at the beginning of Section 5.1. Any other algorithm except ours are implemented by downloading open source codes. Moreover, we have chosen two important comparison metrics to evaluate algorithm performance. One is recognition accuracy, and the other is the average execution time of each algorithm.

The recognition accuracy comparisons of 15 methods on five texture datasets are exhibited in Table 4. In the comparison table, 'CellNet' indicates our scheme without [$softmax(v_k)$, $VAR(v_k)$] optimization, while 'CellNet with softmax' represents the scheme optimized by [$softmax(v_k)$, $VAR(v_k)$]. From this comparison table, it could be easy to see three obvious characteristics in our recognition schemes. Firstly, our two schemes could always extract low-dimensional texture feature vectors. Secondly, our two schemes could always achieve the highest accuracy on OTC12, Brodatz, KTH-TIPS2-b and UIUC, but except on ALOT. Finally, the same as in Table 3, by the optimization of *softmax* and variance, our CellNet method could obtain a further promotion of recognition accuracy.

For examples, the dimensionality of feature vectors by CellNet is only 192 on all datasets, and it only increases to 198 after

Table 4
The experiment comparisons of dimensionality and accuracy, involving 15 methods and five texture datasets.

Compared methods	Length of feature vector	OTC12 (%)	Brodatz (%)	KTH-TIPS2-b (%)	UIUC (%)	A LOT (%)
LBP [39]	102	92.14	91.27	62.69	88.42	94.18
DLBP [42]	14150	91.97	88.37	61.72	83.83	80.52
CLBP [40]	3552	95.78	92.41	64.18	95.79	96.58
BRINT [7]	1296	98.13	90.92	66.67	93.38	96.17
MRELBP [44]	800	99.58	90.98	68.98	94.81	97.29
SSLBP [43]	2400	99.36	89.62	65.57	95.52	96.66
gLBP [45]	1260	99.32	92.70	67.21	95.55	97.87
MLTP [10]	5184	99.48	92.92	68.48	96.97	97.92
ScatNet(PCA) [52]	596	99.06	84.51	68.92	96.18	98.02
RandNet(NNC) [53]	2048	52.45	91.23	60.67	57.06	87.34
FV-AlexNet(SVM) [48]	32768	72.30	95.25	77.90	97.26	99.11
FV-VGGVD(SVM) [51]	65536	82.30	95.72	88.20	97.72	99.48
DD-CSGW(SVM) [26]	1575	98.86	94.16	79.66	92.88	95.62
CellNet	192	99.62	95.98	89.35	97.84	98.63
CellNet with softmax	198	99.76	96.32	89.49	98.11	98.92

softmax and variance optimization. In the meanwhile, the dimensionality by LBP is 102, but the dimensionality by FV-VGGVD has increased to 65536, far higher than by our schemes. The dimensionality by CellNet is always higher than by LBP, because LBP defers to $p+2$ for very sampling resolution, while CellNet complies with $2 \times (p+2)$. Those deep-learning methods, such as FV-AlexNet [48], extensively adopt pooling operation to compress feature maps, so the dimensionality by them is usually very high.

Our CellNet achieves an accuracy of 99.62% on OTC12, outperforming all LBP-based methods and deep-learning methods, such as the MRELBP (99.58%) and the AlexNet (only 72.30%). Similarly, on Brodatz, KTH-TIPS2-b and UIUC, our method also obtains the highest accuracies: 95.98%, 89.35% and 97.84%, respectively. If it is optimized by *softmax* and variance, its accuracy could obtain a further improvement. On ALOT, the recognition accuracy by CellNet is 98.63%, and it increases to 98.92% after optimization, acquiring an increase of 0.31%. However, the highest accuracy is 99.48%, achieved by FV-VGGVD [51], and the second highest is 99.11%, acquired by AlexNet [48]. Such a group of comparisons implies that our schemes cannot always outperform all compared methods on ALOT, and sometimes they fall behind the best two deep-learning methods, FV-VGGVD and AlexNet.

