

Pulsar candidate classification with deep convolutional neural networks

Yuan-Chao Wang^{1,2}, Ming-Tao Li^{1,2}, Zhi-Chen Pan^{3,4,5} and Jian-Hua Zheng^{1,2}

¹ National Space Science Center, Chinese Academy of Sciences, Beijing 100190, China; limingtao@nssc.ac.cn

² University of Chinese Academy of Sciences, Beijing 100049, China

³ National Astronomical Observatories, Chinese Academy of Sciences, Beijing 100101, China

⁴ Center for Astronomical Mega-Science, Chinese Academy of Sciences, Beijing 100101, China

⁵ CAS Key Laboratory of FAST, National Astronomical Observatories, Chinese Academy of Sciences, Beijing 100101, China

Received 2019 January 1; accepted 2019 April 19

Abstract As the performance of dedicated facilities has continually improved, large numbers of pulsar candidates are being received, which makes selecting valuable pulsar signals from the candidates challenging. In this paper, we describe the design for a deep convolutional neural network (CNN) with 11 layers for classifying pulsar candidates. Compared to artificially designed features, the CNN chooses the sub-integrations plot and sub-bands plot for each candidate as inputs without carrying biases. To address the imbalance problem, a data augmentation method based on synthetic minority samples is proposed according to the characteristics of pulsars. The maximum pulses of pulsar candidates were first translated to the same position, and then new samples were generated by adding up multiple subplots of pulsars. The data augmentation method is simple and effective for obtaining varied and representative samples which keep pulsar characteristics. In experiments on the HTRU 1 dataset, it is shown that this model can achieve recall of 0.962 and precision of 0.963.

Key words: pulsars: general — methods: statistical — methods: data analysis

1 INTRODUCTION

Searching for pulsars is an important frontier in radio astronomy. Scientists are paying more attention to pulsars because of their broad impact across physics, astronomy, astronautics (Cordes et al. 2004; Lorimer et al. 1998; Lyne et al. 2004; Hobbs et al. 2009; Sheikh et al. 2006), etc. Many dedicated surveys have been used to search for more pulsar signals, such as the Parkes Multi-beam Pulsar Survey (PMPS, Manchester et al. 2001), High Time Resolution Universe (HTRU, Keith et al. 2010) survey and so on. With the advent of large-scale facilities that can conduct such surveys, such as the Five-hundred-meter Aperture Spherical radio Telescope (FAST, Nan et al. 2011) and the Square Kilometre Array (SKA, Smits et al. 2009), weaker pulsar signals can be received, although being mixed with more and more noise or radio frequency interferences (RFIs), which makes it difficult to identify valuable suspected pulsar signals from large numbers of pulsar candidates. Researchers have applied many successful methods to select pulsar candidates, including manual

selection (Stokes et al. 1986; Johnston et al. 1992), selection with graphical tools (Faulkner et al. 2004; Keith et al. 2009), ranking and scoring approaches (Lee et al. 2013) and machine learning methods (Eatough et al. 2010; Bates et al. 2012; Morello et al. 2014; Zhu et al. 2014; Lyon et al. 2016; Devine et al. 2016; Tan et al. 2018; Guo et al. 2017).

In related works, supervised machine learning methods have become more significant and the major methods in classifying pulsar candidates.

The first published work which attempted to use a machine learning approach to select candidates is Eatough et al. (2010). They implemented artificial neural networks (ANN) with 12 designed experimental features as input vectors. Then Bates et al. (2012) and Morello et al. (2014) performed some work to improve the performance by optimizing designed features with ANN. In these methods, the designed features relied on human experience, and may carry unexpected biases against particular types of pulsar candidates (Morello et al. 2014; Lyon et al. 2016). To address these problems, Lyon et al. (2016) and Tan et al. (2018) selected fundamental and statistical features which

aimed to minimize biases and selection effects (Morello *et al.* 2014) to provide better generalization performance.

In addition to these approaches with artificially designed features, data-driven methods also play an important role in this field. Zhu *et al.* (2014) developed the Pulsar Image-based Classification System (PICS) by using a group of supervised machine learning approaches. It produces classification based on image patterns. The inputs are four important diagnostic plots of candidates rather than extracted features. This avoids possible shortcomings in artificially designed features and relying on excessive information. It has been validated to have a superior ability at recognition in the PALFA survey pipeline and has discovered six new pulsars. To address class imbalance problems in pulsar candidates, Guo *et al.* (2017) used a Deep Convolution Generative Adversarial Network (DCGAN, Radford *et al.* 2015) to generate more candidates and automatically extract deep features at the same time. Then they used deep features to classify data, which helps to make the classifier more accurate.

In this paper, we take a step towards improving performance by the data-driven method. We designed a deep convolutional neural network (CNN) with eight convolutional layers, one flatten layer and two fully connected layers. The inputs are the sub-integrations plot and sub-bands plot in each candidate rather than artificially designed features. To improve class balancing, we designed a simple and useful approach to synthesize more diverse pulsar candidates. New samples were synthesized by adding up multiple subplots of pulsars after maximum pulses of pulsars were shifted to the same position. We tested our model on the HTRU 1 dataset. The results show that our model can provide satisfactory results on both recall and precision. This paper is organized as follows: In Section 2, the dataset applied for training is introduced. Section 3 describes the data augmentation method for pulsar candidates. Section 4 introduces the network architecture and training details. Section 5 presents the experimental results of our model and analyses of its performance. Finally, Section 6 is the conclusions of our work.

2 DATASET

To train and test our model, we need labeled convective datasets. At present, there are relatively few public labeled datasets. The most common one is the HTRU 1 dataset¹, produced by Morello *et al.* (2014).

