Improving pulsar search efficiency in next-generation pulsar surveys with artificial intelligence

Qiuyang Fu[1,2⋆] Mengyao Xue[1†] Weiwei Zhu[1,3] N. D. R. Bhat[4] Kaichao Wu[5] Zihan Zhang[5,6] B. W. Meyers[7,4] Chia Min Tan[4] Youling Yue[1] Jiarui Niu[1] Lingqi Meng[1,2] Ziwei Wu[1] Ziyao Fang[1] Yukai Zhou[8] Jiawei Jin[1,2]

[1]National Astronomical Observatories, Chinese Academy of Sciences, 20A Datun Road, Chaoyang District, Beijing 100101, China
[2]School of Astronomy and Space Science, University of Chinese Academy of Sciences, Beijing, 100049, China
[3]Institute for Frontiers in Astronomy and Astrophysics, Beijing Normal University, Beijing 102206, China
[4]International Centre for Radio Astronomy Research (ICRAR), Curtin University, Bentley, WA 6102, Australia
[5]Computer Network Information Center, Chinese Academy of Sciences,
CAS Informatization Plaza No.2 Dong Sheng Nan Lu, Haidian District, Beijing 100083, China
[6]School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing, 100049, China
[7]Australian SKA Regional Centre (AusSRC), Curtin University, Bentley, WA 6102, Australia
[8]Department of Astronomy, School of Physics, Peking University, Beijing 100871, China

Accepted XXX. Received YYY; in original form ZZZ

ABSTRACT

Pulsar searching with next-generation radio telescopes requires efficiently sifting through millions of candidates generated by search pipelines to identify the most promising ones. This challenge has motivated the utilization of Artificial Intelligence (AI)-based tools. In this work, we explore an optimized pulsar search pipeline that utilizes deep learning to sift "snapshot" candidates generated by folding de-dispersed time series data. This approach significantly accelerates the search process by reducing the time spent on the folding step. We also developed a script to generate simulated pulsars for benchmarking and model fine-tuning. The benchmark analysis used the NGC 5904 globular cluster data and simulated pulsar data, showing that our pipeline reduces candidate folding time by a factor of ∼10 and achieves 100% recall by recovering all known detectable pulsars in the restricted parameter space. We also tested the speed-up using data of known pulsars from a single observation in the Southern-sky MWA Rapid Two-metre (SMART) survey, achieving a conservatively estimated speed-up factor of 60 in the folding step over a large parameter space. We tested the model's ability to classify pulsar candidates using real data collected from the FAST, GBT, MWA, Arecibo, and Parkes, demonstrating that our method can be generalized to different telescopes. The results show that the optimized pipeline identifies pulsars with an accuracy of 0.983 and a recall of 0.9844 on the real dataset. This approach can be used to improve the processing efficiency for the SMART and is also relevant for future SKA pulsar surveys.

Key words: pulsars: general – methods: data analysis

1 INTRODUCTION

Pulsars are rapidly spinning neutron stars with strong gravitational and magnetic fields. The discovery of the first pulsar, PSR 1919+21, marked the advent of pulsar astronomy (Hewish et al. 1968). The first binary pulsar system, PSR B1913+16, provided the first evidence of gravitational waves (Hulse & Taylor 1975). The discovery of radio pulsars relies on telescopes to capture signals originating from beyond our solar system, which have traversed the interstellar medium (ISM). The radio waves interact with free electrons in the ISM during propagation, leading to dispersion, whereby low-frequency signals arrive later than high-frequency signals. This effect is quantified by the dispersion measure (DM), a value directly proportional to the integrated column density of free electrons between the pulsar and observer (Lorimer & Kramer 2005). Additionally, pulsar observations need to account for other propagation effects of the ISM, such as scattering (i.e., pulse broadening) (Jing et al. 2025) and Faraday rotation. These effects are significant at low frequencies ($\lesssim$300 MHz), near the Galactic center, or along the Galactic plane (Rickett 1990).

The vast majority of pulsars, particularly millisecond pulsars, are only moderately luminous and as a result single pulses are difficult to observe directly (Taylor 1991). To enhance the signal-to-noise ratio (SNR), a common technique is to fold the observational data according to the pulsar's rotational period. By aligning and summing multiple pulses, this folding process amplifies the periodic signal

⋆ E-mail:fuqy@bao.ac.cn
† mengyaoxue@nao.cas.cn

above the noise background. This approach is essential for the discovery of a substantial fraction of pulsars.

The number of candidates produced by modern pulsar surveys exceeds the capacity for manual inspection. In 1977, the second Molonglo Survey identified around 2,500 possible candidates across the sky south of declination +20° (Manchester et al. 1978). Three decades later, the Northern part of the High Time Resolution Universe (HTRU) survey found over 80 million candidates across the entire region observed above +30° declination [1] (Barr et al. 2013). Upcoming surveys with the Square Kilometre Array (SKA) will generate even larger candidate sets, making manual inspection intractable. Therefore, it is important to develop methods to assist in filtering candidates.

Early approaches focused on selecting candidate features to build sorting strategies for filtering millions of candidates (Keith et al. 2009). Later, Lee et al. (2013) utilized statistical methods, using six factors from pulsar candidates to determine their identification scores. Meanwhile, machine learning (ML) is increasingly used to screen pulsar candidates. As early as 2010, Eatough et al. (2010) applied artificial neural network (ANN) algorithms to score candidates for pulsar identification. Subsequently, Zhu et al. (2014) proposed a classifier combining a convolutional neural network (CNN), an ANN, and a support vector machine (SVM). Several ML classification models have already been applied in pulsar surveys. For example, an ensemble candidate classifier (Tan et al. 2018) was used in the LOFAR Tied-Array All-Sky Survey (LOTAAS) (Sanidas et al. 2019), and the first-pass processing in the Southern-sky MWA Rapid Two-metre (SMART) survey (Bhat et al. 2023) adopted modest refinements of the LOTAAS classifier. A model incorporating Semi-supervised generative adversarial network (SGAN) (Balakrishnan et al. 2021) has been applied for candidate classification in the HTRU survey with the Parkes telescope (Murriyang). A CoAtNet-based classification model (Cai et al. 2023) has been adopted in the Galactic Plane Pulsar Snapshot (GPPS) survey (Han et al. 2021) with the Five-hundred-meter Aperture Spherical radio Telescope (FAST, Jiang et al. 2020).

The conventional pulsar candidate classifiers evaluate each candidate folded from the full data. Full data is time-sampled and divided into multi-frequency channels in "SIGPROC filterbank format" [2](Lorimer 2011) or "PSRFITS format" [3](Hotan et al. 2004) files. As the observation duration increases, folding time increasingly dominates the pulsar search runtime. So we fold the de-dispersed time series data across stacked frequency channels to reduce computational cost. Our method generates "snapshot" candidates by folding time-series data according to periods, filtering out non-pulsar-like candidates early in the time domain. Then, we fold the full data to obtain complete time and frequency information for further assessment. Consequently, there is a demand for training new feature classifiers specifically adapted to the time domain pulsar features of the candidates.

This paper demonstrates that our pipeline using artificial intelligence (AI) models can significantly reduce the time spent on the folding step, a critical stage in the pulsar search. While AI is mainly used for classifying pulsar candidates, its application in accelerating data processing enhances overall search efficiency. As pulsar surveys produce amounts of data proportional to observation time, current modern low-frequency array surveys such as the LOTAAS, a one-hour integration pulsar survey covering the entire northern sky (Sanidas et al. 2019), and the SMART survey, conducted with the Murchison Widefield Array (MWA, Tingay et al. 2013) to cover the sky south of 30° in declination with 80-minute observation times (Bhat et al. 2023), are already facing computational challenges from massive data rates. Future surveys, such as upcoming projects with the SKA, will further amplify these demands, making speed-up of the pulsar search process essential. AI solutions tailored to folding steps can reduce processing burdens.

Table 1. Pulsar blind search time breakdown with 100 adjacent de-dispersion trials. It presents the number of DM and period combinations generated for 10-, 20-, and 30-minute observation data, as well as the time proportion consumed by each step for these data. The total folding time correlates positively with the number of potential DM and period combinations, and it also increases with longer observations.

Step	10-min obs.	20-min obs.	30-min obs.
DM–Period combinations	51	81	217
De-dispersion & FFT	1.6%	1.1%	0.7%
Accelsearch	17.1%	14.3%	5.9%
Fold	81.3%	84.6%	93.4%

In this paper, we present an AI-accelerated pulsar Fast Fourier Transform (FFT) search pipeline that minimizes folding time to speed up pulsar searches. In Section 2, we introduce an optimized pipeline for the pulsar search, the utilized model architecture, and the applied preprocessing methods for the candidates. In Section 3, we describe the data used in this work, including both real and simulated datasets. In Section 4, we evaluate both the speed-up of the optimized pipeline and the classification performance of the model. Finally, in Section 5, we summarize the method and outline its potential for future pulsar surveys.

