RAA 2019 Vol. **19** No. **3**, 42(12pp) doi: 10.1088/1674–4527/19/3/42

http://www.raa-journal.org *http://iopscience.iop.org/raa*

***R**esearch in*
***A**stronomy and*
***A**strophysics*

A deep learning approach for detecting candidates of supernova remnants

Wei Liu[1], Meng Zhu[1], Cong Dai[1], Bing-Yi Wang[1], Kang Wu[1], Xian-Chuan Yu[1], Wen-Wu Tian[2], Meng-Fei Zhang[2] and Hong-Feng Wang[3]

[1] College of Information Science and Technology, Beijing Normal University, Beijing 100875, China; *yuxianchuan@163.com, weiliu@mail.bnu.edu.cn*

[2] National Astronomical Observatories, Chinese Academy of Sciences, Beijing 100101, China; *tww@bao.ac.cn, zmf@bao.ac.cn*

[3] School of Information Management, Dezhou University, Dezhou 253023, China

Received 2018 May 19; accepted 2018 September 29

Abstract Detecting supernova remnant (SNR) candidates in the interstellar medium is a challenging task because SNRs have weak radio signals and irregular shapes. The use of a convolutional neural network is a deep learning method that can help us extract various features from images. To extract SNRs from astronomical images and estimate the positions of SNR candidates, we design the SNR-Net model composed of a training component and a detection component. In addition, transfer learning is used to initialize the network parameters, which improves the speed and accuracy of network training. We apply a T-T plot (of the different brightness temperatures of map pixels at two different frequencies) to calculate the spectral index of SNR candidates. To accelerate the scientific computing process, we take advantage of innovative hardware architecture, such as deep learning optimized graphics processing units, which increases the speed of computation by a factor of 5. A case study suggests that SNR-Net may be applicable to detecting extended sources in the images automatically.

Key words: methods: data analysis — techniques: image processing — stars: fundamental parameters

1 INTRODUCTION

Detecting interesting objects in astronomy (e.g., stars, galaxies, solar system objects, and extragalactic supernovae) is a routine task. With the development of digital sky surveying, an ever-increasing deluge of data has been generated in astronomy in recent decades (Laureijs et al. 2012). Astronomical radiation in the radio band is very weak and prone to disturbances.

Compared with the expected rate of supernova production from OB star counts, pulsar creation rates, iron abundance and observations of supernovae in other galaxies, Li et al. (1991) believe that there should be over 1000 supernova remnants (SNRs) detectable in the Galactic plane. However, there are only $\sim$250 SNRs that have been found despite an increasing number of surveys of the region (Blair et al. 1981). SNRs in the Andromeda galaxy are mainly detected in optical (e.g., Dodorico et al. 1980; Dennefeld & Kunth 1981; Blair et al. 1981) and combined optical and radio (Braun & Walterbos 1993) studies. Using optical data from the Local Group Galaxy Survey, Sasaki et al. (2012) searched for optical counterparts of the X-ray SNRs and candidates showing optical Hα, [S$_{\rm II}$], and [O$_{\rm III}$] emission from the radiative shock. Hollitt & Johnston-Hollitt (2012) explored the response of the circle Hough transform to a representative sample of different extended circular or arc-like astronomical objects, e.g., SNRs. However, Hough transform is unable to identify more details of SNRs. Beaumont et al. (2011) applied support vector machines (SVMs), a machine learning algorithm, to the task of classifying structures in the interstellar medium (ISM). Their study suggests that SVMs can be applied to classify an SNR that lies behind the M17 molecular cloud. In research into deep learning, Charnock & Moss (2017) apply deep recurrent neural networks that are supervised learning algorithms, to classify supernovae. However, the performance of the network is highly sensitive to the amount of training data. At present, researchers

have relied on the manual identification of features in the detection of SNRs. At the same time, the spectral index (Tian & Leahy 2005) is an important feature of SNRs and its values range from 0 to 1.

Deep learning (Yu & Deng 2010; Schmidhuber 2015; Sainath et al. 2015; Lecun et al. 2015; Ding et al. 2015; Jaderberg et al. 2014) is also known as deep structured learning, hierarchical learning, and deep machine learning. It is a branch of machine learning based on a set of algorithms that attempt to model high-level abstractions in data. Such an algorithm can automatically extract the features of a target object and detect the location of the target (Cheng & Han 2016). SNRs often mix with bright objects, which makes them difficult to extract. However, Convolutional Neural Networks (CNNs) can learn rich feature representations for a wide range of SNR images, and thus are powerful feature extractors.

In this paper, we design the SNR-Net model consisting of two parts: a training component and a detection component. In the training phase, we use datasets of SNRs and non-supernova remnants (N-SNRs) to train the classifier in the SNR-Net, so that the classifier has an ability to classify SNRs and N-SNRs. In the detection phase, we first transform the size of the test image and then use the trained model to get a feature map of the SNR. Moreover, all points on the feature map are traversed and the possible regions are identified as candidates for SNRs. The SNR-Net model detects candidates of SNRs according to their morphology. Using radio signal only it is sometime impossible to distinguish the remnant from the ISM background, so we need multi-band further identification of SNRs. We then employ the T-T plot method (Leahy & Tian 2005) to compute the spectral index of candidate SNRs. It should be noted that we identify the candidate SNRs only from radio data. Our collection of SNRs is derived from radio data. The main work of the present paper is to obtain SNR candidates according to their shape in the radio wavelength band, and the spectral index of candidate SNRs.

Our paper is structured as follows. The datasets are described in Section 2. In Section 3, we describe details of the relevant classification algorithms and the proposed method. Section 4 discusses experimental results obtained from SNR-Net using data from Section 2. Finally, conclusions are presented in Section 5.