Our methods are always the best on any texture datasets except on ALOT. The possible reasons to explain such a phenomena, could be analyzed as follows. Firstly, ALOT consists of 250 texture classes and 100,000 images, so some deep methods, even containing a large number of parameters, such as FV-VGGVD could still be well trained sufficiently by high-dimensional training features. Secondly, any of the first 4 datasets only contains a small class quantity of less than 30, far smaller than ALOT's quantity 250. Thirdly, the dimensionality of feature vectors by our schemes is far lower than by FV-VGGVD or FV-AlexNet. Usually, on a large class-quantity dataset, it's often more difficult to simultaneously maximize inter-class distance and minimize inner-class distance by low-dimensional features than by high-dimensional ones. ALOT's class quantity is 250, too big to our schemes. Limited by the length of feature vectors, our methods are often hard to precisely recognize the textures with so many classes. As a result, our methods cannot usually outperform FV-AlexNet and FV-VGGVD on ALOT.

Besides recognition accuracy, the average time of each image is also an important evaluation criterion in texture recognition [30,46]. Time consumption determines the total requirement of computation cost, and it also determines whether an algorithm could be applied in real-time systems. Considering these factors, we have also arranged a group of experiments to assess the average time per image of each algorithm.

Table 5 exhibits our experiment results. In this table, the average time of every algorithm is separated into two primary parts. One is the average time in feature extraction (T_E), and the other is the average time in feature matching (T_M). Training time is usually too long, especially in some deep-learning methods such as ScatNet [52], FV-VGGVD [51] and DD-CSGW [29], so it is not considered. In each testing, the start time and end time of each image-batch operation are recorded by TensorFlow time library so as to calculate the duration time of each batch, and total duration time is obtained by adding all image-batch duration time together. Then, the average execution time of each image is calculated by dividing total duration time by image quantity. Finally, the average time of each image separately in the two stages, shown in Table 5, is obtained by repeating each algorithm ten times on testing images.

From Table 5, we could see three typical characteristics about the time cost of compared methods. Firstly, T_E is always much longer than T_M for any method on a same dataset. For example, our CellNet needs 315.5 ms to extract feature vectors on Brodatz, however it needs only 30.4ms to recognize a feature. The former is about 10 times of the latter. Secondly, on every dataset, the T_E by our two schemes, *i.e.*, CellNet and CellNet with softmax, is always much shorter than by any of the 5 deep-learning methods such as RandNet [53], although longer than by any of the 8 non-deep-learning methods such as MLTP and BRINT. For example, on UIUC CellNet's T_E is 267.1 ms, approximate 18ms shorter than FV-AlexNet's 285 ms, and 106 ms longer than MRELBP's 161 ms. Thirdly, the T_M by our two methods is always the shortest on any dataset, except by LBP. For example, on ALOT CellNet's T_M is 35.2 ms, only 7.2 ms longer than LBP's 28 ms, however 4.8 ms shorter than the least one of deep methods, ScatNet's 40 ms. Finally, for either T_E and T_M, the optimization by *softmax* and variance could only raise a slight, even negligible time cost. For example, on KTH-TIPS2-b, CellNet's T_E and T_M are 378.6 ms and 119.3 ms, respectively. However, after optimization, T_E and T_M only increase to 383.3 ms and 119.5 ms, respectively. An increase of about 4.7ms happens on T_E, meanwhile a negligible increase of 0.2 ms occurs on T_M.

Feature extraction involves usually many stages, such as the histogram statistics in non-deep methods and the layer-by-layer convolutions in deep ones, while feature matching is only a forward matrix computation of classifier. Therefore T_E is often much longer than T_M. The feature dimensionality generated by deep learning methods is usually much higher than by our methods. Our methods only involve recurrent convolutions, so they need shorter T_E than any compared deep-learning ones, and longer than any other compared non-deep ones. Moreover, feature matching time is mainly dependent of vector dimensionality. In general, the larger

Table 5
The average execution time per image of 15 methods on five datasets, in millisecond (ms).