2 METHODS

2.1 Pipeline strategy for speed-up via optimized folding step

The ATNF Pulsar Catalogue (Manchester et al. 2005) in version 2.6.3 [4] reports over 4,343 pulsars, most of them discovered from pulsar surveys. The PulsaR Exploration and Search TOolkit (PRESTO)[5](Ransom 2011), has helped to discover over a thousand pulsars. Building on PRESTO's success, we explore an AI-accelerated, FFT-based search pipeline for speeding up the pulsar search process as shown in Figure 1.

The blind search for pulsars based on PRESTO consists of four stages, beginning with the identification and removal of radio frequency interference (RFI) using PRESTO's "**rfifind**" tool to create a channel mask file. Second, a range of trial DM values was used to generate one-dimensional (1-D) de-dispersed time series for each trial, thereby correcting for unknown dispersion time delays. Each trial generates a separate "**.dat**" file. Third, the corrected data undergo a Fourier transformation to identify pulsar periods by finding peaks in the power spectrum, and then harmonic summing and peak detection produce candidate parameters for folding. In the final stage, the data are folded for each DM-period combination, generating diagnostic plots to validate potential pulsar candidates. This FFT-based

[1] https://www.jb.man.ac.uk/pulsar/surveys.html
[2] https://sigproc.sourceforge.net/
[3] https://psrchive.sourceforge.net/
[4] https://www.atnf.csiro.au/research/pulsar/psrcat/
[5] https://github.com/scottransom/presto

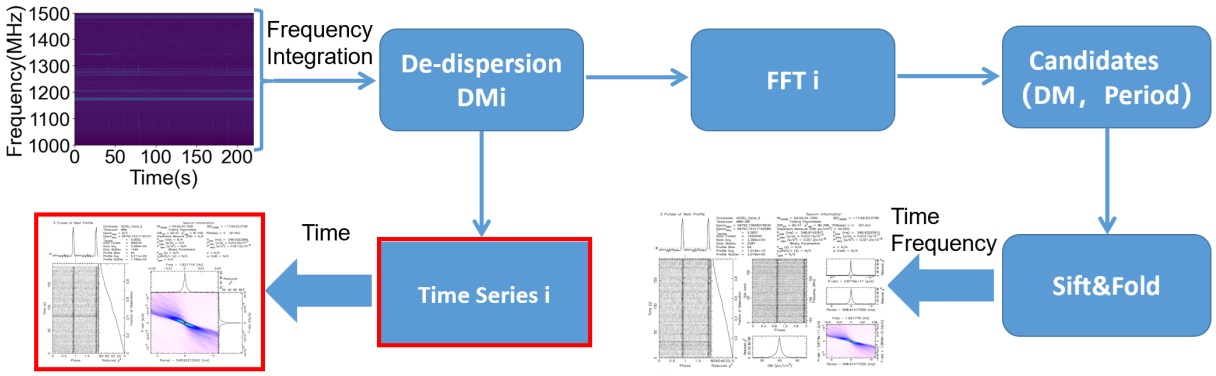

Figure 1. In the FFT-based pulsar search, after trying all possible DM values to remove dispersion delay and applying FFT to get potential periods (Cooley & Tukey 1965), we obtain a series of candidate DM-period combinations. These candidates can then be folded using the full multi-frequency data or the corresponding de-dispersed time series data.

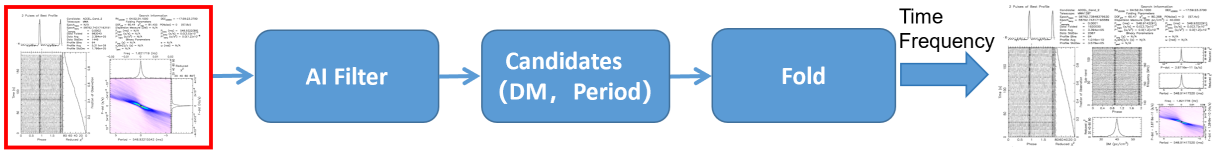

Figure 2. We adopt a three-stage approach to reduce folding time. First, we fold the time series data to get initial pulsar candidates. Next, we filter these candidates using an AI classifier based on time-domain features. Finally, we fold the remaining candidates using the full data to confirm their authenticity with additional frequency information.

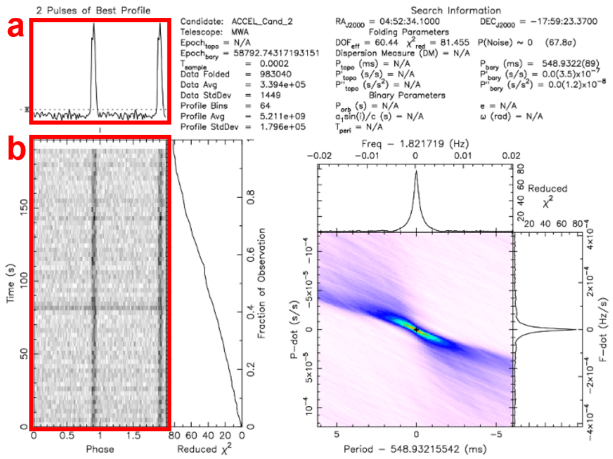

Figure 3. The candidate plot exhibits two features: feature "a" corresponds to the average pulse profile obtained by integrating the folded data along the time axis, and feature "b" is a time-phase diagram, where the x-axis shows two phases and the y-axis indicates the integration time.

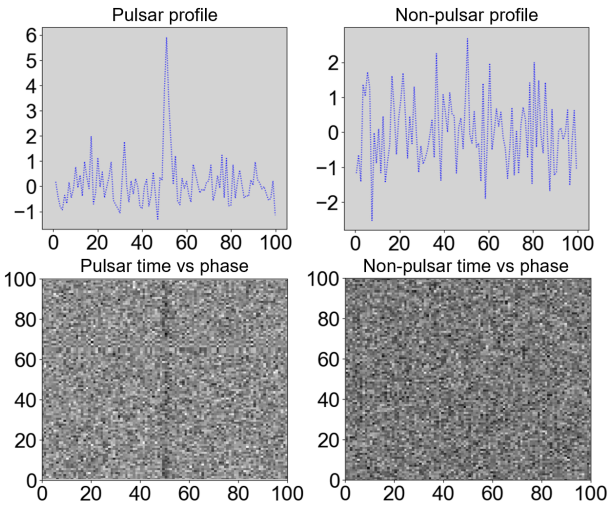

Figure 4. Top row: average pulse profiles (feature "a"), where the left panel shows a real pulsar profile characterized by sharp peaks at certain phases due to inherent periodicity and the averaging out of noise, and the right panel shows the profile of a non-pulsar candidate. Bottom row: time-phase diagrams (feature "b"), where the left panel illustrates a real pulsar, characterized by periodic signals consistently aligned at the same phase over a time span, whereas the right panel depicts a non-pulsar candidate lacking such phase coherence.

pulsar search involves a two-dimensional (2-D) grid of periods and DM values. Folding with potential combinations enhances the SNR, enabling the detection of weak pulsar signals. While folding full data at specific DM values and periods yields valuable time and frequency information, the computational cost of processing large data with numerous parameter combinations is high.

To quantitatively estimate the proportion of computation time that the folding steps account for in the overall pulsar search process, we processed 30 minutes of FAST observation data (137 GB total). We divided the data into 10-, 20-, and 30-minute segments to analyze the relationship between folding time and observation length, using a single computing node equipped with an Intel Xeon Silver 4314 processor operating at 2.40 GHz. The data contains only total intensity, with a sampling time of 49.152 µs, and 4096 frequency channels. All pulsar search steps run on a single CPU core. We performed 100 adjacent DM trials using a single de-dispersion operation, at nearly the same cost as a single trial. None of the trials contained known pulsars. We applied FFT and acceleration searches using default parameters (zmax = 200, numharm = 8, sigma = 2.0) to each resulting

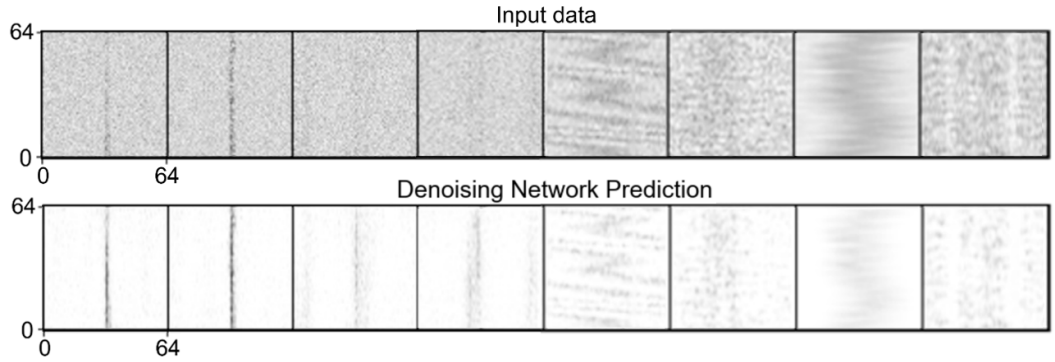

Figure 5. We input the time-phase features into a Denoising Autoencoder (DAE). Each instance's x-axis represents the phase bins, while the y-axis corresponds to the time intervals. The example feature arrays are resized to dimensions of 64 by 64. The first four instances are positive samples, and the next four are negative. Using the denoising mechanism of the DAE, it effectively suppresses noise and highlights features.