2 DATA

We use a dataset from the Multi-Array Galactic Plane Imaging Survey (MAGPIS), the NRAO Very Large Array (VLA) Sky Surveys (NVSS), the Molonglo Observatory Synthesis Telescope (Rucinski et al. 2003) and the Canadian Galactic Plane Survey (CGPS). The MAGPIS database includes a wide dynamic range of high-sensitivity VLA images for the region $5^\circ < \ell < 48.5^\circ$, $\mid$ b $\mid < 0.8^\circ$ (Giveon et al. 2005). NVSS is a radio continuum survey covering the sky north of -40° declination (Condon et al. 1998). The central frequency and bandwidth of the CGPS are 1420 MHz and 408 MHz, respectively (Taylor et al. 2003). At present, there are three types of SNR. If a remnant has a 'shell' or 'filled-center' structure, it belongs to type 'S' or type 'F', respectively. If it has a 'composite' (or 'combination') radio structure with a combination of shell and filled-center characteristics, it belongs to type 'C'. Figure 1(a)–(c) illustrate the three types of SNR. We do not distinguish pulsar wind nebulae in this paper.

Many samples are needed to train the network, so we collect diverse types of SNR from the catalog of galactic SNRs listed in Green (2004) and crop 290 images of SNRs from the dataset. In the present study, we first select 108 SNRs from 290 SNRs, which include SNRs of type C, S and F. We then extend the 108 SNR samples (C-type:16, F-type:2, S-type:90) to 15 000 samples by rotating, mirroring and shifting. The augmentation part of the SNR sample is implemented by calling the image processing package in the Keras framework. Image rotation is achieved by setting various rotation angles of the image. Image shifting includes horizontal and vertical shifts. We also set the fraction of total height of the image in the Keras framework to realize image shift transformation. The mirror operation of the image is achieved using the vertical flip parameter of the image set in the Keras framework. The remaining objects are used to test the classifier. The data augmentation results are shown in Figure 2. At the same time, we collect 15 000 N-SNRs, which are cropped from areas near the SNR images (e.g., see (d)–(f) in Fig. 1).

3 RELEVANT CLASSIFICATION ALGORITHMS AND THE PROPOSED METHOD

3.1 Relevant Algorithms

In order to understand the application domain of this paper, we now discuss two relevant algorithms: Random Forest (RF) (Breiman 2001) and Support Vector Machine (SVM) (Cortes & Vapnik 1995).

RF is developed by Breiman (2001) and integrates multiple trees based on the idea of integrated learning. Its basic unit is a decision tree. RF uses a bagging approach and classification and regression trees (CART). In bagging, each classifier is built individually by working with a boot-

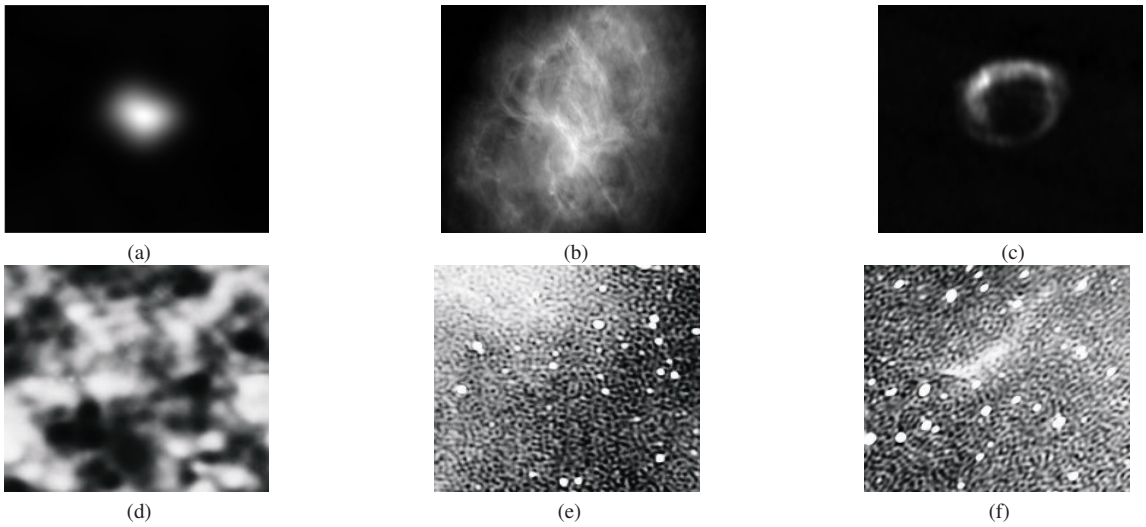

Fig. 1 Various types of SNR. (a) (G54.1+0.3), (b) (G184.6–5.8), (c) (G1.9+0.3) represent SNRs of type C, F, and S respectively. (d)–(f) represent N-SNRs.

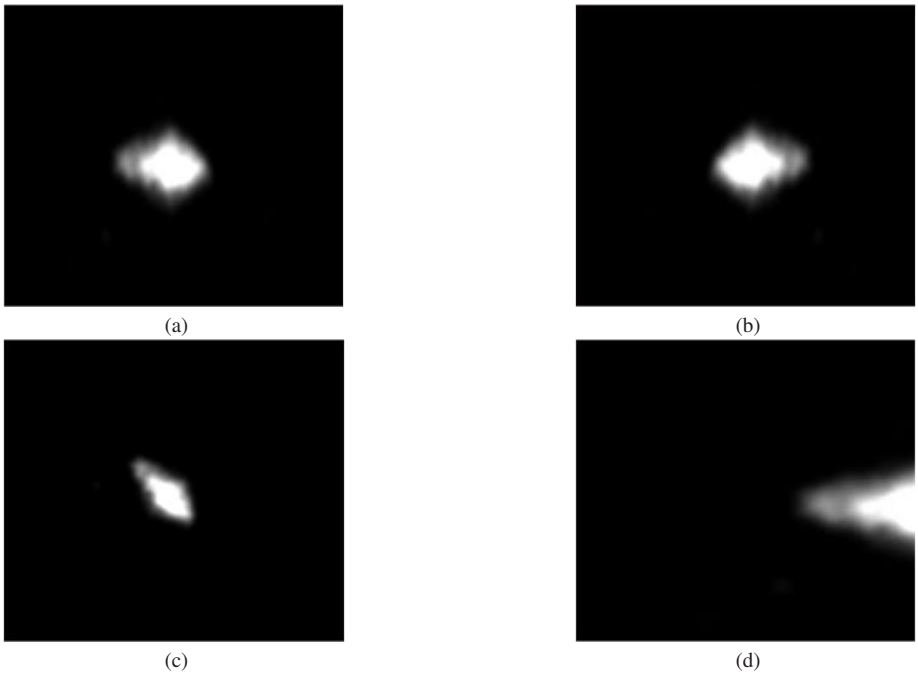

Fig. 2 Sample expansions of an image. (a) Original image (G130.7+3.1). (b) Results after mirroring. (c) Results after rotation. (d) Results after shifting.