This dataset is a part of the outputs from new processing of HTRU intermediate Galactic latitude data (Morello *et al.* 2014). It contains 1196 pulsars from 521 distinct

sources with varying spin periods, duty cycles and signal to noise ratios (SNRs). Furthermore, it has 89 995 non-pulsar candidates. It has been examined in some recent works (Morello *et al.* 2014; Lyon *et al.* 2016; Guo *et al.* 2017; Ford 2017). In this paper, we implemented it to train and measure our model.

Figure 1 is an example of a pulsar candidate (pulsar_0023) in the HTRU 1 dataset with its four most important subplots. The first subplot is a folded profile plot. It was obtained by summing the signal in all frequencies and periods. The pulse profile of a typical pulsar would be composed of one or several narrow peaks above the noise floor. The lower left one is the sub-integrations plot, and it is obtained by summing the data from different frequency channels. It reflects the intensity of the signal during the observation time. For an ideal pulsar signal, the signal would be observed throughout the observation period, so one or several vertical stripes will form corresponding to the peak positions in the profile curve. The sub-bands plot is at the upper right. By summing data over all periods, it reflects the intensity of the signal at different frequencies. Since radio pulsars are broadband, there should be one or more vertical stripes in most frequencies. In the dispersion measure (DM)-SNR curve at the lower right, the SNR as a function of DM is recorded. As the pulse passes through the interstellar medium, it would disperse. The dispersion curve shows the corresponding SNR of the pulse curve when different dispersion values are used for de-dispersion. Therefore, if it is a pulsar signal, the curve will have a peak at the non-zero position, which means the correct value is used for de-dispersion.

In the training process, the inputs are the sub-integrations subplot and sub-bands subplot of each candidate. In HTRU 1, the data size of subplots in each candidate may be different. For convenience when extracting features, each subplot data point was resized to the same size, 64×64 . The HTRU 1 dataset is very imbalanced (approximately 1:75). As is well known, class imbalance has a negative effect on the performance of the resulting classifiers (He & Garcia 2009; Buda *et al.* 2018). So, before training our model, it is necessary to address this effect.

3 DATA AUGMENTATION

In this section, the data augmentation methods that are employed to address class imbalance are introduced. In deep learning, data augmentation techniques, such as rotation, rescaling, shifting, shearing, local warping, adding noise, etc., are frequently applied to increase the diversity of the classes and reduce overfitting of the model. In this situation, we need more pulsar candidates for training, but the traditional techniques (such as rotation or shearing) are not

¹ <http://astronomy.swin.edu.au/~vmorello/>

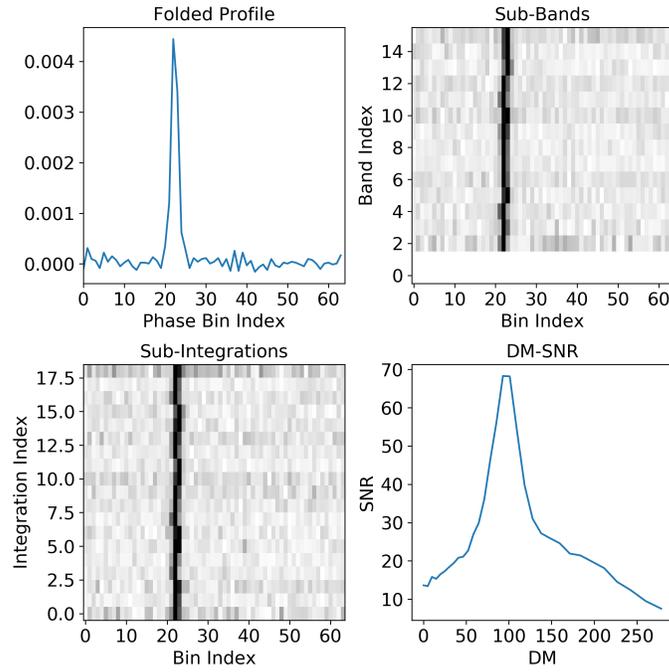


Fig. 1 A pulsar example (pulsar_0023) from the HTRU 1 dataset. The four subplots are the folded profile plot (*upper left*), sub-integrations plot (*lower left*), sub-bands plot (*upper right*) and DM-SNR curve (*lower right*).