Compared methods	Length of feature	OTC12		Brodatz		KTH-TIPS2-b		UIUC		ALOT	
		TE	TM	TE	TM	TE	TM	TE	TM	TE	TM
LBP [39]	102	46	26	58	23	69	98	49	26	54	28
DLBP [42]	14150	193	122	241	115	290	500	205	125	228	129
CLBP [40]	3552	76	81	95	72	114	317	81	86	90	79
BRINT [7]	1296	129	58	161	52	194	203	137	59	152	55
MRELBP [44]	800	152	46	190	41	228	171	161	466	179	50
SSLBP [43]	2400	94	69	118	65	141	274	100	74	111	77
gLBP [45]	1260	95	71	119	66	143	279	101	73	112	74
MLTP [10]	5184	181	101	226	98	272	386	192	112	214	106
ScatNet(PCA) [52]	596	296	38	370	35	444	148	314	42	349	40
RandNet(NNC) [53]	2048	278	64	348	60	417	241	295	67	328	70
FV-AlexNet(SVM) [48]	32768	269	138	336	128	404	510	285	150	317	148
FV-VGGVD(SVM) [51]	65536	328	165	410	160	492	628	348	169	387	172
DD-CSGW (SVM) [29]	1395	364	75	455	77	546	212	386	71	430	81
CellNet	192	252.4	33.1	315.5	30.4	378.6	119.3	267.1	38.5	297.4	35.2
CellNet with softmax	198	253.1	33.2	317.8	30.5	383.3	119.5	269.8	38.6	298.7	35.3

the dimensionality is, the longer the matching time is. Therefore, our T_M is often longer only than LBP's, but much shorter than any other's.

6. Conclusions

This paper proposes the improved version of CellNN with local binary constraints, and then presents an original feature extraction framework utilizing the recurrent convolutions of CellNN. In the framework, the state feature maps, generated by recurrent convolutions, could be efficiently compressed by rotation-invariant mapping and low-frequency merging. The final feature vectors of texture images are formed from the joint-distribution pattern histograms of feature maps. Moreover, an optimization scheme of multi-resolution feature concatenation is put to use so as to promote the robustness of texture features. In our method, a fully-connected neural network is also trained to work as a recognizer. The experiment results on five texture datasets show that our method could always effectively extract low-dimensional texture features. In terms of recognition accuracy, it could always surpass any compared method, including state-of-the-art gLBP on the small datasets with small texture-class quantity, although it often falls slightly behind some deep-learning ones such as FV-AlexNet on the ALOT, a big dataset with texture class quantity beyond 200. Moreover, as for time performance, in feature extraction our method could often surpass any deep-learning algorithm, although it often falls behind any other non-deep algorithm on every dataset. Meanwhile, in terms of matching time, our method could often outperform any other compared one except LBP. Furthermore, it could even achieve a further promotion of accuracy by the optimization of softmax and variance.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (NSFC) under Grant No. 61972072. We would like to thank the Editors and Reviewers for their revision suggestions.

Supplementary material

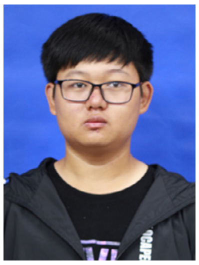
Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.neucom.2019.12.119 .