strap sample of the input data (Alam & Vuong 2013). In CART, Gini index is used as the evaluation criterion for the selection attributes of the CART tree and is defined as

$$\mathrm{Gini}(D) = 1 - \sum_{i}^{c} P_i^2 \ , \tag{1}$$

where c is the number of classes in the target variable and p_i is the ratio of this class. If the selected attribute is A, the formula for the Gini index of the split dataset D is

$$\mathrm{Gini}_A(D) = \sum_{j=1}^{k} \frac{|D_j|}{|D|} \mathrm{Gini}(D_j) \ , \tag{2}$$

where K indicates that the sample D is divided into k parts. The number of datasets D splitting into the D_j dataset is K. We use the Gini index gain value as the basis for the selection of the decision tree:

$$\Delta\mathrm{Gini}(A) = \mathrm{Gini}(D) - \mathrm{Gini}_A(D) \ . \tag{3}$$

When the attribute is selected by the decision tree, the maximum value of the Gini index should be considered as the condition of the split node. At each node, the selected features are searched for best split. For new datasets, each case of the datasets is passed down to each of the decision trees. The forest chooses the class having the most output of votes.

SVM constructs a hyperplane in a high-dimensional space, which aims to determine the location of decision boundaries that produce the optimal separation of classes (Cortes & Vapnik 1995). Given a dataset of n points of the form

$$(\boldsymbol{x}_k, y_k) \quad k = 1, 2, \ldots, n, \tag{4}$$

where the y_k are either 1 or -1, each indicating the class to which the point $\boldsymbol{x}_k$ belongs. The hyperplane is defined as

$$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0 \ , \tag{5}$$

where $\boldsymbol{w}$ is the normal vector to the hyperplane and b is the offset. In order to ensure data points lie on the correct side of the margin, the constraints state can be defined as

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x}) - b \geq 1, \quad i = 1, \ldots, n \ . \tag{6}$$

Assuming the training data are not linearly separable, we introduce the hinge loss function:

$$\max(0, 1 - y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i - b)) \ . \tag{7}$$

The classifier is constructed by optimizing the following function:

$$\left[\frac{1}{n}\sum_{i=1}^{n} \max(0, 1 - y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i - b))\right] + \lambda ||\boldsymbol{w}||^2 \ , \tag{8}$$

where λ represents the parameter which determines the tradeoff between increasing the margin-size and ensuring that $\boldsymbol{x}_i$ lie on the correct side of the margin.

3.2 SNR-Net

This section describes the algorithm of SNR-Net. Figure 3 describes the structure of the algorithm, which includes the designs of SNR-Net. This convolutional structure is inspired by the AlexNet (Cheng et al. 2015) image classification system. Recent studies (Herbert 2016, Cheng & Han 2016) have illustrated that stacking many convolutions as AlexNet did can better capture object locations. SNR-Net consists of an input layer and an output layer, as well as multiple hidden layers. The hidden layers of SNR-Net consist of convolutional layers, pooling layers and fully connected layers as shown in Figure 4. The convolutional layer is the core building block of SNR-Net. The parameters of each layer consist of k learned filters (or kernels), which have a small receptive field. In this work, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input, and then producing an activation map of that filter:

$$y = \sum_{i=1}^{k} W_i * x + b \ , \tag{9}$$

where W are the weights, $*$ is a convolution operator, b is a bias parameter, and y is the output feature map. The input x in our method is an image of SNRs, so we adjust the typical CNN structure to meet the demands of SNR classification. The Rectified Linear Unit (ReLU) layer promotes the performance of the nonlinear decision function. In our work, we the select ReLU as the activation function of SNR-Net. The formula for the ReLU is as follows:

$$f(x) = \max(0, x) \ . \tag{10}$$

The pooling layer is used to gradually reduce the size of the representation space, the number of parameters, and the computation time in the network and also prevents over-fitting. The output of the network is classifier Softmax. A subset of features is selected as the input vector of Softmax for training and recognition.

SNR-Net essentially maps from input layer to output layer. By training SNR-Net with the known patterns, the network can map between input and output pairs. SNR-Net adopts a multi-layer feed-forward neural network and uses the back-propagation algorithm. In the forward propagation process of SNR-Net, the probability of the category to which the input belongs is calculated (Chen et al. 2016).

$$Y_i^l = f(W_i^{l-1} * x_i^{l-1} + b) \ , \tag{11}$$

where Y stands for the result of the activation function, f denotes the activation function ReLu, l represents the layer number, i denotes the ith feature map, and x represents input of the layer $l-1$.

In this back propagation process, a gradient descent algorithm is used to optimize the network parameters, which is prevalent in the field of optimization algorithms (Chen et al. 2016). Thus, the optimization object becomes

$$\begin{aligned} J(\theta) = &\sum_{i=1}^{N} \left(\frac{1}{2}\|Y(x_i, \theta) - \gamma\|^2\right) \\ &+ \lambda \sum_{l}^{L} \sum (\|\theta^{(l)}\|^2) \ , \end{aligned} \tag{12}$$

where Y denotes the result of the network and γ denotes the target output. The second item on the right is a regularizer. λ controls the strength of preference for weights, and

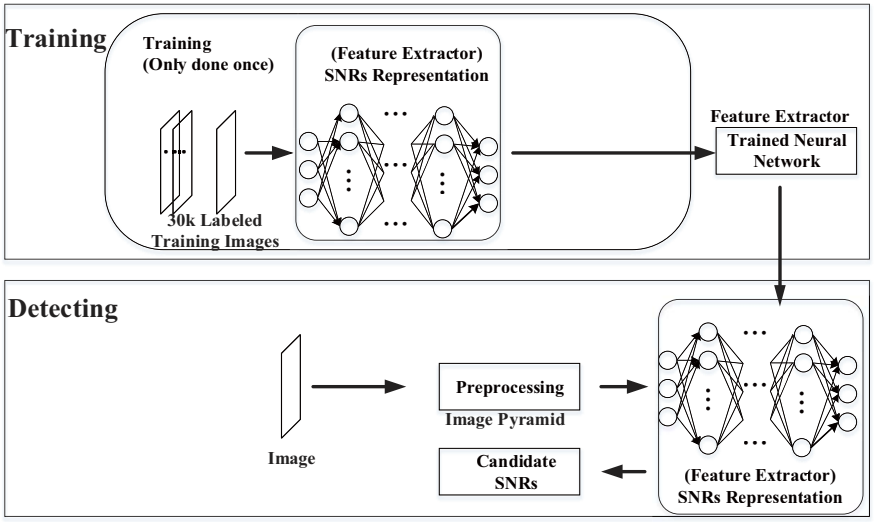

Fig. 3 Project structure of SNR-Net.