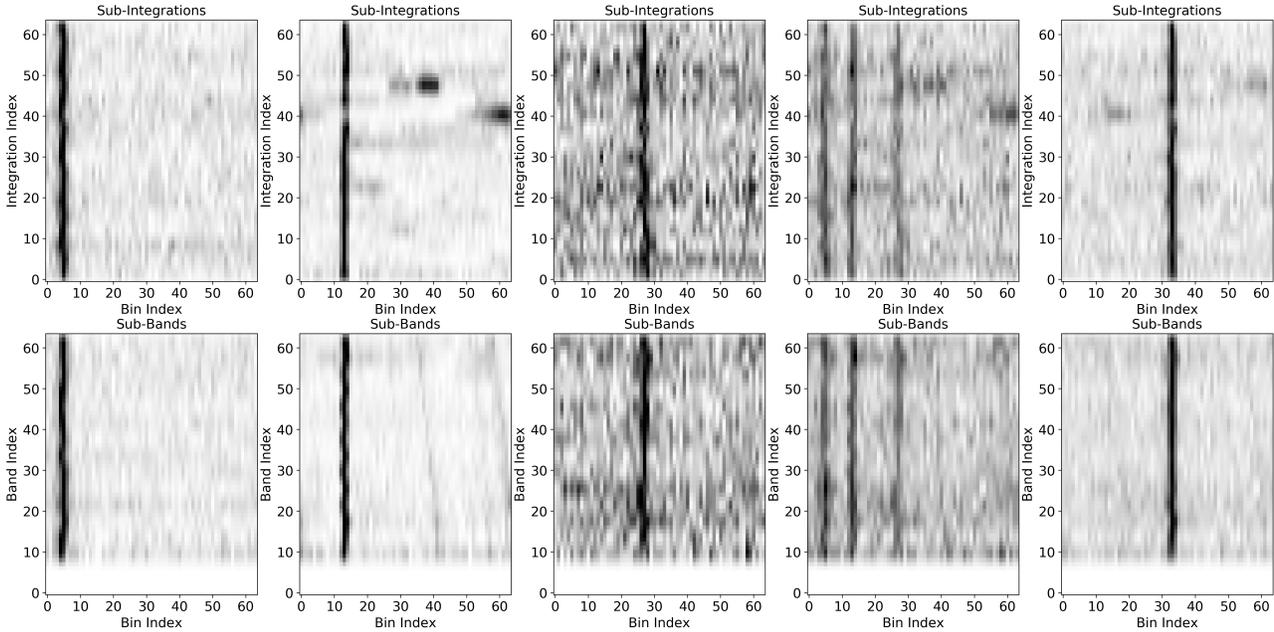


Fig. 2 An example of image synthesis. From left to right are pulsar_25, pulsar_62, pulsar_147, one of the synthetic samples without firstly adjusting pulse positions and one of the synthetic samples after firstly adjusting pulse positions ($\alpha = \beta = \gamma = \frac{1}{3}$). The upper row displays the sub-integrations plots and the lower row depicts the sub-bands plots.

suitable for pulsar characteristics. Considering this, a simple and specific image synthesis approach is designed to “generate” more pulsar candidates.

Our method is an oversampling method based on minority synthetic samples. The details are as follows:

(1) Translating the maximum pulse to the same middle position for all pulsar candidates.

(2) Randomly selecting three candidates A , B and C from training pulsar sub-dataset every time.

(3) Adding up their corresponding sub-integrations plots or sub-bands plots with random coefficients.

$$\begin{aligned} \text{sub_int}(\text{new}) = & \alpha \cdot \text{sub_int}(A) + \beta \cdot \text{sub_int}(B) \\ & + \gamma \cdot \text{sub_int}(C), \end{aligned} \quad (1)$$

$$\begin{aligned} \text{sub_band}(\text{new}) = & \alpha \cdot \text{sub_band}(A) + \beta \cdot \text{sub_band}(B) \\ & + \gamma \cdot \text{sub_band}(C), \end{aligned} \quad (2)$$

where $\alpha + \beta + \gamma = 1$, and α , β and γ are randomly selected.

(4) Repeat steps 1–3 until enough samples are produced.

In this way, some new pulsar candidates were generated as supplementary positive samples in training. It should be noticed that making an adjustment to pulse positions before selecting samples in step 1 is necessary to keep the characteristics of pulsars. Because positions of pulse phase for different pulsars varies, adding up multiple subplots in the above method may lead to multi-pulse plots, which are messy and may change the sample distribution (an example is shown in Fig. 2 column 4). The inputs are subplots with the same size, so this method did not use features of varied pulse width directly. In each subplot, vertical stripe width is related to its pulse width. When adding up different subplots, this aspect will influence the intensity or width of vertical stripes, and enhance diversity of training data. In addition, to further increase diversity, pulse position of positive pulsar candidates can be shifted randomly, when the generative process has finished.

Figure 2 shows a synthetic sample. In this figure, the upper row is sub-integrations plots and the lower row is the corresponding sub-bands plots. The first three columns are data from three pulsars. The fourth column is one of their failed synthetic samples without making adjustment to pulse positions before applying steps 2 and 3. In the last column, one of the correct synthetic samples is displayed. The new synthetic sample exhibits different noises and signal strengths that correspond with the original samples.

In order to investigate the distribution of new samples, Figure 3 depicts a visualization of 500 original pulsar samples and 500 synthetic samples shown by using t-distributed Stochastic Neighbor Embedding (t-SNE, Der Maaten & Hinton 2008). The t-SNE is a non-linear dimensionality reduction algorithm utilized for visualization. In Figure 3, the corresponding high-dimensional data are reduced to two dimensions for visualization. These synthetic samples keep the characteristics of original samples while having some changes in the strength of signals or noise.

Table 1 Number of Samples in the Three Parts

| Part | All | Pulsar | Non-pulsar |
|----------------|--------|--------|------------|
| Training set | 75 507 | 21 510 | 53 997 |
| Validation set | 18 240 | 240 | 18 000 |
| Testing set | 18 240 | 239 | 17 799 |

Table 2 Performance of Different Methods on the HTRU 1 Dataset

| Reference | Method | Recall | Precision | F-Score |
|-------------------|-------------|--------|-----------|---------|
| Guo et al. (2017) | CNN-1 | 0.956 | 0.950 | 0.953 |
| | CNN-2 | 0.953 | 0.951 | 0.952 |
| | DCGAN-SVM-1 | 0.963 | 0.965 | 0.964 |
| | DCGAN-SVM-2 | 0.966 | 0.961 | 0.963 |
| Our method | DCNN-C | 0.851 | 0.848 | 0.849 |
| | DCNN-S | 0.962 | 0.963 | 0.962 |

4 MODEL

4.1 Convolutional Neural Networks

In this section, CNNs are briefly introduced. CNNs have achieved outstanding performance in computer vision, speech recognition and natural language processing in recent years (Nielsen 2015). Many CNN modules have been designed for different tasks, such as AlexNet (Krizhevsky et al. 2012), VGG (Simonyan & Zisserman 2015), ResNets (He et al. 2016), CapsuleNet (Sabour et al. 2017), etc. These models have become deeper and more complex to meet the needs of different tasks.