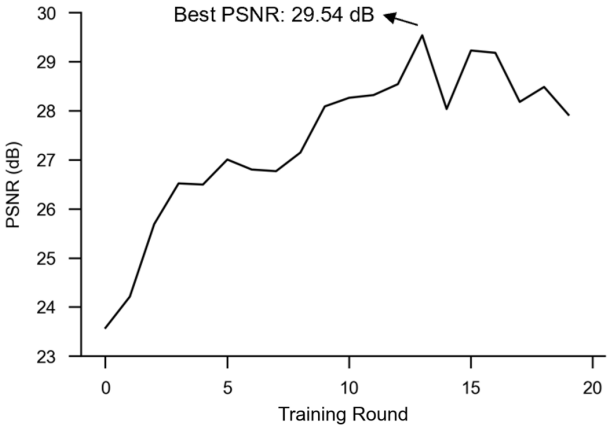

Figure 6. The variation of PSNR values calculated from 400 test samples over the training rounds. PSNR quantifies the difference between the original and denoised images, where higher values indicate better denoising and reconstruction quality.

time series, filtered the results with ACCEL_sift.py, and folded the full data over all DM–period combinations without fine searches. As a result, the Accelsearch and Fold steps dominated the overall computation time. The processing time for each step is detailed in Table 1. The folding time for potential candidates scales with the number of DM and period combinations, and increases further with longer observation duration. The result shows that during the pulsar blind search process, the folding step constitutes a substantial fraction of the computational time allocation. It presents challenges for large-scale, long-duration pulsar surveys due to the computational cost caused by folding numerous DM and period combinations on full data.

We explore a search strategy to address computational bottlenecks in the folding stage by filtering "snapshot" candidates generated from folding 1-D time series data. This approach substantially decreases the number of full data foldings, thus reducing the usage of computational resources, as shown in Figure 2. Specifically, after de-dispersion with PRESTO, we obtain 1-D de-dispersed time series data by integrating frequencies. The time series data occupies a space equal to the full data size divided by the number of frequency channels. We fold this 1-D data at the corresponding DM values using the candidate period. The folded candidate diagnostic plots solely contain time information, as depicted in Figure 3. The feature labeled "a" represents the pulse profile, illustrating the radiation characteristics of a pulsar in a 1-D array. Feature "b" is referred to as the time-phase diagram, representing signals that consistently occur at the same phase across an intrinsic time span in a 2-D array.

Current approaches that integrate ML classifiers into PRESTO-based pipelines such as the LOTAAS search pipeline and the first-pass SMART pipeline. These methods reduce the burden of manually inspecting large numbers of candidates by applying classifiers to folded candidates containing time and frequency information. However, they do not address the computational challenges faced by modern pulsar surveys. Our method first uses candidate parameters to fold 1-D data summed over frequency, then applies a classification model to classify pulsar features, filtering parameters without pulse characteristics in the time domain. This approach allows efficient pre-filtering before folding the full data, reducing the number of folding operations and thereby decreasing the overall processing time.

2.2 Preprocessing of "snapshots" candidates

Each candidate generated by PRESTO includes an image and a "**.pfd**" file with candidate information. The features "a" and "b" in the image, as shown in Figure 3, are used as identifying characteristics. The array size of these features in the information file is determined by the "-n" parameter used in the folding command, which defines the number of bins (data points) in the profile phase. For instance, the x-axis of the profile in Figure 4 shows the number of bins within one folded period, which by default corresponds to the number of time samples per period. Consequently, the choice of bin size influences the resulting SNR. However, as neural networks typically require inputs with consistent dimensions, standardizing the data shape becomes necessary. Earlier pulsar classification models resized the input data to a specified size (Zhu et al. 2014; Balakrishnan et al. 2021; Cai et al. 2023). It should be noted that this resizing approach can impact the overall performance of the pulsar classifier. Thus, we resize feature arrays to 64, 96, and 128 bins using downsampling and interpolation, creating three distinct samples for a multi-input fusion model. This approach minimizes the impact on classification performance caused by resizing feature arrays to a uniform size. In addition, it increases the width of the network, enabling the model to capture features across multiple scales.

After extracting data from the "**.pfd**" file, the existence of weak candidates increases the risk of features being overshadowed by noise. We use a Denoising Autoencoder (DAE) (Vincent et al. 2008) for denoising in the data preprocessing stage, focusing on time-phase

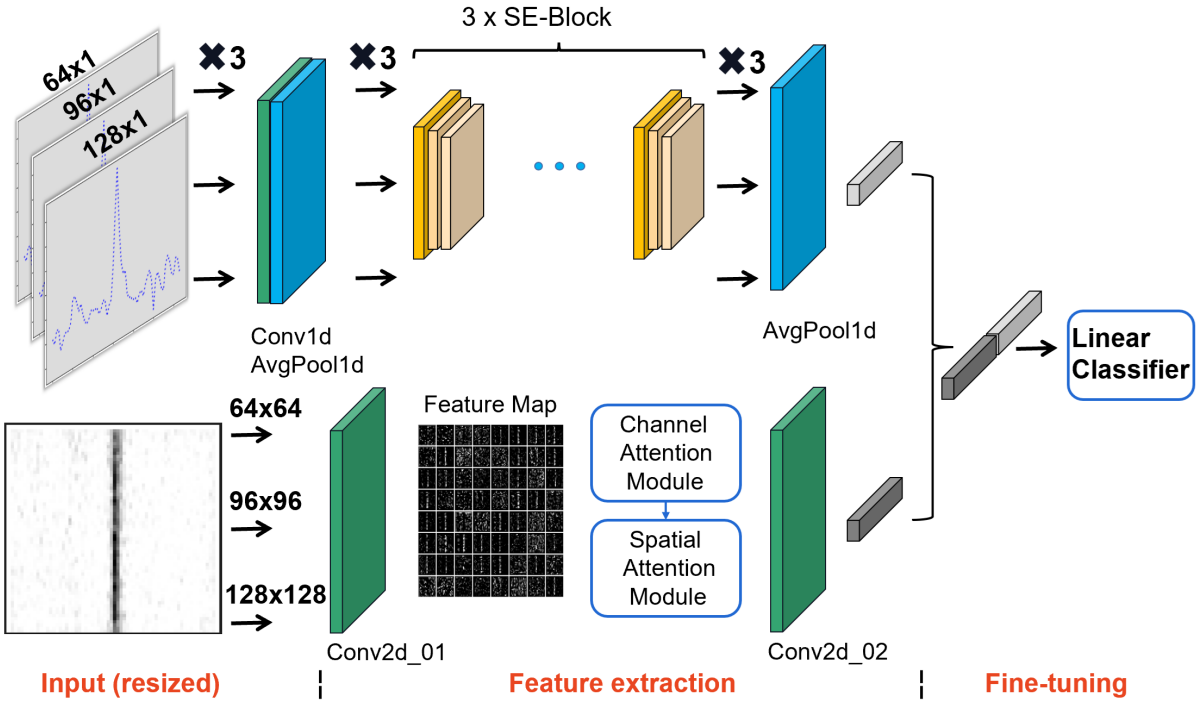

Figure 7. An illustration of the model used for sifting "snapshots" candidates. The first step is to resize each candidate feature array to fixed lengths of 64, 96, and 128 bins, respectively, which enables better adaptation to varying candidate data sizes. The resized data is then input into the corresponding feature extraction modules. The extracted features are flattened and fed into a linear classifier to identify pulsar characteristics. When applying the model to candidates from other telescopes outside the training set, a fine-tuning step can be introduced to improve performance.