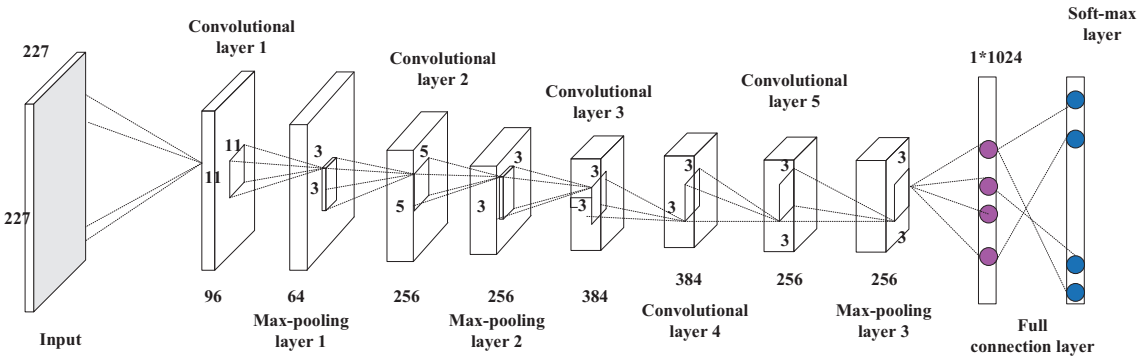

Fig. 4 Convolution architecture for SNR recognition. This is the main structure of the convolutional networks. We show the size and quantity of the convolution kernel of the convolution layer and the pooling layer.

l indexes the layer number. We try to minimize $J(\theta)$ as a function of θ.

3.3 How to Train SNR-Net

The training process for SNR-Net includes two steps: a feed forward propagation and back propagation pass. In the process of back propagation, the algorithm of gradient descent is used to adjust the weights of each layer. The original SNR and N-SNR dataset is divided into a training set and a test set, the training set is used to train the SNR-Net, while the test set is used to test the model results. In the training phase, we use the training set to train the SNR-Net model and save the well-trained model. The performance of well-trained model is tested and verified through the test set. SNR-Net is a network with large layers containing input, hidden and output layers. Details of the algorithm are given in Algorithm 1.

4 EXPERIMENTAL ANALYSIS

4.1 Datasets and Experimental Setting

We run our program with the Caffe framework and train our model on a TitanX graphics processing unit. Training the CNN requires a lot of data. Our dataset consists of two categories of object, SNRs and N-SNRs, and each category has a total of 15 000 images. We label the 30 000 images as described in Section 2. The training dataset and the validation dataset are both involved in the training process,

Algorithm 1 The convolution neural network training process

Input:
The number of training set (labeled spectra), $N_{\rm label}$;
The number of test set (unlabeled spectra), $N_{\rm test}$;
The number of samples taken from the training set for each training, $\rm batchsize$;
The number of training times in the full sample of the training set, $\rm epoch$;
The learning rate of stochastic gradient descent, lr;
Output:
classification result, C;

1: **for** The loop count $<$ epoch **do**
2: Choose batchsize samples from training set.
3: Save 1-D CNN model.
4: Save weights and parameters.
5: Test trained neural network.
6: **return** C.

and they are different datasets including different objects. The 30 000 training samples are divided into the training dataset and the validation dataset (and there is no intersection between them). A total of 22 500 images are randomly selected from the sample for training and the remaining images are reserved as a validation set to verify whether our network has over-fitting. We extend the remaining 182 (a total of 290 samples, 108 as in the training set) SNR sample images in the same way (i.e., rotation, mirroring, shifting) to 4065 samples as a positive sample of the test dataset. In the model testing phase, we collect 4096 negative samples of N-SNRs (not participating in model training) as the model test dataset.

The details of convolution networks described in Section 3 are shown in Table 1. The input of the network is an image with 227×227. Conv1 and Conv2 denote the convolution operation, and MaxPooling represents the pooling operation. The kernel size of the Conv1 layer is 11×11, and the pooling layer is 3×3. After the convolution operation, the number of feature maps of the network output is 256. FC denotes the fully connected layer, and its kernel size is 1024. The last layer of the network is the Softmax function, which is used to identify which class the input belongs to. We apply the dropout technique to prevent over-fitting. The main idea of dropout is to randomly drop units (along with their connections) from the neural network during training and the parameter of dropout is 0.5. Output of the SNR-Net model is trained by the images with the SGD method (Jin et al. 2014) after 30 epochs and the learning rate is set to 0.01. The ReLU layer promotes the performance of the nonlinear decision function. It is recommended that ReLU be used, because it makes the neural network training speed several times faster than those previously achieved (Krizhevsky et al. 2012). We use the Softmax function as a classifier. Based on the outputs of the classifier, the samples are classified into different classes.

An image pyramid is a multi-scale representation of an image, such that the SNR detection is scale-invariant. We adopt a sliding window for SNR detection. When we are training the network, the target image has been converted to 227×227 pixels. For the test image, therefore, we apply a 227×227 window to the model for classification and slide the window across the test image. We finally find the candidates for SNRs. However, the SNR candidates are not necessarily 227×227 in size, and thus we need to make a multi-scale transform. We resize the input image down to or up to the specified size. The size after transformation is a maximum of twice the dimensions of the input image, and decreases from this input image in multiples of 0.79 until the input image dimension is less than 227×227 pixels. In this way, we get many enlarged or reduced images and use all the images as the set of inputs for SNR candidates. We apply the pre-trained parameters to initialize the networks rather than randomly set parameters (Cheng et al. 2015). Models based on CNNs are popular because they accelerate the learning process.