A typical CNN has convolution layers, pooling layers and fully connected layers. Convolution layers are used to extract features. In convolution layers, convolution is a specialized kind of linear operation (Goodfellow et al. 2016) on feature maps from previous layers with learnable kernels. Then an activation function (such as sigmoid, hyperbolic tangent, softmax, rectified linear unit or Leaky ReLU) makes a linear or non-linear operation on the output of the kernels to form the output feature maps. In this way, each of the output feature maps can be combined with more than one input feature map (Alom et al. 2018). The new feature maps will be the inputs of the next layer. The convolution can be expressed as

$$x_j^l = f\left(\sum_{i \in M_j} w_{ij}^l * x_i^{l-1} + b_j^l\right), \quad (3)$$

where x_j^l is the output of the current layer, x_i^{l-1} is the previous layer output, w_{ij}^l is the kernel weight for the present layer, b_j^l is the biases for the current layer and f represents the activation function.

Pooling layers are usually sandwiched between convolution layers. They represent a down sampled operation (such as average pooling or max-pooling) on the input maps to compress data and reduce overfitting. Fully connected layers are usually at the ends of CNNs. They

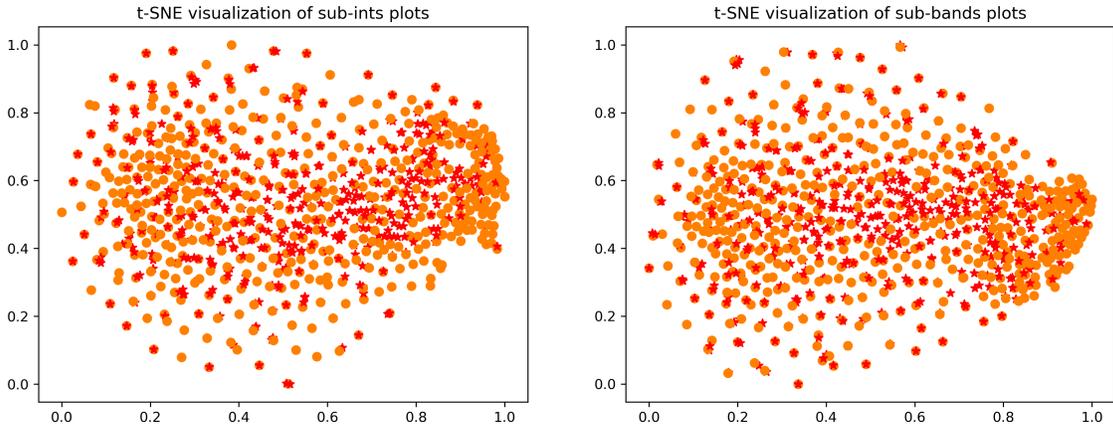


Fig. 3 Visualization of plots using t-SNE. ‘●’ marks the 500 original pulsar data, while ‘★’ signifies 500 synthetic samples. The left panel is based on sub-integrations, and the right one is based on sub-bands. We can find that these synthetic samples vary slightly from the original data.

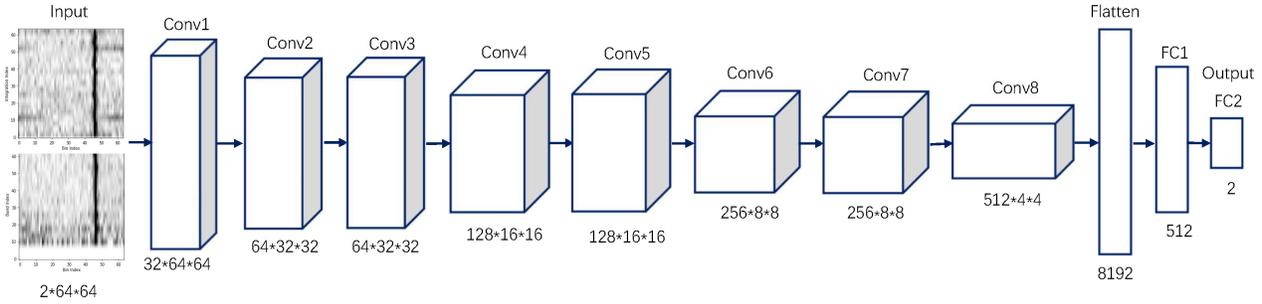


Fig. 4 The CNN architecture implemented in this design. This model takes in grayscale subplots with size $64 \times 64 \times 2$ and outputs the classification probability scores for two classes. The network has a total of 11 layers with eight convolutional layers, one flatten layer and two fully connected layers.

are used to integrate extracted features from the preceding layers and output the score of each class. In general, there would be a flatten layer before fully connected layers are used for flattening any input tensor into a vector.

4.2 Model Architecture

For this classification task, a deep CNN model was designed with eight convolutional layers, one flatten layer and two fully connected layers. The network takes sub-integrations plots and sub-bands plots as inputs. The input size is $64 \times 64 \times 2$, and outputs a tensor of size 1×2 , which means the predicted probability of a non-pulsar or pulsar, respectively. The network architecture is shown in Figure 4.

In the convolutional layers, Leaky ReLU was chosen as the activation function, and ReLU and sigmoid activation was used for the two fully connected layers respectively.

In order to reduce overfitting, some useful techniques were used, including batch normalization (Ioffe & Szegedy 2015), dropout (Hinton et al. 2012) and L2 regularization.