features (bottom two panels of Figure 4). The DAE creates noisy inputs by adding random noise to clean data and is trained to output a reconstruction of the original clean samples. Its objective is to minimize the difference between the reconstructed output and the clean input, making the model capable of denoising and recovering structural features. We implement the DAE with a U-Net (Ronneberger et al. 2015), adapted for image-to-image translation, to remove noise. In this setup, U-Net serves as the network architecture of the DAE, which is effective for denoising because its encoder-decoder architecture uses skip connections to pass high-resolution features from encoder to decoder, enabling the model to suppress noise while preserving fine structural details. We selected 4,000 positive pulsar samples as the original images and adopted a training strategy that adds increasing levels of Gaussian noise with more rounds (Ho et al. 2020). This generates noisy positive samples for the model to learn denoising. Before training, we randomly sampled 400 examples for evaluation. We used the peak signal-to-noise ratio (PSNR) between the original and denoised images to assess reconstruction quality (Jähne 2005). This process involves training the U-Net model on a dataset of noisy original images, using a loss function that penalizes the discrepancy between the denoised output and the original image. See appendix A for the definitions of PSNR and the loss function. The U-Net architecture, consisting of contracting and expansive paths, captures context and localizes features precisely, effectively denoising images by preserving structures and suppressing noise artifacts. Table A1 summarizes the model we used. After training, the U-Net model effectively transforms noisy inputs into relatively clean ones. This process enhances the quality of time-phase features, as shown in Figure 5. And the PSNR results of the denoising model are shown in Figure 6. The results show that the denoising model enhances the weak features of real pulsar samples embedded in heavy noise.

2.3 AI model for sifting "snapshots" candidates

We used two features, the pulse profile and the time-phase ("a" and "b" in Figure 3), to score the pulsar candidates. As introduced in the previous Section 2.2, we preprocess the candidate data by resizing it to various sizes and denoising the 2-D data. We then extract features from the profile using a one-dimensional residual network (1-D ResNet). ResNet is designed to pass information forward more effectively by adding shortcut connections between layers, which prevents the loss of important information when the model becomes deep (He et al. 2016). Our model requires sufficient depth to learn the characteristics of the profile features, while avoiding the risk that deeper models extract more generalized patterns and overlook important details. So we use a simplified, customized ResNet model to ensure effective learning of the distinguishing characteristics of the profile features. We further incorporate a Squeeze-and-Excitation attention (SE attention) mechanism, which helps the model focus on important parts of the profile features. It works by adaptively reweighting different convolutional channels, each corresponding to a feature map. Each channel acts as a specialized filter that highlights a specific aspect of profile features, allowing the model to emphasize the most informative representations (Hu et al. 2018). For time-phase features, we use CNNs with the Convolutional Block Attention Module (CBAM), which enables the model to focus on the most informative spatial and channel regions (Woo et al. 2018). It focuses on important positions in the time-phase feature map for spatial information and on the most relevant feature channels for channel information. Following feature extraction, a fully connected layer with each input connected to all outputs is used to classify the extracted features. It produces two logits, which are unnormalized scores representing the model's raw predictions for the binary classes. These logits are converted into a probability distribution through a Softmax activa-

tion, ensuring non-negative values and normalization. The model is trained by minimizing the cross-entropy loss, which quantifies the difference between the predicted probabilities and the true labels. For a single sample, the loss is defined as:

$$L = -\sum_{i=1}^{2} y_i \log(p_i) \tag{1}$$

where y_i is the one-hot encoded label for class i in the binary classification, with $[y_0, y_1] = [1, 0]$ for the non-pulsar class and $[y_0, y_1] = [0, 1]$ for the pulsar class, and p_i is the predicted probability for class i obtained by the Softmax function. This hybrid model, as shown in Figure 7, handles multi-size inputs and employs different neural networks for each feature, thereby enhancing feature extraction capability. We implemented the hybrid model with Tinygrad [6], a lightweight deep learning framework that supports various hardware accelerators. The details of this model are presented in Table A2. In the following paragraphs, we introduce each module and provide detailed descriptions of how they are used.

CNNs are deep learning models excelling in computer vision tasks like image classification and object detection (LeCun et al. 1998; Krizhevsky et al. 2012). They use convolutional layers to extract features and fully connected layers to make predictions. They include pooling layers to reduce computation and memory usage, and activation functions like ReLU (Nair & Hinton 2010) and sigmoid to introduce nonlinearity. Techniques such as dropout and batch normalization (Ioffe & Szegedy 2015) further enhance performance by preventing overfitting and improving training stability. Inspired by GoogLeNet's multi-scale approach (Szegedy et al. 2015), our architecture replaces parallel multi-kernel convolutions and pooling with multi-sized input resizing. This enables simultaneous capture of multi-scale features within a single layer, broadening network width to enhance model performance.

ResNet is a deep neural network architecture that uses residual connections to train deeper networks effectively and address vanishing gradients (He et al. 2016). Our bottleneck residual block, consisting of three convolutional layers and a shortcut connection, facilitates this process. We use a 1-D ResNet with SE attention to classify pulse profile features. The SE attention mechanism helps the network focus on the most relevant features, improving classification accuracy. The profile features are resized to dimensions of 64, 96, and 128 bins. The network processes these resized features to extract convolutional features, which are flattened, concatenated, and input into a linear classifier for pulsar classification. The ResNet architecture allows capturing discriminative features from diagnostic plots at multiple levels, aiming for better classification performance.

CBAM enhances CNN representational power by integrating channel and spatial attention mechanisms (Woo et al. 2018). The channel attention recalibrates feature responses to highlight informative channels, while spatial attention adjusts responses based on spatial relationships to focus on important regions. We preprocess candidate time-phase features using a DAE for denoising, then resize the data to 64, 96, and 128 bins. Each resized array is then processed through convolutional layers to extract features. The attention module applies channel attention before spatial attention, allowing the spatial mechanism to act on more informative channel features. These attention mechanisms are applied to the feature maps to compute weights that emphasize important regions within them. Then these features are flattened, concatenated, and input into a linear classifier.

Fine-tuning adjusts the parameters of a pre-trained model using a task-specific dataset to better adapt it to the target task. In our work, we fine-tuned a pre-trained neural network on a small dataset to improve classification for pulsar candidates from the MWA. We used a pre-trained network for feature extraction, taking advantage of its ability to capture hierarchical data representations. Due to the diverse data distributions of pulsar candidates from different telescopes, we provided an Application Programming Interface (API) to incorporate additional data into the fine-tuning training. By freezing the feature extraction module and fine-tuning only the final linear classifier (Yosinski et al. 2014), we aimed to improve its performance in identifying pulsar candidates from telescopes outside the training data.

[6] https://github.com/tinygrad/tinygrad

Table 2. Candidates dataset for training and validation

Telescope	Samples Positive : Negative (1:1)
FAST	3000
Parkes	5000
Arecibo	4000

3 DATA

3.1 Observational data for pipeline benchmarking

We have compiled comprehensive datasets of pulsar candidates collected from telescopes operating at different frequencies and under varying radio environments. We train and validate the model with an expert-reviewed dataset of real pulsar samples from three telescopes (FAST, Parkes, and Arecibo). We assess the generalizability of the trained model using candidate datasets collected from the MWA and the Green Bank Telescope (GBT). Additionally, we demonstrate the speed-up of our search strategy using globular cluster data from FAST, data from a single observation of the SMART survey, and simulated data containing multiple pulsars with specified ranges of DM values.

The data from FAST, Parkes, and Arecibo were used as training and validation datasets (Table 2), while observations from the MWA and GBT served as novel data environments to evaluate model generalizability. To determine whether a folded candidate is a pulsar, we perform feature identification on the two key characteristics in the candidate diagnostic plot. This supervised deep learning task requires manually annotated data. Experts inspect each candidate and label it as 1 for a true pulsar and 0 for a non-pulsar. We note that our model's reliance on time-domain features makes it susceptible to performance degradation from pulsar-like negative candidates (e.g., narrow-band RFIs). During dataset construction, we implemented manual cleansing of the training set to eliminate such RFIs.

We assembled a dataset of pulsar candidates, including both positive and negative samples, generated from five pulsar surveys processed using PRESTO. The FAST candidates were sourced from the Commensal Radio Astronomy FAST Survey (CRAFTS) (Li et al. 2018), the Parkes candidates were collected from the High Time Resolution Universe South Low Latitude (HTRU-S LowLat) pulsar survey (Keith et al. 2010), and the Arecibo candidates from the Pulsar Arecibo L-band Feed Array (PALFA) survey (Cordes et al. 2006). In addition, we collected Green Bank North Celestial Cap (GBNCC) survey candidates from the GBT (Stovall et al. 2014), and incorporated pulsar candidates from the SMART survey, conducted with the MWA.