The experimental environments include an Intel Core i5-4590 3.3-GHz CPU, 20-GB random access memory, and a Titan X GPU.

4.2 Experimental Results and Analysis

In this section, we give the visualized results generated by SNR-Net. Our proposed method is compared with SVM and RF which are widely used as supervised learning techniques in astronomical classification tasks. In this paper, we used principal component analysis (PCA) (Cheng & Han 2016) to reduce the dimensions of the features in order to improve the speed of the classifier, and then we utilized SVM and RF to classify the processed results. Moreover, the experimental results show that the proposed algorithm can obtain higher classification accuracy.

Table 1 The Structure and Setting Parameters for Convolution Networks

No.	Layer	Kernel size	Feature map	Activation function
1	input	227×227	0	–
2	Conv1	11×11	96	ReLU
3	MaxPooling_1	3×3	96	–
4	Conv2	1×16	256	ReLU
5	MaxPooling_2	3×3	256	–
6	Conv3	3×3	384	ReLU
7	Conv4	3×3	384	ReLU
8	Conv5	3×3	256	ReLU
9	MaxPooling_3	3×3	256	–
10	FC	1×1024	–	ReLU
11	output	1×2	–	Softmax

Figure 5 gives the visualized results generated by SNR-Net in the training phase. It can be seen from the experimental results that SNR-Net can extract detailed features from the image of SNRs. In other words, we get a well-trained SNR-Net model, which is beneficial for SNR classification in the next steps.

During the training process, we choose a suitable training method (i.e., a variable learning rate method) adaptive to the system to improve the accuracy as shown in Figure 6. We use classification accuracy to justify the effect of data augmentation on network classification performance.

Figure 7 shows the classification accuracy of the network for SNRs and N-SNRs without data augmentation operation. As can be seen from Figure 7, it is obvious that the network causes over-fitting. Due to a lack of sufficient training data used in classification modeling, its initialized weights and threshold are difficult to determine, requiring repeated training to determine the network structure and various parameters, easily causing the over-fitting and affecting the network generalization ability. We apply three techniques to prevent over-fitting: data augmentation, regularization, and dropout.

Figure 8 shows the classification results of SNR-Net using data augmentation. In the training phase, the accuracy of our model reaches 99.68% and the loss value drops to 0.01.

To speed up the convergence, we first pre-train each layer using face datasets (Rothe et al. 2016) to initial weights, and then fine tune the weights with SNR and N-SNR datasets. The experimental results in Table 2 show that pre-trained neural networks (SNR-Net) achieve higher training accuracy than neural networks that are not pre-trained (SNR-Net1). In addition, we use the 8161 images that are not used in the training as the test data set. The accuracy and loss of SVM, RF, SNR-Net and SNR-Net1 on the dataset are respectively shown in Table 2, which presents the performance of SVM, RF, SNR-Net and SNR-Net1 in SNR and N-SNR classification. The results in Table 2 show that the test accuracy of SNR-Net is 98.45%. The accuracy of the dataset is defined as:

Table 2 Comparison of different methods. Accuracy and Loss represent the training accuracy and training loss in the SVM, RF, SNR-Net and SNR-Net1 training process, respectively. Test accuracy represents the classification results on the test dataset obtained using the well-trained model.

Algorithm	Accuracy (%)	Loss	Test accuracy (%)
SVM	87.34	0.81	83.63
RF	96.23	0.08	94.16
SNR-Net	99.68	0.01	98.45
SNR-Net1	99.48	0.05	98.16

$$\text{accuracy} = \frac{1}{N}\sum_{i=1}^{N} I\left(y_i = f(x_i)\right) \ , \qquad (13)$$

where I is an indicator function and N represents the number of samples, i.e., the I function value is 1 when the predicted value of $f(x_i)$ is equal to the label value y_i, and zero otherwise. A higher value of accuracy indicates a better ability to identify SNRs. The confusion matrices for the classification results are shown in Figure 9.

In Figure 9, the confusion matrix shows that false positives are more than false negatives. The reason for this is that the morphology of the ionized hydrogen zone is very similar to that of SNRs. A Receiver Operating Characteristic (ROC) curve is a plot of the true positive rate against the false positive rate, which is commonly used for evaluating binary classification. In order to compare the performance of different methods, we show the ROC of SNR-Net, RF and SVM in Figure 10. Furthermore, we use the T-T plot method to calculate the spectral index of the candidates to improve the classification accuracy, because the spectrum index values of SNRs generally range from 0 to 1. We take the spectrum index as an important feature of SNRs.

The trained neural network can distinguish candidate SNRs in the ISM. However, because of errors in the neural network, there are a lot of blurred results that are approximately right or wrong near the candidates for SNRs. In order to get the best result, we apply the probability threshold method to handle the network outputs; i.e., we choose the regions where the probability is greater than a threshold as the SNR candidates.

The data source we select is the Canadian Galactic Plane Survey (Taylor et al. 2003) ($82.2^\circ < \ell < 87.3^\circ$,

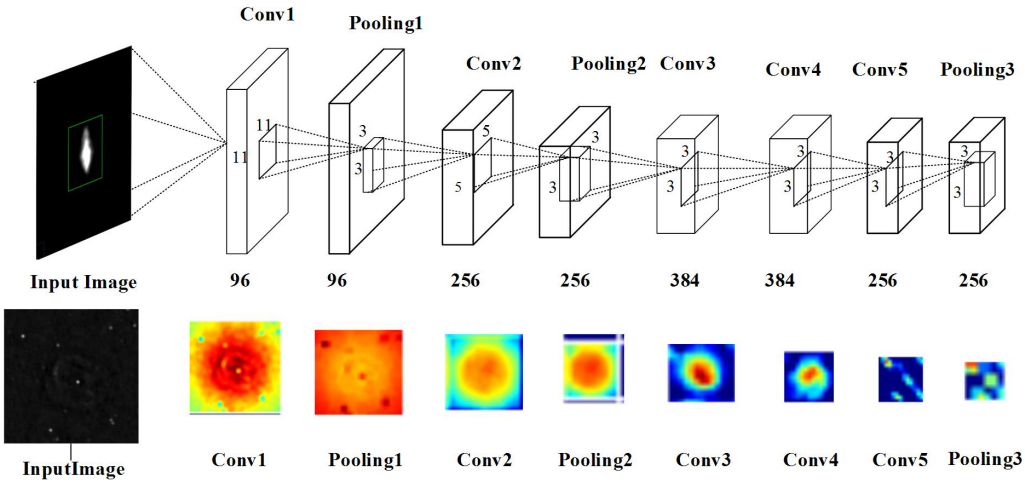

Fig. 5 Convolution architecture for SNR recognition. This is the main structure of the convolutional networks. We show the size and quantity of the convolution kernel of the convolution layer and the pooling layer. The second line shows the visual results of the convolution and pooling operations, as well as the process of extracting features from the network.