References

[1] R. Peyret, A. Bouridane, F. Khelifi, M.A. Tahir, S. Al-Maadeed, Automatic classification of colorectal and prostatic histologic tumor images using multiscale multispectral local binary pattern texture features and stacked generalization, Neurocomputing 275 (2018) 83–93.
[2] C. Singh, E. Walia, K.P. Kaur, Color texture description with novel local binary patterns for effective image retrieval, Pattern Recognit. 76 (2018) 50–68.
[3] X. Tan, B. Triggs, Enhanced local texture feature sets for face recognition under difficult lighting conditions, IEEE Trans. Image Process. 19 (6) (2010) 1635–1650.
[4] A. Satpathy, X. Jiang, H.-L. Eng, LBP-based edge-texture features for object recognition, IEEE Trans. Image Process. 23 (5) (2014) 1953–1964.
[5] R.M. Haralick, K. Shanmugam, I. Dinstein, Textural features for image classification, IEEE Trans. Syst. Man Cybern. SMC-3 (6) (1973) 610–621.
[6] T. Ojala, M. Pietikäinen, D. Harwood, A comparative study of texture measures with classification based on feature distributions, Pattern Recognit. 29 (1) (1996) 51–59.
[7] L. Liu, Y. Long, P. Fieguth, S. Lao, G. Zhao, BRINT: binary rotation invariant and noise tolerant texture classification, IEEE Trans. Image Process. 23 (7) (2014) 3071–3084.
[8] A.F.C.I. W. Gomez W. C. A. Pereira, Analysis of co-occurrence texture statistics as a function of gray-level quantization for classifying breast ultrasound, Pattern Recognit. Lett. 31 (10) (2013) 1889–1899.
[9] J. Zujovic, T.N. Pappas, D.L. Neuhoff, Structural texture similarity metrics for image analysis and retrieval, IEEE Trans. Image Process. 22 (7) (2013) 2545–2558.
[10] L. Ji, Y. Ren, X. Pu, G. Liu, Median local ternary patterns optimized with rotation-invariant uniform-three mapping for noisy texture classification, Pattern Recognit. 79 (2018) 387–401.
[11] R. Maani, S. Kalra, Y. Yang, Rotation invariant local frequency descriptors for texture classification, IEEE Trans. Image Process. 22 (6) (2013) 2409–2419.
[12] T. Brandtberg, Virtual hexagonal and multi-scale operator for fuzzy rank order texture classification using one-dimensional generalised Fourier analysis, in: Proceedings of 23rd International Conference on Pattern Recognition (ICPR), 2016, pp. 2018–2024.
[13] E. de Ves, D. Acevedo, A. Ruedin, X. Benavent, A statistical model for magnitudes and angles of wavelet frame coefficients and its application to texture retrieval, Pattern Recognit. 47 (9) (2014) 2925–2939.
[14] C. Li, G. Duan, F. Zhong, Rotation invariant texture retrieval considering the scale dependence of Gabor wavelet, IEEE Trans. Image Process. 24 (8) (2015) 2344–2354.
[15] Y. Song, S. Zhang, B. He, Q. Sha, Y. Shen, T. Yan, R. Nian, A. Lendasse, Gaussian derivative models and ensemble extreme learning machine for texture image classification, Neurocomputing 277 (2018) 53–64.
[16] X. Wang, Y. Wang, X. Yang, H. Zuo, Texture classification based on SIFT features and bag-of-words in compressed domain, in: Proceedings of 5th International Congress on Image and Signal Processing, 2012, pp. 941–945.
[17] B. Umit, S. Abdulkadir, Texture classification using scale invariant feature transform and bag-of-words, in: Proceedings of 23rd Signal Processing and Communications Applications Conference (SIU), 2015, pp. 152–155.
[18] J. Ou, Y. Li, Vector-kernel convolutional neural networks, Neurocomputing 330 (2019) 253–258.