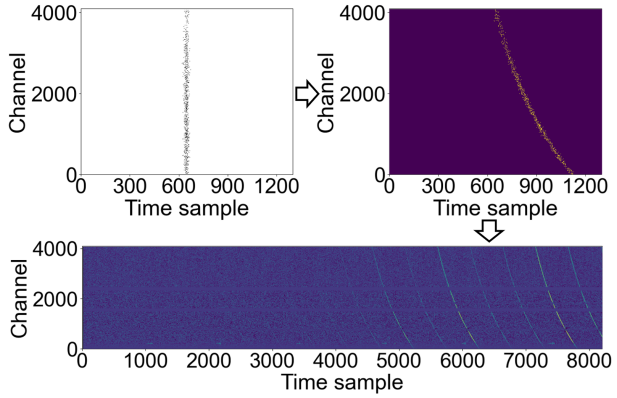

Figure 8. Simulated data on pulsars were generated using our developed script that injects signals with dispersion delays into observational data. The process of simulating pulsars on FAST observational data is shown in the figure. The x-axis represents time samples, and the y-axis represents frequency channels. The simulation of pulsars involves generating single pulses with different intensities, applying time delays corresponding to the specified DM, and adding these pulses to the appropriate time samples.

We validated the performance of our method using real globular cluster data. Pulsar searches typically require scanning a wide range of celestial coordinates to discover new pulsars. However, within globular clusters, the detection of multiple pulsars with similar DM values becomes possible. This capability arises due to the dense stellar environment inherent to globular clusters, which provides favorable conditions for the formation and retention of pulsars, thereby facilitating the detection of multiple pulsars exhibiting similar DM values (Ransom 2007). We validate the feasibility of our speed-up strategy by evaluating whether all known globular cluster pulsars' "snapshot" candidates are retained as top-ranked candidates within narrow DM ranges.

We also evaluated the speed-up ratio using data of all detectable known pulsars from an 80-minute single observation of the SMART survey. Our test makes use of beamformed data on 43 known pulsars from observation ID 1267459328, collected during Phase II MWA's compact configuration (Wayth et al. 2018). We performed a large-scale blind search on these known pulsar data using PRESTO. After RFI removal, we applied de-dispersion processing over a DM range of 0-500 pc cm^{-3} with a total of 6,624 trials. Then, for each trial, we apply the FFT and remove red noise. Using the parameters "-zmax 100, -numharm 16 and -sigma 2.0", we conduct an acceleration search followed by filtering with ACCEL_sift.py to obtain a series of potential pulsar candidates. The number of candidates generated from pointings that include known pulsars within a single observation is typically higher than that from pointings without pulsars. Therefore, if the observation field is covered with a sufficient number of pointings, the average number of candidates per pointing is unlikely to exceed that of the pointings with known pulsars. This experiment thus provides a conservative estimate of the potential speed-up achievable by our method for large-scale pulsar survey data processing. This processing pipeline allows us to evaluate the speed-up of our method within a parameter space close to that of real pulsar surveys. We applied our model to classify the "snapshot" candidates and selected a small subset for folding using the full data. The number of candidates in this subset serves as an indicator of the speed-up of our method.

3.2 Simulated data for model performance optimization

We developed a pulsar signal simulation script that periodically injects single pulses into PSRFITS-format files. The simulated pulse parameters are extracted from real observed single pulses, and we simulate the dispersion effect of pulsars by adding time delays to the signals. These simulated pulsars support analysis of the speed-up of our pipeline and help fine-tune our pre-trained models to better adapt to the data distribution of newly constructed telescopes.

Currently, there are several software packages that can simulate pulsar data. One such software is SIGPROC (Lorimer 2011), which is capable of generating filterbank format files for simulated pulsars with specified DM values and periods. One drawback of SIGPROC is its inability to generate data in the PSRFITS format. Furthermore, the method of generating Gaussian noise is inadequate for accurately simulating complex observational environments. Luo et al. (2022) developed an inventive software package for simulating channelized, high-time resolution radio telescope data streams. This versatile tool enables researchers to generate synthetic data tailored to specific telescope parameters and observing systems, while also supporting the injection of simulated signals into real datasets. This software lacks GPU support and cannot allow the injection of multiple pulsars in a single processing run, which makes it not well-suited for large-scale pulsar simulation in our work.

Considering the aforementioned factors, we have developed a script using PyCUDA [7], specifically designed to use NVIDIA GPUs for speed-up. This script enables efficient injection of simulated pulses into observational data. As shown in Figure 8, we extract multiple sets of parameters for the 2-D Gaussian function from real single pulses, then randomly sample one parameter set for each injected pulse and optimize it to achieve the desired SNR. Then add the specified dispersion delay to the simulated pulses and inject them into the observational data. Following this, we use the PyCUDA framework for parallel computation of dispersion delay and for parallel periodic injection of delayed pulses into observational data. As a result, we obtain simulated pulsar data with realistic background noise.

Hence, it is feasible to simulate data containing a specified number of pulsars by injecting signals with distinct parameters within a defined DM range. This approach allows multiple simulated pulsars to coexist within the same DM range. By searching for pulsars within this DM range in the injected data and examining the ranking of these simulated pulsars among all candidates, we can evaluate the speed-up of our model in the folding stage of the pulsar search pipeline. Due to differences in radio environments and operating frequencies, pulsar candidates generated by different telescopes exhibit variations in data distribution. For newly constructed telescopes, they do not have enough samples to train models, so we can fine-tune the models with a small number of both real and simulated pulsar candidates, preserving the feature extraction ability obtained from training a large dataset. We validated the feasibility of the approach through fine-tuning experiments on MWA pulsar candidates.

4 RESULTS

4.1 Pipeline speed-up

We benchmarked the speed-up of our strategy using 30 minutes of FAST observation data from project PT2020_0074, targeting the

[7] https://github.com/inducer/pycuda

globular cluster NGC 5904. The observation parameters had a sample time of 49.152 µs, 4096 frequency channels, and 4 polarizations. The 30-minute observation generates 545 GB of full data. We need to fold with all possible combinations of DM values and periods. After de-dispersion with the set of planned DM values, the size of each single de-dispersed time series is reduced by a factor of 4096×4, resulting in a final size of 34 MB. Following the speed-up strategy described in Section 2.1, we fold the de-dispersed time series data, with each series corresponding to a particular DM value. The time required to fold the de-dispersed time series data (T_1) is much shorter than the time required to fold the full data (T_0). The speed-up in the folding step time is given by the ratio N_1/N_0,

$$\text{Speed-up} = \frac{N_0 \times T_1 + N_1 \times T_0}{N_0 \times T_0} \approx \frac{N_1}{N_0}, \quad \text{as } T_1 \ll T_0 \tag{2}$$

where N_0 is the number of DM-period combinations generated by PRESTO, and N_1 is the number of candidates that require folding after AI model filtering. This optimization significantly reduces the computational cost of the pulsar search by limiting full data folding, which focuses on a subset of the most promising pulsar candidates.

In the globular cluster NGC 5904, there are seven known pulsars (Zhang et al. 2023). Based on their DM range from 29.31 to 30.05 pc cm^{-3}, the full data were de-dispersed from 29.05 to 30.25 pc cm^{-3} in steps of 0.05 pc cm^{-3} after careful RFI removal. We then applied acceleration search using the parameters "-zmax 500, -wmax 500 and -numharm 32", and filtered the results with ACCEL_sift.py, producing 1906 candidates. A total of 1264 were excluded due to DM problems (including cases with insufficient numbers across DM trials), with another 106 filtered out as harmonics, leaving 536 pulsar candidates. If we use the standard threshold of 0.5, 74 candidates were selected for further folding on the full data, including the fundamental-period and harmonic candidates of the seven known pulsars. Each known pulsar detected in the observation may correspond to multiple candidates, including both fundamental-period and harmonic ones. Our acceleration strategy performs optimally when it maximizes the filtering out of non-pulsar candidates without missing any known pulsars. To achieve this, we need the fundamental-period candidates corresponding to the known pulsars in this globular cluster to be retained after filtering by our model. Among these, the candidate with the lowest predicted probability of being a pulsar still ranks within the top 13% of all candidates, corresponding to the top 72 candidates. This shows that the fundamental-period and harmonic candidates of known pulsars are ranked near the top of all candidates, ensuring that our strategy reduces the number of folding parameters while avoiding any missed pulsars. After using our AI model, the number of folding parameters was reduced by a factor of 10, resulting in a tenfold speed-up of the folding process. Thus, AI effectively speeds up the folding process tenfold while successfully identifying all detectable pulsars in the global cluster.