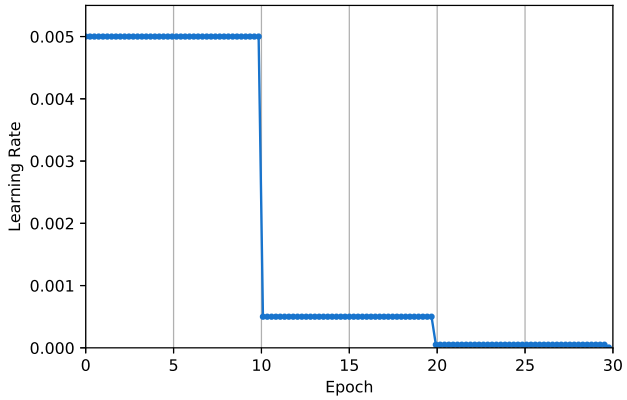

Fig. 6 Change in the learning rate with number of iterations.

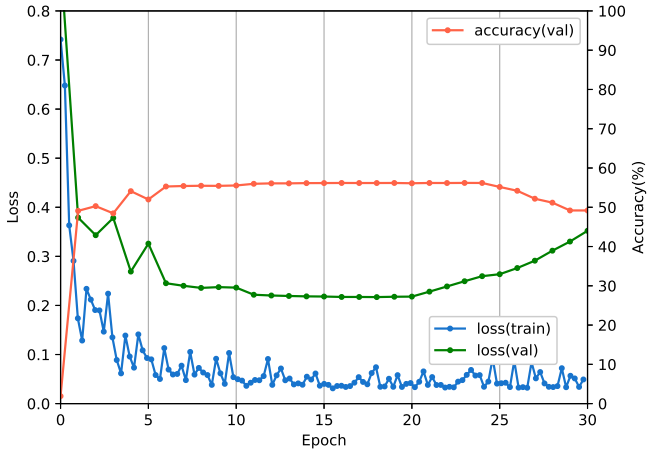

Fig. 7 Over-fitting in SNR-Net learning. Training loss is shown in *blue*, validation loss in *green*, both as a function of the number of training cycles. The validation loss increases while the training loss steadily decreases, then a situation of over-fitting may have occurred. The best predictive and fitted model would be where the validation loss has its global minimum.

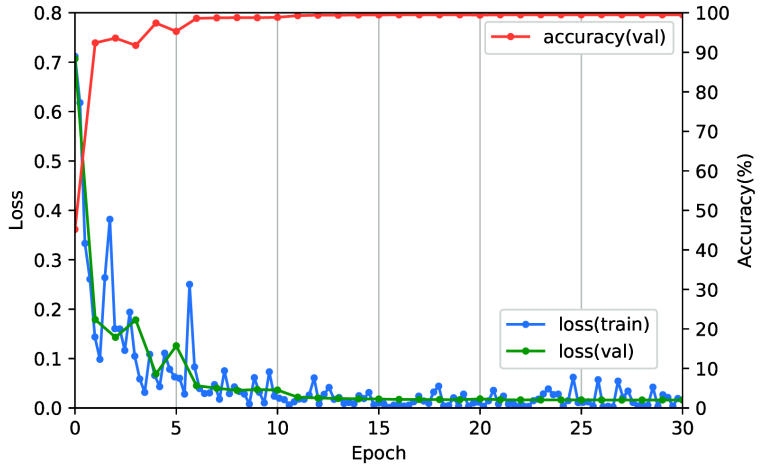

Fig. 8 Accuracy and loss values for different iteration times. The training loss and validation loss both decrease, so it effectively avoids over-fitting.

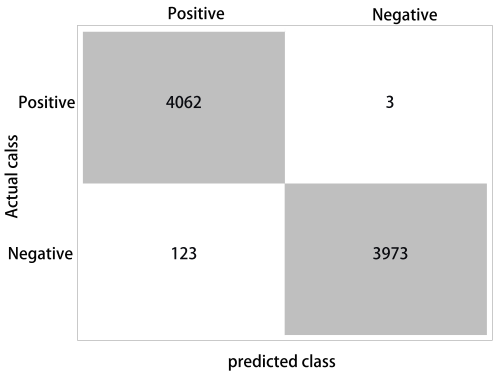

Fig. 9 Confusion matrix. The classification results are obtained by the trained classifier. In this confusion matrix, of the 4065 actual SNRs, the system predicted that 4062 were SNRs, and of the 3096 N-SNRs, it predicted that 123 were SNRs and 2973 were N-SNRs. For 4065 SNR samples, the classifier correctly classified 4062 images and incorrectly classified three images. For 4096 N-SNRs samples, the classifier correctly classified 3973 images and incorrectly classified 123 images.

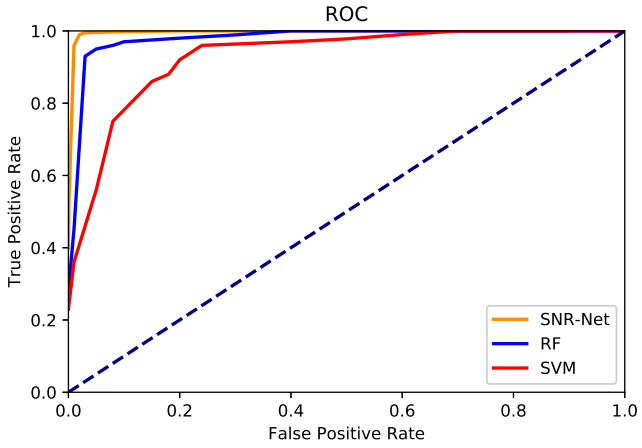

Fig. 10 ROC comparison of different methods. The ROC of SNR-Net, RF and SVM are respectively shown in *orange*, *blue* and *red*.