[19] S.P. Adhikari, C. Yang, K. Slot, M. Strzelecki, H. Kim, Hybrid no-propagation learning for multilayer neural networks, Neurocomputing 321 (2018) 28–35.
[20] Z. Kang, X. Shi, et al., Partition level multiview subspace clustering, Neural Networks 122 (2020) 279–288.
[21] Z. Kang, H. Pang, et al., Robust graph learning from noisy data, IEEE Transactions on Cybernetics (2020), doi:10.1109/TCYB.2018.2887094.
[22] Z. Kang, L. Wen, et al., Low-rank kernel learning for graph-based clustering, Knowledge-Based Systems 163 (2019) 510–517.
[23] A. Liu, Y. Laili, Balance gate controlled deep neural network, Neurocomputing 320 (2018) 183–194.
[24] C. Hong, J. Yu, J. Wan, D. Tao, M. Wang, Multimodal deep autoencoder for human pose recovery, IEEE Trans. Image Process. 24 (12) (2015) 5659–5670.
[25] J. Yu, M. Tan, H. Zhang, D. Tao, Y. Rui, Hierarchical deep click feature prediction for fine-grained image recognition, IEEE Trans. Pattern Anal. Mach. Intell. (2019) 1, doi:10.1109/TPAMI.2019.2932058.
[26] V. Andrearczyk, P.F. Whelan, Convolutional neural network on three orthogonal planes for dynamic texture classification, Pattern Recognit. 76 (2018) 36–49.
[27] A. Gömez-Rłos, S. Tabik, J. Luengo, A. Shihavuddin, B. Krawczyk, F. Herrera, Towards highly accurate coral texture images classification using deep convolutional neural networks and data augmentation, Expert Syst. Appl. 118 (2019) 315–328.
[28] A. Shahriari, Learning deep filter banks in parallel for texture recognition, in: Proceedings of IEEE International Conference on Image Processing (ICIP), 2016, pp. 1634–1638.
[29] C. Li, Y. Huang, Deep decomposition of circularly symmetric Gabor wavelet for rotation-invariant texture image classification, in: Proceedings of IEEE International Conference on Image Processing (ICIP), 2017, pp. 2702–2706.
[30] S. Basu, S. Mukhopadhyay, M. Karki, R. DiBiano, S. Ganguly, R. Nemani, S. Gayaka, Deep neural networks for texture classification-a theoretical analysis, Neural Netw. 97 (2018) 173–182.
[31] L.O. Chua, L. Yang, Cellular neural networks: theory, IEEE Trans. Circuits Syst. 35 (10) (1988) 1257–1272.
[32] L. Ji, X. Pu, H. Qu, G. Liu, One-dimensional pairwise CNN for the global alignment of two DNA sequences, Neurocomputing 149 (2015) 505–514.
[33] R. Perfetti, E. Ricci, D. Casali, G. Costantini, Cellular neural networks with virtual template expansion for retinal vessel segmentation, IEEE Trans. Circuits Syst. II: Express Briefs 54 (2) (2007) 141–145.
[34] X. Hu, G. Feng, S. Duan, L. Liu, Multilayer RTD-memristor-based cellular neural networks for color image processing, Neurocomputing 162 (2015) 150–162.
[35] Y.-W. Shou, C.-T. Lin, Image descreening by GA-CNN-based texture classification, IEEE Trans. Circuits and Syst. I: Regul. Pap. 51 (11) (2004) 2287–2299.
[36] M. Milanova, U. Bäker, Object recognition in image sequences with cellular neural networks, Neurocomputing 31 (1) (2000) 125–141.
[37] Q. Gao, G.S. Moschytz, Fingerprint feature matching using CNNs, in: Proceedings of IEEE International Symposium on Circuits and Systems, 3, 2004, pp. II-I–73.
[38] L. Wang, W. Liu, H. Shi, J.M. Zurada, Cellular neural networks with transient chaos, IEEE Trans. Circuits Syst. II Express Briefs 54 (5) (2007) 440–444.
[39] T. Ojala, M. Pietikäinen, T. Maenpaa, Multiresolution gray-scale and rotation invariant texture classification with local binary patterns, IEEE Trans. Pattern Anal. Mach. Intell. 24 (7) (2002) 971–987.
[40] Z. Guo, L. Zhang, D. Zhang, A completed modeling of local binary pattern operator for texture classification, IEEE Trans. Image Process. 19 (6) (2010) 1657–1663.
[41] S. Liao, M.W.K. Law, A.C.S. Chung, Dominant local binary patterns for texture classification, IEEE Trans. Image Process. 18 (5) (2009) 1107–1118.