We added batch normalization with a momentum of 0.9 before each activation layer. This has been demonstrated to help accelerate training processes (Alom et al. 2018). Then, dropout was applied with alpha of 0.25 after each activation in all convolutional layers and an alpha of 0.55 for the first fully connected layer. The key idea of dropout is to randomly delete some units during training (Srivastava et al. 2014) with a fixed probability (alpha). In addition, L2 regularization incorporates a regularization term on the cost function for weight decay. In this model, we used L2 regularization with a parameter of 0.02.

4.3 Details

The loss function utilized for optimizing the model is a binary cross-entropy cost function. The binary cross-entropy cost function can be expressed as

$$L = -\frac{1}{n} \sum_x [y \ln \hat{y} + (1 - y) \ln(1 - \hat{y})], \quad (4)$$

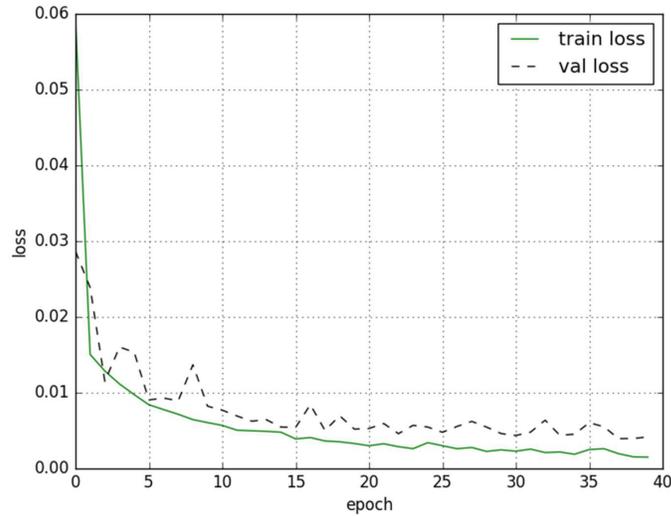


Fig. 5 The learning curve of one training process.

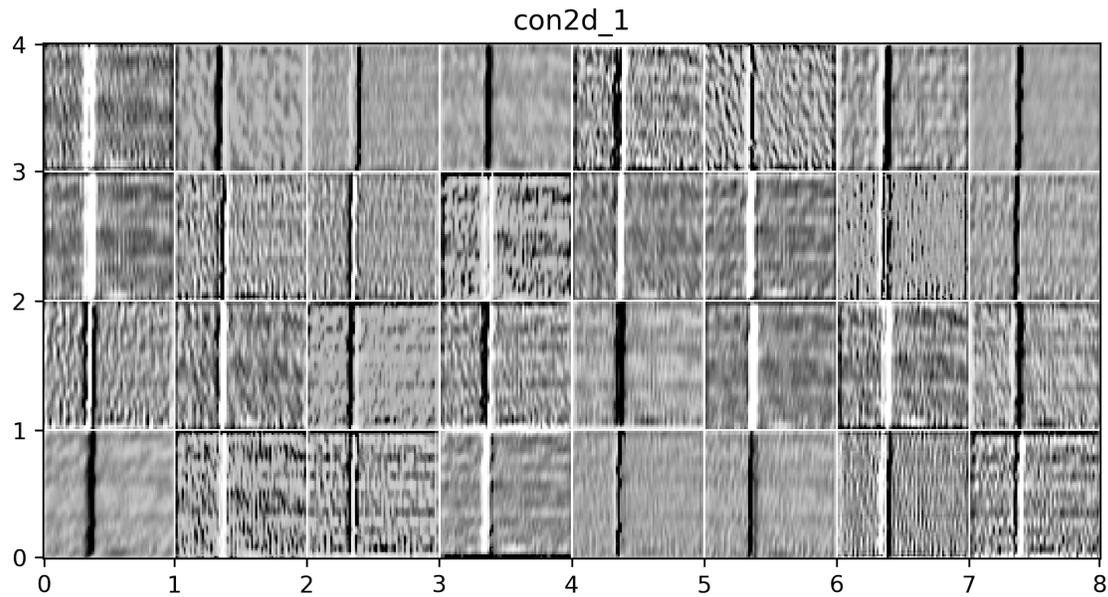


Fig. 6 Visualization of 32 feature maps from the first convolutional layer of the network. Each row has eight feature maps.

where the sum is over all training inputs x , n is the total number of training data, and y is the actual label with \hat{y} being the corresponding predicted label.

Our model is optimized using a first-order gradient-based optimization algorithm called Adam with default parameters (Kingma & Ba 2015) and a batch size of 512. In each epoch, 512 candidates are randomly selected as a batch from the training set to train the model and update parameters, which will repeat until all training data have been used. The implementation is based on Python 3.6 and Keras 2.2.4.

5 RESULTS

5.1 Evaluation Metric

To provide comprehensive assessments of model performance on this imbalanced dataset, we examined three metrics: recall, precision and F1. These metrics are defined as follows:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad (5)$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad (6)$$