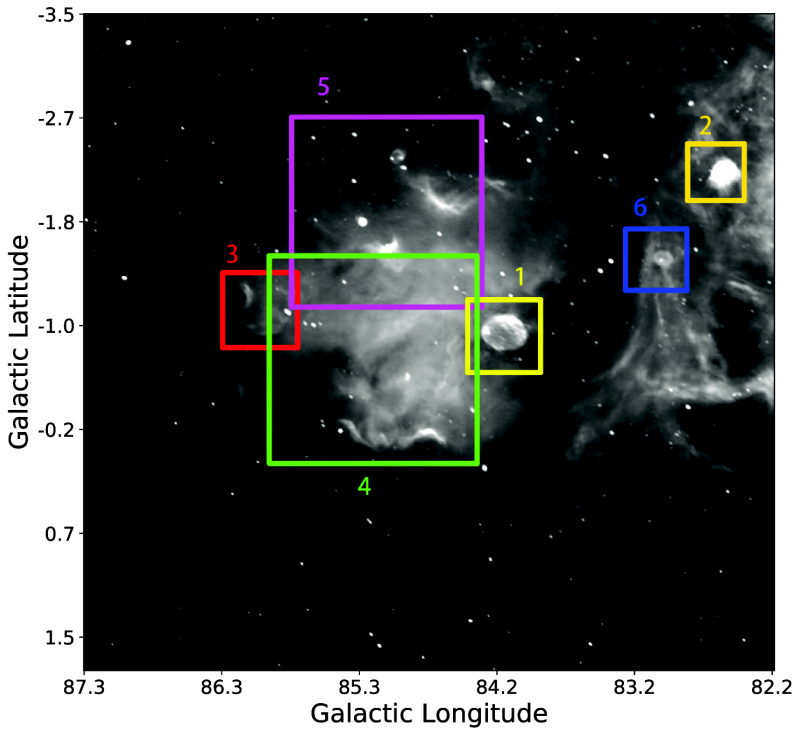

Fig. 11 The image shows the CGPS map at 1420 MHz ($82.2^\circ < \ell < 87.3^\circ$, $-3.5^\circ < b < 1.4^\circ$). Six candidate SNR areas are predicted by the SNR-Net model according to morphology. The probabilities of candidate SNR regions (1–6) predicted by the SNR-Net model are 0.953, 0.946, 0.937, 0.923, 0.916, and 0.913, respectively.

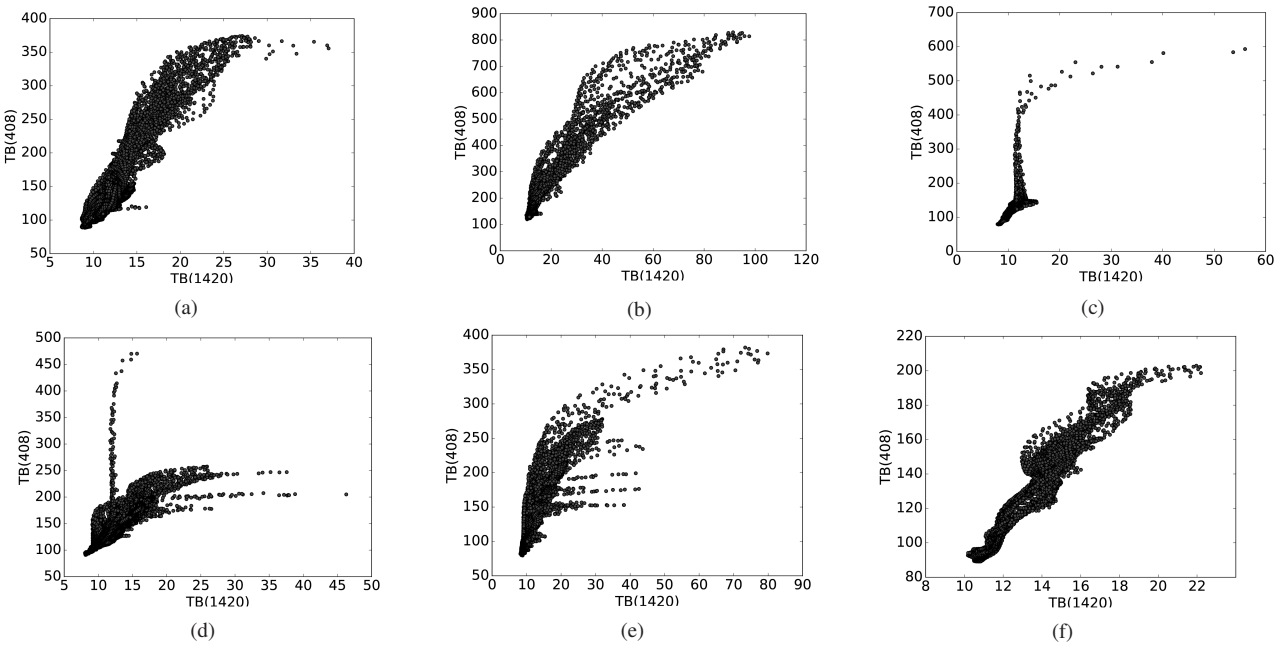

Fig. 12 T-T plots of SNR candidates at 408 MHz and 1420 MHz. (a)–(f), respectively, represent the T-T plots of areas 1–6 in Fig. 11.

$-3.5^\circ < b < 1.4^\circ$). We set the probability threshold at 0.9 and obtain the candidate regions predicted by the SNR-Net model, as shown in Figure 11. We use the T-T plot method to calculate the spectral index. The principle of the T-T plot method is that spectral indices ($T_\nu = T_\circ \nu^{-\beta}$) are calculated from a fit of a linear relation to the T_1–T_2 values of all pixels within a given map region (Leahy & Tian 2005).

Table 3 Comparison of different methods. Accuracy and Loss represent the accuracy and loss in the CNN training process, respectively. Test accuracy represents the classification results on the test dataset obtained using the CNN trained model.

Area	β	α
1	2.33	0.33
2	1.91	–0.09
3	2.21	0.21
4	1.91	–0.09
5	1.84	–0.16
6	2.02	0.02

The flux density spectral index α ($S_\nu \propto \nu^{-\alpha}$) is related to β by $\beta = \alpha + 2$.