[42] K.E. Rakesh Mehta, Dominant rotated local binary patterns (DRLBP) for texture classification, Pattern Recognit. Lett. 71 (2016) 16–22.
[43] Z. Guo, X. Wang, J. Zhou, J. You, Robust texture image representation by scale selective local binary patterns, IEEE Trans. Image Process. 25 (2) (2016) 687–699.
[44] L. Liu, S. Lao, P.W. Fieguth, Y. Guo, X. Wang, M. Pietikainen, Median robust extended local binary pattern for texture classification, IEEE Trans. Image Process. 25 (3) (2016) 1368–1381.
[45] L. Ji, Y. Ren, G. Liu, X. Pu, Training-based gradient LBP feature models for multiresolution texture classification, IEEE Trans. Cybern. 48 (9) (2018) 2683–2696.
[46] L. Liu, P. Fieguth, Y. Guo, X. Wang, M. Pietikainen, Local binary features for texture classification: taxonomy and experimental study, Pattern Recognit. 62 (2017) 135–160.
[47] G. Huang, Z. Liu, L. van der Maaten, K.Q. Weinberger, Densely connected convolutional networks, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2261–2269.
[48] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: F. Pereira, C.J.C. Burges, L. Bottou, K.Q. Weinberger (Eds.), Proceedings of Advances in Neural Information Processing Systems 25, Curran Associates, Inc., 2012, pp. 1097–1105.
[49] A. Toledo, M. Pinzolas, J.J. Ibarrola, G. Lera, Improvement of the neighborhood based Levenberg–Marquardt algorithm by local adaptation of the learning coefficient, IEEE Trans. Neural Netw. 16 (4) (2005) 988–992.
[50] X. Fu, S. Li, M. Fairbank, D.C. Wunsch, E. Alonso, Training recurrent neural networks with the Levenberg–Marquardt algorithm for optimal control of a grid-connected converter, IEEE Trans. Neural Netw. Learn. Syst. 26 (9) (2015) 1900–1912.
[51] M. Cimpoi, S. Maji, A. Vedaldi, Deep filter banks for texture recognition and segmentation, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 3828–3836.
[52] J. Bruna, S. Mallat, Invariant scattering convolution networks, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1872–1886.
[53] T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, Y. Ma, PCANet: a simple deep learning baseline for image classification? IEEE Trans. Image Process. 24 (12) (2015) 5017–5032.

Luping Ji received his B.S. degree in Mechanical & Electronic Engineering from Beijing Institute of Technology, Beijing, China, in 1999. Then, he received his M.S. (2005) and Ph.D (2008) degrees from school of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China. Currently, he works as an associate professor in school of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China. His current research interests include Neural Networks and Pattern Recognition.



Mingzhe Chang is an undergraduate student in School of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC), Chengdu, China, since 2017. Currently, he is studying for his B.S. degree (No. 2017060105011), supervised by Dr. Luping Ji. His research interests include cellular neural networks, texture recognition and image segmentation.

Yulin Shen received his B.S. degree in Computer Science & Technology from Southeast University, Nanjing, China, in 2018. Currently, he is pursuing his M.S. degree (No. 201921080522), supervised by Dr. Luping Ji, in Computer Science & Technology at University of Electronic Science and Technology of China, Chengdu, China. His research interests include Computer Vision, Neural Networks and Pattern Recognition.

Qian Zhang received her B.S. and M.S. degrees in School of Science respectively in 2015 and 2018 from Xi'an Polytechnic University, Xi'an, China. Currently, she is pursuing her doctoral degree (No. 201811081226) in School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China. Her research interests include Graph-based Semi-supervised learning, Gaussian process and Bayesian inference.