We used data from 43 known pulsars in a single observation of the SMART survey to evaluate the speed-up across a large parameter space. Using the search pipeline described in Section 3.1, we processed the data and obtained 25,691 potential DM-period combinations. These combinations were then folded on 1-D time series data to generate the "snapshot" candidates. We applied a standard threshold of 0.5 to identify and select candidates. Our model selected 436 candidates for further folding with the full data. Manual inspection of these selected candidates confirmed that they include the fundamental candidates of all 43 known pulsars, as well as some harmonic candidates. As mentioned in Section 3.2, pointings containing known pulsars within a single observation typically produce more candidates than those without pulsars. Therefore, the average number of candidates from searches on known pulsar data tends to be somewhat higher than the average number generated across all pointings in the survey. Using the same search pipeline, we performed a blind search on 960 pointings from the same observation and obtained a total of 569,869 candidates. Each pointing containing a known pulsar produced an average of 597 candidates, slightly higher than the 593 candidates from other pointings. Among the candidates generated from these 960 pointings, our AI model selected 6,478, with only 1.14% marked for further verification. So, our speed-up tests using known pulsar data from a single observation in the SMART survey provide a relatively conservative estimate. Consistent with the previous analysis on globular cluster data, this result suggests that our method can accelerate the folding step by approximately a factor of 60 in large-scale pulsar searches conducted with the MWA-SMART survey. This experiment shows significantly better performance compared to the globular cluster test. This improvement is mainly due to the excellent radio environment of MWA (Offringa et al. 2015), which results in much less RFI than FAST. In addition, the globular cluster test was conducted in a restricted parameter space that contained multiple known pulsars, leading to a higher number of pulsar-like candidates. Considering the above analysis, our method performs better in the large-scale parameter space typical of pulsar surveys.

We applied the simulation program described in Section 3.2 to 10 minutes of data from the FAST pulsar backend, while the telescope was not pointing to any known pulsars. This data served as the background for periodically injecting pulses to simulate pulsars. We generate 20 simulated pulsars with the following injected parameters: the DM ranges from 91 to 109 pc cm^{-3}, the SNR ranges from 5 to 15, and the duty cycle ranges from 1% to 10%. We evaluated the model on simulated data to verify the tenfold speed-up during the folding process. Using PRESTO with 0.1 DM value increments between 90 and 110 pc cm^{-3} generated 200 de-dispersion trails. The application of the acceleration search parameters "-zmax 200, -wmax 200 and -numharm 16", followed by filtering with ACCEL_sift.py, found 2141 candidates. After removing 1697 candidates due to DM problems and excluding 143 harmonics, 299 pulsar candidates remained. Among them, 48 candidates had predicted pulsar probabilities above 50% from the classification model, successfully recovering all 20 simulated pulsars. And the 20 injected pulsars ranked within the top 22 scores, representing the top 7.4% of all pulsar candidates. Using FAST observational globular cluster data and simulated data, our trained model consistently ranked known pulsars within the top 10% of all candidates. This demonstrated that our strategy recovered all known pulsars and achieved a tenfold speed-up in the folding step within a restricted parameter space.

Table 3. Confusion matrix for pulsar classification.

	Predicted: Pulsar	**Predicted: Non-pulsar**
Actual: Pulsar	True Positive (TP)	False Negative (FN)
Actual: Non-pulsar	False Positive (FP)	True Negative (TN)

4.2 Evaluation of AI model performance

We used the following four metrics to evaluate the model's performance for pulsar classification, calculating them based on the confusion matrix presented in Table 3.

(i) Accuracy measures the overall correctness of the model by calculating the proportion of true results (both true positives and true negatives) among the total number of cases. It is defined as the ratio of correctly predicted instances to the total instances.

Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$

(ii) Precision indicates the proportion of positive identifications that were actually correct. It is defined as the ratio of true positive results to the sum of true positives and false positives.

Precision = $\frac{TP}{TP+FP}$

(iii) Recall measures the proportion of actual positives that were correctly identified by the model. It is defined as the ratio of true positive results to the sum of true positives and false negatives.

Recall = $\frac{TP}{TP+FN}$

(iv) F-score (F1) is the harmonic mean of precision and recall. It provides a metric that balances both the precision and recall of a model.

F-score = $2 \times \frac{Precision \times Recall}{Precision+Recall}$

The evaluation protocol comprised three distinct phases: Phase I assessed the model's performance on 3,000 candidates from an independent validation subset (25%) of the complete datasets (12,000 samples) from FAST/Arecibo/Parkes. Phase II conducted blind testing with 10,000 fundamental candidates (4,000 positive and 6,000 negative) from the MWA-SMART pulsar survey, where the 4,000 positive candidates are essentially multiple detections of 86 known pulsars, as a consequence of the MWA complex tied-array beam patterns with multiple strong side lobes (Bhat et al. 2023). The results demonstrate maintained detection efficacy (96.65% recall) in independent test datasets. After fine-tuning with 1,000 samples from MWA, the recall increased to 97.925%. Phase III evaluated the GBNCC survey candidates (from Zhu et al. 2014) using our trained model with cross-validation, comparing the results with prior work. All phases applied consistent pre-processing pipelines and selection criteria, ensuring comparability across different survey configurations.

In Phase II and Phase III, we evaluate classification performance using our model (Multi) and the PICS model (Zhu et al. 2014), with both models applying thresholds of 0.5 and 0.3. A threshold of 0.5 is used in binary classification to separate the two classes based on equal probability. Since non-pulsar candidates greatly exceed pulsar candidates in pulsar surveys, and our method serves as an initial filtering step where missing pulsar candidates are undesirable, we also test a lower threshold of 0.3. The PICS model is tested with the recommended "clfl2_PALFA.pkl" weights. Since PICS adopts an ensemble structure that independently trains on four features (profile, time-phase, frequency-phase, and DM curve) and combines their outputs using logistic regression (Hosmer Jr et al. 2013), we design a comparison experiment as follows. In Phase II, we average the outputs of PICS's ANN (for profile) and CNN (for time-phase) to represent its prediction based on time-domain features, and compare the results with Multi on SMART survey candidates. In Phase III, PICS uses all four features for classification, while Multi relies on time-domain features, allowing us to analyze the impact of using temporal information alone.

4.2.1 *Evaluation of validation data*

As shown in Table 4, the internal validation achieved 98.3% accuracy on the 3,000 candidates test subset, with precision and recall rates of 98.25% and 98.44%, respectively. Notably, the model demonstrated robust performance across different telescopes within the training data: FAST (96.6% recall), Parkes (98.6% recall), and Arecibo (99.28% recall). This consistency suggests effective generalization across varying observational parameters and instrumental characteristics.

Table 4. Performance metrics for different telescopes (accuracy, precision, recall, F1).

Telescope	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)
FAST	98.17	99.22	96.60	97.90
Parkes	98.11	98.03	98.60	98.31
Arecibo	98.73	98.10	99.28	98.69
Overall	**98.30**	**98.25**	**98.44**	**98.34**

4.2.2 *Model evaluation with MWA data*

Table 5 summarizes the results of the SMART survey. Both models use time-domain features for classification. The numbers indicate the counts of correctly classified positive samples and misclassified negative samples. The PICS model downsampling algorithm standardizes the feature dimensions by reducing the profile features to 64 and the time-phase features to 48. This leads to a decrease in precision for candidates from the SMART survey, which originally had a bin number of 100. Consequently, our pulsar time-domain feature classification model, designed for multi-dimensional input, effectively handles the varying number of bins among the original candidates. Due to the excellent radio environment of MWA (Offringa et al. 2015), the number of pulsar-like negative candidates caused by narrow-band RFIs is very low. Multi model exhibits consistently strong performance on both positive and negative pulsar candidates.

For the SMART survey, the classification performance is further improved through the fine-tuning strategy described in Section 2.3. A fine-tuning dataset is constructed using 200 pulsars and 500 non-pulsars from the SMART survey, added with 300 simulated pulsars generated based on MWA data, resulting in a total of 1,000 samples. Fine-tuning is performed on the final classification layer to adapt the model to the data distribution of SMART pulsar candidates, enhancing its discrimination capability for both positive and negative samples. The remaining misclassified pulsars are mostly candidates generated from bright sources observed through MWA sidelobes. These candidates are characterized by faint, diffuse vertical structures in their time-phase features, appearing at a fixed pulse phase and confined to a limited portion of the time axis. These structures correspond to periodic pulses occurring during part of the observation, caused by bright pulsars observed via the sidelobes of the beamformed pointing. In the SMART test dataset, all known pulsars were correctly identified by the Multi model. Only a small number of positive samples observed through sidelobe responses were not identified.