In this paper, T_1 is the brightness temperature of a map pixel at 1420 MHz and T_2 is that at 408 MHz. Through the T-T plot method, we obtain the spectral index of the SNR candidates as shown in Figure 11. The probabilities refer to the scores of the SNR candidates, which are derived from the trained classifier. It is the final layer of the network that yields the actual probability scores for each class label. The SNR-Net model predicts that the candidate SNR areas contain three SNRs (candidates of SNRs 1, 3, 6 in Fig. 11) in the Green Catalogue of Galactic Supernova Remnants (G84.2–0.8, G85.9–0.6, G83.0–0.3). Candidate area 5 contains part of an SNR (G85.4+0.7). These six regions (Fig. 11) yield the T-T plots, as shown in Figure 12. We use all the pixels at 408 MHz and 1420 MHz in each SNR candidate region to fit the slope. Subplots (a)–(f) in Figure 12 represent the spectral indices of SNR candidates 1–6 in Figure 11. Table 3 gives the spectral indices of SNR candidates.

5 CONCLUSIONS

We introduced a deep learning method for feature extraction from radio images. Our developing SNR-Net model consists of two components: network training and target object detection. We first train a network using datasets that cover three SNR types. The algorithm trains the network parameters to learn features of the input images, which can be fed to a simple classifier. In the detection phase, SNR-Net is combined with a size transformation algorithm. As a result, SNR-Net can identify SNR candidates of various size and report a probability for the SNR position. In this paper, we used a deep learning algorithm and a T-T plot algorithm to build a model for searching for SNR candidates. The advantage of using CNNs for feature extraction is that users do not have to worry about specific features. CNNs are powerful feature extractors, capable of automatically learning complex features for object recognition and provide features superior to hand-crafted features (Ng et al. 2015). We have listed the software at the Astrophysics Source Code Library (*http://ascl.net/code/v/1742*).

Acknowledgements This study is supported by the National Natural Science Foundation of China (No. 41272359), the Ministry of Land and Resources for the Public Welfare Industry Research Projects (201511079-02) and the Natural Science Foundation of Shandong (No. ZR2015FL006). We gratefully acknowledge NVIDIA for GPU donation.

References

Alam, M. S., & Vuong, S. T. 2013, in IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, 663

Beaumont, C. N., Williams, J. P., & Goodman, A. A. 2011, ApJ, 741, 14

Blair, W. P., Kirshner, R. P., & Chevalier, R. A. 1981, ApJ, 247, 879

Braun, R., & Walterbos, R. A. M. 1993, A&AS, 98, 327

Breiman, L. 2001, Machine Learning, 45, 5

Charnock, T., & Moss, A. 2017, ApJ, 837, L28

Chen, Y., Jiang, H., Li, C., Jia, X., & Ghamisi, P. 2016, IEEE Transactions on Geoscience & Remote Sensing, 54, 6232

Cheng, G., Han, J. W., 2016, Isprs Journal of Photogrammetry and Remote Sensing, 117 (*https://www.sciencedirect.com/ science/article/pii/S0924271616300144*)

Cheng, B., Zhang, D., & Shen, D. 2015, IEEE Trans Biomed Eng, 62, 1805

Condon, J. J., Cotton, W. D., Greisen, E. W., et al. 1998, AJ, 115, 1693

Cortes, C., & Vapnik, V. 1995, Machine Learning, 20, 273

Dennefeld, M., & Kunth, D. 1981, AJ, 86, 989

Ding, S., Lin, L., Wang, G., & Chao, H. 2015, Pattern Recognition, 48, 2993

Dodorico, S., Dopita, M. A., & Benvenuti, P. 1980, A&AS, 40, 67

Giveon, U., Becker, R. H., Helfand, D. J., & White, R. L. 2005, AJ, 130, 156

Green, D. A. 2004, Bulletin of the Astronomical Society of India, 32, 335

He, J., Zhang, Y., Zhou, Y., & Zhang, L. 2016, Iet Signal Processing, 10, 1000

Herbert, S. J. 2016, Electronics Letters, 52, 1963

Hollitt, C., & Johnston-Hollitt, M. 2012, PASA, 29, 309

Jaderberg, M., Simonyan, K., Vedaldi, A., & Zisserman, A. 2014, International Journal of Computer Vision, 116, 1

Jin, J., Fu, K., & Zhang, C. 2014, IEEE Transactions on Intelligent Transportation Systems, 15, 1991

Jolliffe, I. T. 1986, Principal component analysis (Springer Series in Statistics, Berlin: Springer, 1986)

Krizhevsky, A., Sutskever, I., & Hinton, G. E. 2012, Advances in Neural Information Processing Systems, 1097

Laureijs, R., Gondoin, P., Duvet, L., et al. 2012, in Proc. SPIE, 8442, Space Telescopes and Instrumentation 2012: Optical, Infrared, and Millimeter Wave, 84420T

Leahy, D., & Tian, W. 2005, A&A, 440, 929

Lecun, Y., Bengio, Y., & Hinton, G. 2015, Nature, 521, 436

Li, Z., Wheeler, J. C., Bash, F. N., & Jefferys, W. H. 1991, ApJ, 378, 93

Ng, Y. H., Yang, F., & Davis, L. S. 2015, Computer Vision and Pattern Recognition Workshops, 53

Rothe, R., Timofte, R., & Gool, L. V. 2016, International Journal of Computer Vision, 1

Rucinski, S., Carroll, K., Kuschnig, R., Matthews, J., & Stibrany, P. 2003, Advances in Space Research, 31, 371

Sainath, T. N., Kingsbury, B., Saon, G., et al. 2015, Neural Networks, 64, 39

Sasaki, M., Pietsch, W., Haberl, F., et al. 2012, A&A, 544, A144

Schmidhuber, J. 2015, Deep Learning in Neural Networks (Elsevier Science Ltd.)

Taylor, A. R., Gibson, S. J., Peracaula, M., et al. 2003, AJ, 125, 3145

Tian, W. W., & Leahy, D. 2005, A&A, 436, 187

Yu, D., & Deng, L. 2010, IEEE Signal Processing Magazine, 28, 145