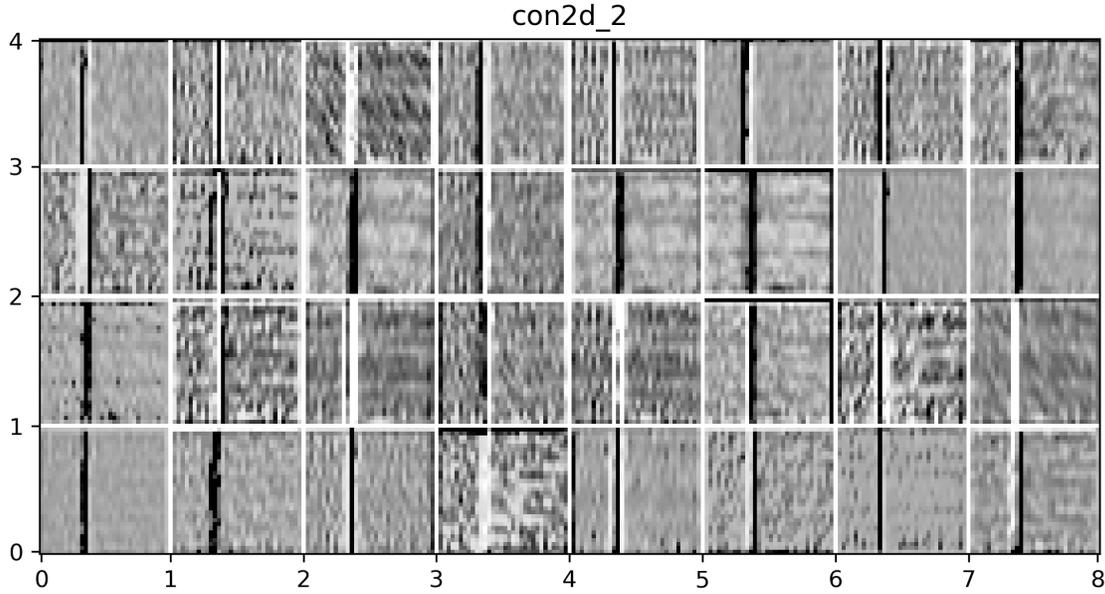


Fig. 7 Visualization of 32 feature maps from the second convolutional layer of the network. Each row has eight feature maps.

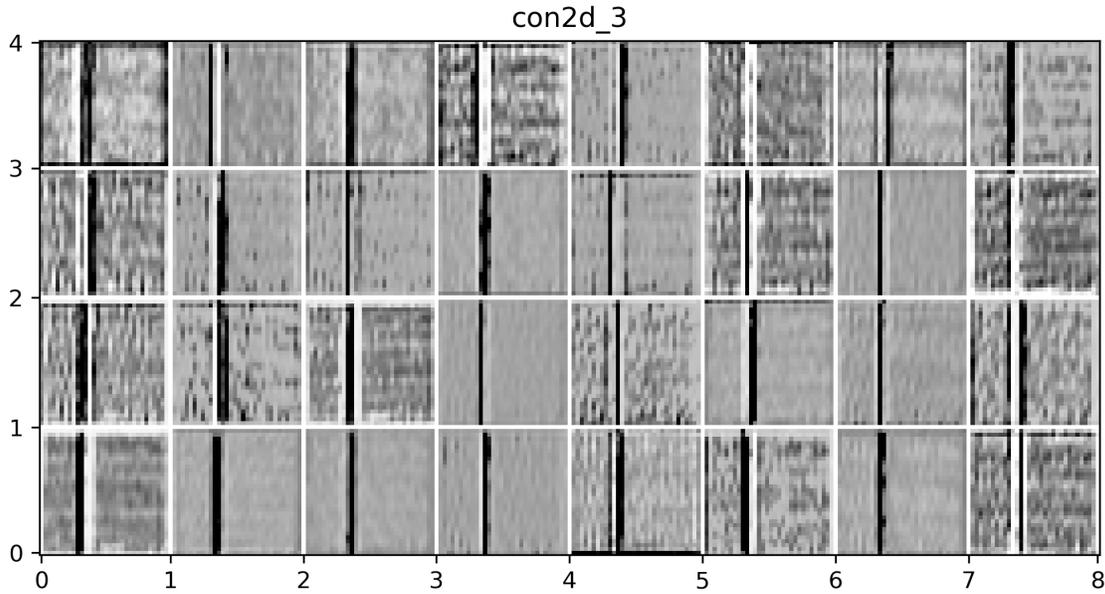


Fig. 8 Visualization of 32 feature maps from the third convolutional layer of the network. Each row has eight feature maps.

$$F1 = 2 \cdot \frac{\text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}}, \quad (7)$$

where TP is True Positive and TN is True Negative. These mean positive data (pulsar candidates) or negative data (RFI) being correctly recognized, respectively. FN is False Negative while FP is False Positive, and they represent the part of the positive data or negative data being incorrectly labeled, respectively. Recall indicates the fraction of pulsars correctly being recognized, and precision gives a measure of the fraction of pulsars in the predicted positive data. F1 is a harmonic mean of recall and precision. To identify

the largest possible fraction of pulsars while returning a minimal amount of mislabeled noise or RFI (Morello et al. 2014), a good model should provide a high recall as well as a high precision.

5.2 Experiment on HTRU 1

Before training, the dataset was divided randomly into three parts: training data (60%), validation data (20%) and testing data (20%). The first and second parts of the data were used to train the model while the testing part was used