4.2.3 *GBT data comparison*

The results from the GBT are summarized in the following Table 6. The PICS model considers both time and frequency features, while Multi focuses only on time-domain features. For the GBNCC dataset, the Multi model effectively identifies the intrinsic features of both fundamental and harmonic candidates. However, the model's capacity to filter out RFI is significantly impeded by the lack of detailed frequency and DM information. In the future, we will train a

Table 5. SMART results

Model	Threshold	Pulsar (4,000)	Non-pulsar (6,000)
PICS	>0.5	3587	362
	>0.3	3789	931
MULTI	>0.5	3866	270
	>0.3	3910	360
Finetuned MULTI	>0.5	3917	203
	>0.3	3940	300

Table 6. GBNCC results

Model	Fundamental (56)	Harmonic (221)	Non-pulsar (10,000)
PICS (>0.5)	53	207	43
PICS (>0.3)	53	211	132
MULTI (>0.5)	56	219	762
MULTI (>0.3)	56	221	1077

model using advanced algorithms to identify candidates with detailed time and frequency information, enhancing RFI filtering.

5 DISCUSSION

The SKA, as the next-generation radio telescope, is predicted to detect over 20,000 new pulsars, of which 7,000-9,000 of them from SKA-1 Low (Keane et al. 2014; Xue et al. 2017), fundamentally advancing our understanding of neutron star populations and extreme astrophysical phenomena. Due to the large field of view (FoV) of radio telescope arrays, even with long single observation times used in pulsar surveys, it is possible to complete the pulsar survey plan in a short amount of time compared to single-dish telescopes. For example, each observation in the SMART survey uses an 80-minute dwell time, allowing the full survey coverage to be completed in less than 100 hours of telescope time (Bhat et al. 2023). As the observation time for a single point is extended, the resulting full data size increases correspondingly. This directly increases the time required to fold the increasing size of the full data. Note that our method effectively accelerates pulsar searches if folding dominates computational time. Our work addresses this critical bottleneck in pulsar search pipelines through three key innovations.

First, we develop a time-domain feature extraction method from pre-folded "snapshot" candidates that enables pulsar identification with 98.44% recall before full folding. This performance is validated on both cross-validation and independent holdout datasets. This early-stage filtering reduces the required full-fold operations by 90% in real and simulated data within a restricted parameter space. On a large parameter space approximating the processing in the SMART survey, our method improved the efficiency of the folding step by a factor of 60. This corresponds to a 98% reduction in required full-fold operations.

Second, we developed an efficient pulsar simulation script that injects simulated pulsars with diverse parameters into real observational noise data, addressing the data scarcity challenge for newly built telescopes. This approach effectively resolves data distribution discrepancies across candidates from different telescopes, particularly when insufficient real pulsar samples are available for model fine-tuning.

Third, the hybrid network architecture demonstrates robust adaptability to varying input dimensions. Our denoising network further enhances performance by reinforcing 2-D features. When processing pulsar candidates with different bin sizes, the model maintains >98% classification accuracy through the learnable feature layers.

The MWA-SMART survey demonstrates the advantage of a large FoV combined with extended single observation times, achieving a 20-fold increase in dwell time away from the Galactic plane compared to the HTRU survey (Bhat et al. 2023; Keith et al. 2010). The future SKA survey will also benefit from a relatively large FoV and longer observations, making it critical to reduce computational overhead in the folding step. By fine-tuning the model using the simulated pulsar samples from the newly constructed radio telescope, a robust model can be established during the preliminary investigation phase of the pulsar survey. The methodology establishes a reliable pathway for deploying an AI model in pulsar surveys of next-generation telescopes such as the SKA.

ACKNOWLEDGEMENTS

We sincerely thank the referee, Scott Ransom, for his highly insightful and constructive comments which were so valuable in helping us improve the manuscript. This work is supported by the National SKA Program of China No. 2020SKA0120200, the National Natural Science Foundation of China (NSFC) grant Nos. 12421003, 12503055, 12041303, 62176247, 12203070, 12041304, the Beijing Nova Program (No. 20250484786), the Postdoctoral Fellowship Program of CPSF under Grant Number GZB20250737. This scientific work uses data obtained from Inyarrimanha Ilgari Bundara, the CSIRO Murchison Radio-astronomy Observatory. We acknowledge the Wajarri Yamaji People as the Traditional Owners and Native Title Holders of the observatory site. Support for the operation of the MWA is provided by the Australian Government (NCRIS), under a contract to Curtin University administered by Astronomy Australia Limited (AAL). We acknowledge the Pawsey Supercomputing Centre which is supported by the Western Australian and Australian Governments. This work made use of the data from FAST (Five-hundred-meter Aperture Spherical radio Telescope). FAST is a Chinese national mega-science facility, operated by National Astronomical Observatories, Chinese Academy of Sciences.

CODE AND DATA AVAILABILITY

The data supporting this article will be shared when a reasonable request is made to the corresponding author. The code used for model implementation and experiments in this study is publicly available at: https://github.com/ifuqy/Multi. The repository also provides the trained model weights used in our experiments, along with usage instructions.

REFERENCES

Balakrishnan V., Champion D., Barr E., Kramer M., Sengar R., Bailes M., 2021, Monthly Notices of the Royal Astronomical Society, 505, 1180

Barr E. D., et al., 2013, Monthly Notices of the Royal Astronomical Society, 435, 2234

Bhat N., et al., 2023, Publications of the Astronomical Society of Australia, 40, e021

Cai N., Han J., Jing W., Zhang Z., Zhou D., Chen X., 2023, Research in Astronomy and Astrophysics, 23, 104005
Cooley J. W., Tukey J. W., 1965, Mathematics of computation, 19, 297
Cordes J. M., et al., 2006, The Astrophysical Journal, 637, 446
Eatough R. P., Molkenthin N., Kramer M., Noutsos A., Keith M., Stappers B., Lyne A., 2010, Monthly Notices of the Royal Astronomical Society, 407, 2443
Han J., et al., 2021, Research in Astronomy and Astrophysics, 21, 107
He K., Zhang X., Ren S., Sun J., 2016, in Proceedings of the IEEE conference on computer vision and pattern recognition. pp 770–778
Hewish A., Bell S., Pilkington J., Scott P., Collins R., 1968, Nature, 217, 709
Ho J., Jain A., Abbeel P., 2020, Advances in neural information processing systems, 33, 6840
Hosmer Jr D. W., Lemeshow S., Sturdivant R. X., 2013, Applied logistic regression. John Wiley & Sons
Hotan A. W., van Straten W., Manchester R. N., 2004, Publications of the Astronomical Society of Australia, 21, 302
Hu J., Shen L., Sun G., 2018, in Proceedings of the IEEE conference on computer vision and pattern recognition. pp 7132–7141
Hulse R. A., Taylor J. H., 1975, Astrophysical Journal, vol. 195, Jan. 15, 1975, pt. 2, p. L51-L53., 195, L51
Ioffe S., Szegedy C., 2015, in International conference on machine learning. pp 448–456
Jähne B., 2005, Digital image processing. Springer
Jiang P., et al., 2020, Research in Astronomy and Astrophysics, 20, 064
Jing W., et al., 2025, arXiv preprint arXiv:2506.14519
Keane E., et al., 2014, arXiv preprint arXiv:1501.00056
Keith M., Eatough R., Lyne A., Kramer M., Possenti A., Camilo F., Manchester R., 2009, Monthly Notices of the Royal Astronomical Society, 395, 837
Keith M., et al., 2010, Monthly Notices of the Royal Astronomical Society, 409, 619
Krizhevsky A., Sutskever I., Hinton G. E., 2012, Advances in neural information processing systems, 25
LeCun Y., Bottou L., Bengio Y., Haffner P., 1998, Proceedings of the IEEE, 86, 2278
Lee K., et al., 2013, Monthly Notices of the Royal Astronomical Society, 433, 688
Li D., et al., 2018, IEEE Microwave Magazine, 19, 112
Lorimer D., 2011, Astrophysics Source Code Library, pp ascl–1107
Lorimer D. R., Kramer M., 2005, Handbook of pulsar astronomy. Cambridge Observing Handbooks for Research Astronomers Vol. 4, Cambridge university press
Luo R., et al., 2022, Monthly Notices of the Royal Astronomical Society, 513, 5881
Manchester R., Lyne A., Taylor J., Durdin J., Large M., Little A., 1978, Monthly Notices of the Royal Astronomical Society, 185, 409
Manchester R. N., Hobbs G. B., Teoh A., Hobbs M., 2005, The Astronomical Journal, 129, 1993
Nair V., Hinton G. E., 2010, in Proceedings of the 27th international conference on machine learning (ICML-10). pp 807–814
Offringa A., et al., 2015, Publications of the Astronomical Society of Australia, 32, e008
Ransom S. M., 2007, Proceedings of the International Astronomical Union, 3, 291
Ransom S., 2011, Astrophysics source code library, pp ascl–1107
Rickett B. J., 1990, Annual review of astronomy and astrophysics, 28, 561
Ronneberger O., Fischer P., Brox T., 2015, in Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18. pp 234–241
Sanidas S., et al., 2019, Astronomy & Astrophysics, 626, A104
Stovall K., et al., 2014, The Astrophysical Journal, 791, 67
Szegedy C., et al., 2015, in Proceedings of the IEEE conference on computer vision and pattern recognition. pp 1–9
Tan C. M., Lyon R., Stappers B., Cooper S., Hessels J., Kondratiev V., Michilli D., Sanidas S., 2018, Monthly Notices of the Royal Astronomical Society, 474, 4571
Taylor J. H., 1991, Proceedings of the IEEE, 79, 1054
Tingay S. J., et al., 2013, Publications of the Astronomical Society of Australia, 30, e007
Vincent P., Larochelle H., Bengio Y., Manzagol P.-A., 2008, in Proceedings of the 25th international conference on Machine learning. pp 1096–1103
Wayth R. B., et al., 2018, Publications of the Astronomical Society of Australia, 35, e033
Woo S., Park J., Lee J.-Y., Kweon I. S., 2018, in Proceedings of the European conference on computer vision (ECCV). pp 3–19
Xue M., et al., 2017, Publications of the Astronomical Society of Australia, 34, e070
Yosinski J., Clune J., Bengio Y., Lipson H., 2014, Advances in neural information processing systems, 27
Zhang L., et al., 2023, The Astrophysical Journal Supplement Series, 269, 56
Zhu W. W., et al., 2014, The Astrophysical Journal, 781, 117