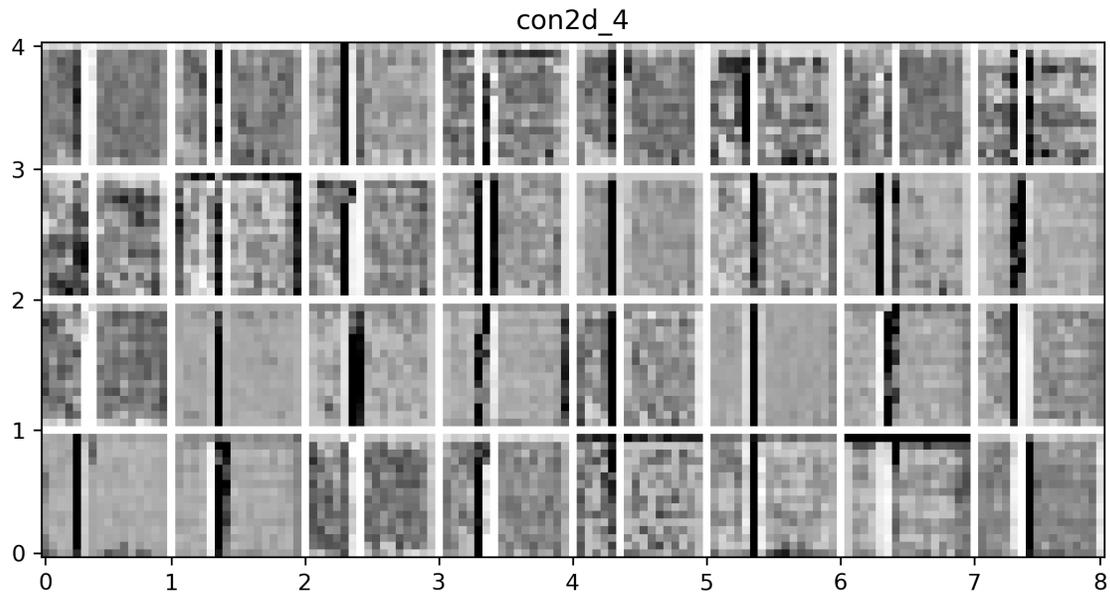


Fig. 9 Visualization of 32 feature maps from the fourth convolutional layer of the network. Each row has eight feature maps.

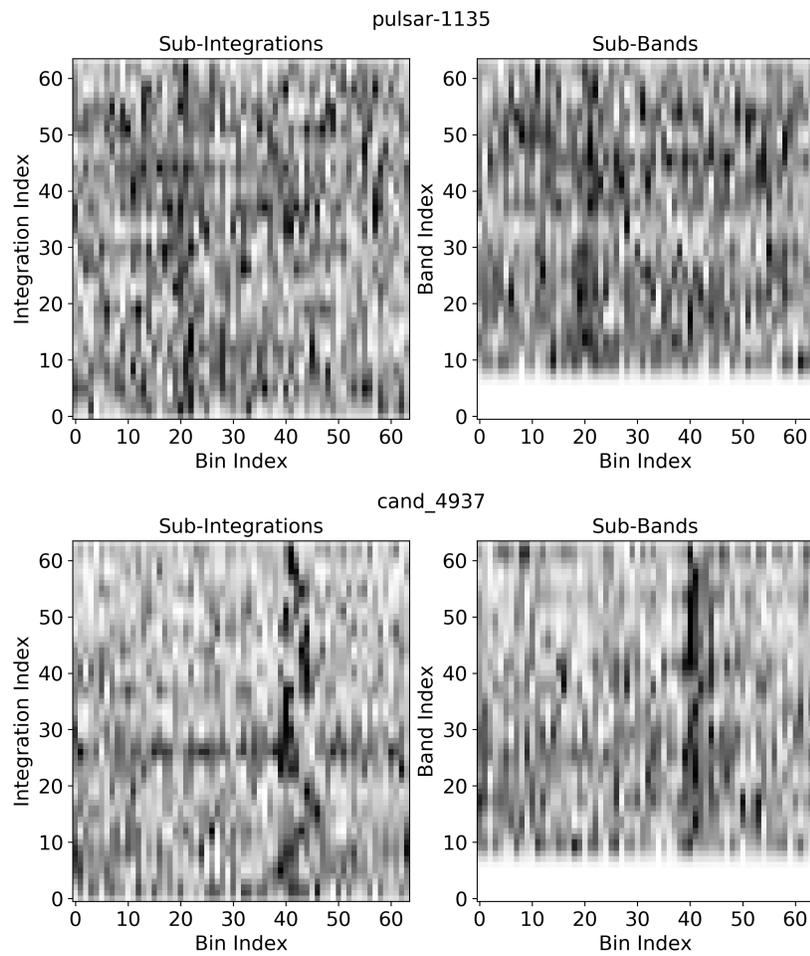


Fig. 10 Two mislabeled examples. Subplots in the first row are from pulsar_1135, and the others are from cand_4937. Their features in these subplots are not obvious.

to test the performance of the trained model. The proposed data augmentation in Section 3 only acted on the training data. With these augmentation methods, the number of pulsar sub-datasets for training was extended by a factor of 30. So, in each training process, the training set had 75 507 samples and the validation set employed for hyperparameter selection had 18 240 samples. More details are provided in Table 1.

The training operation was processed for 40 epochs, and the learning curve of one training process is shown in Figure 5. The entire training process was repeated 30 times, with different random data partitions, to obtain an average score for candidates.

Our results are listed in Table 2. On testing data, our CNN model without synthetic samples (only copying minority pulsar candidates 32 times when training) can achieve recall of 0.851 and precision of 0.848 (shown in Table 2 as DCNN-C). When the model was trained with data augmentation, our model showed excellent recall, precision and F1. It can achieve recall of 0.962 while precision is 0.963 (shown in Table 2 as DCNN-S). We think that the improvement is based on diversity of the synthetic samples. We also list some representative results of the data-driven method from Guo *et al.* (2017) as a contrast. In Guo *et al.* (2017), they tested the CNN model with the same architecture as in Zhu *et al.* (2014). Meanwhile, they introduced the DCGAN+SVM method, in which they used DCGAN to generate synthetic samples and extract deep features and a support vector machine (SVM) to produce better classifications with deep features. With the help of these synthetic samples, our CNN model has improved approximately 0.6% in recall and 1.2% in precision compared to the CNN in Guo *et al.* (2017). It can reach the level of using the DCGAN model. Though a Generative Adversarial Network (GAN) can generate new samples by adversarial training, it is difficult to train. In comparison, our synthetic method is simpler than using GAN to generate samples.

5.3 Analyses

To analyze features extracted by this model, intermediate convolution layer outputs of pulsar_0023 were visualized. Figure 6 to Figure 9 depict some feature maps of convolution layers 1 to 4. By comparing them, we can find that it extracts detailed features in shallow layers, and as the layers deepen, feature maps become more abstract and highlight the most important vertical stripe areas.

From the testing results, our model has shown the ability to classify pulsar candidates. We are more concerned with the generalization performance of the model and the classification performance of specific signal types,

such as weaker pulsar signals. We have analyzed the mis-labeled testing candidates and found that there are some candidates which are difficult to classify by input subplots. Figure 10 provides two examples of them, one is from pulsar_1135 while the other is from cand_4937. With some interference, cand_4937 is more pulsar-like than the signal of pulsar_1135. For these candidates, characteristics of these subplots are not obvious to identify. So if we want to make progress on accuracy, the model needs more resolved information.

6 CONCLUSIONS

In this work, we have proposed a deep CNN architecture for classification of pulsar candidates. By using plots as inputs, it is an end-to-end model and avoids potential bias of artificially designed features. To address the problem of imbalance, a simple and effective way was designed to synthesize more minority samples, which keep pulsar characteristics. With the aid of synthetic samples, the model has demonstrated the ability to classify pulsar candidates in the HTRU 1 dataset. It can extract features well and classify pulsar candidates reliably. The deep CNN has good application prospects in this field. To improve generalization performance further, we need to add more abundant and independent data or other valuable information (such as the DM-SNR curve).

References

- Alom, M. Z., Taha, T. M., Yakopcic, C., *et al.* 2018, arXiv:1803.01164
- Bates, S. D., Bailes, M., Barsdell, B. R., *et al.* 2012, MNRAS, 427, 1052
- Buda, M., Maki, A., & Mazurowski, M. A. 2018, Neural Networks, 106, 249
- Cordes, J. M., Kramer, M., Lazio, T. J. W., *et al.* 2004, New Astron. Rev., 48, 1413
- Der Maaten, L. V., & Hinton, G. E. 2008, Journal of Machine Learning Research, 9, 2579
- Devine, T. R., Goseva-Popstojanova, K., & McLaughlin, M. 2016, MNRAS, 459, 1519
- Eatough, R. P., Molkenthin, N., Kramer, M., *et al.* 2010, MNRAS, 407, 2443
- Faulkner, A. J., Stairs, I. H., Kramer, M., *et al.* 2004, MNRAS, 355, 147
- Ford, J. M. 2017, Pulsar Search Using Supervised Machine Learning, PhD thesis, Nova Southeastern University
- Goodfellow, I., Bengio, Y., & Courville, A. 2016, Deep Learning (MIT Press)
- Guo, P., Duan, F., Wang, P., Yao, Y., & Xin, X. 2017, arXiv:1711.10339

- He, H., & Garcia, E. A. 2009, *IEEE Transactions on Knowledge and Data Engineering*, 21, 1263
- He, K., Zhang, X., Ren, S., & Sun, J. 2016, in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 770
- Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. 2012, *Computer Science*, 3, 212
- Hobbs, G. B., Bailes, M., Bhat, N. D. R., et al. 2009, *PASA*, 26, 103
- Ioffe, S., & Szegedy, C. 2015, in *International Conference on Machine Learning*, 448
- Johnston, S., Lyne, A. G., Manchester, R. N., et al. 1992, *MNRAS*, 255, 401
- Keith, M. J., Eatough, R. P., Lyne, A. G., et al. 2009, *MNRAS*, 395, 837
- Keith, M. J., Jameson, A., van Straten, W., et al. 2010, *MNRAS*, 409, 619
- Kingma, D. P., & Ba, J. 2015, in *International Conference on Learning Representations*, 1
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. 2012, in *Advances in Neural Information Processing Systems*, 1097
- Lee, K. J., Stovall, K., Jenet, F. A., et al. 2013, *MNRAS*, 433, 688
- Lorimer, D. R., Lyne, A. G., & Camilo, F. 1998, *A&A*, 331, 1002
- Lyne, A. G., Burgay, M., Kramer, M., et al. 2004, *Science*, 303, 1153
- Lyon, R. J., Stappers, B. W., Cooper, S., Brooke, J. M., & Knowles, J. D. 2016, 459, arXiv:1603.05166
- Manchester, R. N., Lyne, A. G., Camilo, F., et al. 2001, *MNRAS*, 328, 17
- Morello, V., Barr, E. D., Bailes, M., et al. 2014, *MNRAS*, 443, 1651
- Nan, R., Li, D., Jin, C., et al. 2011, *International Journal of Modern Physics D*, 20, 989
- Nielsen, M. A. 2015, *Neural Networks and Deep Learning* (Determination Press)
- Radford, A., Metz, L., & Chintala, S. 2015, arXiv:1511.06434
- Sabour, S., Frosst, N., & Hinton, G. E. 2017, in *Advances in Neural Information Processing Systems*, 3856
- Sheikh, S. I., Pines, D. J., Ray, P. S., et al. 2006, *Journal of Guidance Control Dynamics*, 29, 49
- Simonyan, K., & Zisserman, A. 2015, in *International Conference on Learning Representations*
- Smits, R., Kramer, M., Stappers, B., et al. 2009, *A&A*, 493, 1161
- Srivastava, N., Hinton, G., Krizhevsky, A., et al. 2014, *Journal of Machine Learning Research*, 15, 1929
- Stokes, G. H., Segelstein, D. J., Taylor, J. H., & Dewey, R. J. 1986, *ApJ*, 311, 694
- Tan, C. M., Lyon, R. J., Stappers, B. W., et al. 2018, *MNRAS*, 474, 4571
- Zhu, W. W., Berndsen, A., Madsen, E. C., et al. 2014, *ApJ*, 781, 117