APPENDIX A: MODEL DETAILS

The Peak Signal-to-Noise Ratio (PSNR) is computed as:

$$\begin{aligned}
\mathrm{PSNR} &= 20 \cdot \log_{10}\left(\frac{\mathrm{MAX}}{\sqrt{\mathrm{MSE}+\varepsilon}}\right) \\
\mathrm{MAX} &= \max(\mathrm{pred}_{\max}, \mathrm{orig}_{\max}) - \min(\mathrm{pred}_{\min}, \mathrm{orig}_{\min}) \\
\mathrm{MSE} &= \frac{1}{N}\sum_{i=1}^{N}(\mathrm{pred}_i - \mathrm{orig}_i)^2
\end{aligned} \tag{A1}$$

where MAX is the dynamic range of the images (i.e., the difference between the maximum and minimum pixel values), MSE is the mean squared error between the predicted and target images, and ε is a small constant (e.g., $\varepsilon = 10^{-8}$) added for numerical stability to avoid division by zero.

The loss function of the denoising model is defined as:

$$\begin{aligned}
\mathcal{L}_{\mathrm{total}} &= \lambda_{\mathrm{MSE}} \cdot \mathcal{L}_{\mathrm{MSE}} + \lambda_{\mathrm{SSIM}} \cdot \mathcal{L}_{\mathrm{SSIM}} + \lambda_{\mathrm{grad}} \cdot \mathcal{L}_{\mathrm{grad}} \\
\lambda_{\mathrm{MSE}} &= 0.6, \quad \lambda_{\mathrm{SSIM}} = 0.2, \quad \lambda_{\mathrm{grad}} = 0.2 \\
\mathcal{L}_{\mathrm{SSIM}} &= 1 - \mathrm{SSIM}(\mathrm{pred}, \mathrm{orig}) \\
\mathcal{L}_{\mathrm{grad}} &= \frac{1}{2} \cdot \left\|\nabla_y \mathrm{pred} - \nabla_y \mathrm{orig}\right\|_1 + \frac{1}{2} \cdot \left(1 - \|\nabla_x \mathrm{pred}\|_1\right)
\end{aligned} \tag{A2}$$

Here, $\mathcal{L}_{\mathrm{MSE}}$ is the mean squared error, $\mathcal{L}_{\mathrm{SSIM}}$ controls the contribution of the SSIM (Structural Similarity Index Measure) loss , and $\mathcal{L}_{\mathrm{grad}}$ weights the gradient loss for preserving edges and textures.

The SSIM is computed as:

$$\begin{aligned}
\mathrm{SSIM}(x, y) &= \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \\
\mu_x &= \mathrm{mean}(x), \quad \mu_y = \mathrm{mean}(y) \\
\sigma_x^2 &= \mathrm{mean}\left((x-\mu_x)^2\right), \quad \sigma_y^2 = \mathrm{mean}\left((y-\mu_y)^2\right) \\
\sigma_{xy} &= \mathrm{mean}\left((x-\mu_x)(y-\mu_y)\right) \\
L &= \max(x, y) - \min(x, y) \\
C_1 &= (0.01 \cdot L)^2, \quad C_2 = (0.03 \cdot L)^2
\end{aligned} \tag{A3}$$

where x and y denote the predicted and original images, respectively.

This paper has been typeset from a TEX/LATEX file prepared by the author.

Table A1. Architecture of the U-Net model for image denoising.

Module	Layer Type	Configuration	Output Shape
Encoder (Downsampling path)			
Down1	Conv2D + SiLU	$1 \rightarrow 32$, kernel=5, padding=2	$B \times 32 \times H \times W$
Down2	Conv2D + SiLU	$32 \rightarrow 64$, kernel=5, padding=2	$B \times 64 \times H \times W$
	MaxPool2D	kernel=2, stride=2	$B \times 64 \times \frac{H}{2} \times \frac{W}{2}$
Down3	Conv2D + SiLU	$64 \rightarrow 64$, kernel=5, padding=2	$B \times 64 \times \frac{H}{2} \times \frac{W}{2}$
	MaxPool2D	kernel=2, stride=2	$B \times 64 \times \frac{H}{4} \times \frac{W}{4}$
Decoder (Upsampling path)			
Up1	Conv2D + SiLU	$64 \rightarrow 64$, kernel=5, padding=2	$B \times 64 \times \frac{H}{4} \times \frac{W}{4}$
	Upsample (scale=2) + Skip Add	-	$B \times 64 \times \frac{H}{2} \times \frac{W}{2}$
Up2	Conv2D + SiLU	$64 \rightarrow 32$, kernel=5, padding=2	$B \times 32 \times \frac{H}{2} \times \frac{W}{2}$
	Upsample (scale=2) + Skip Add	-	$B \times 32 \times H \times W$
Up3	Conv2D + SiLU	$32 \rightarrow 1$, kernel=5, padding=2	$B \times 1 \times H \times W$

Notes: Conv2D = 2D convolution layer for feature extraction; SiLU = activation function; MaxPool2D = downsampling by max pooling; Upsample = feature map upscaling; Skip Add = adding encoder features to decoder for detail recovery; B, H, W = batch size, height, width.

Table A2. Architecture of the classifier model.

Module	Layer Type	Configuration	Output Shape
SE-ResNet10_1D Branch (for profile)			
Init	Conv1D + BN	$1 \rightarrow 8$, kernel=7, stride=2, padding=3	$B \times 8 \times L$
	AvgPool1D	kernel=2, stride=1, padding=1	$B \times 8 \times L$
Block1	Bottleneck1D + SE	$8 \rightarrow 16$, expansion=2	$B \times 16 \times L$
Block2	Bottleneck1D + SE	$16 \rightarrow 16$, expansion=2	$B \times 16 \times L$
Block3	Bottleneck1D + SE (downsample)	$16 \rightarrow 32$, expansion=2	$B \times 32 \times L/2$
Pool	AvgPool1D	kernel=2, stride=1, padding=1	$B \times 32 \times L/2$
Flatten	-	-	$B \times F_1$
CNN_Attention Branch (for time-phase)			
Conv1	Conv2D + BN	$1 \rightarrow 8$, kernel=(5,3), stride=1, padding=(2,1)	$B \times 8 \times H \times W$
Pool1	AvgPool2D	kernel=2	$B \times 8 \times H/2 \times W/2$
CA	Channel Attention	reduction ratio=8	$B \times 8 \times H/2 \times W/2$
SA	Spatial Attention	kernel=5 (along H)	$B \times 8 \times H/2 \times W/2$
Conv2	Conv2D + BN	$8 \rightarrow 16$, kernel=(5,3), stride=2	$B \times 16 \times H/4 \times W/4$
Pool2	AvgPool2D	kernel=2	$B \times 16 \times H/8 \times W/8$
Flatten	-	-	$B \times F_2$
Feature Fusion and Classifier			
Fusion	Concatenate	$[F_1, F_2] \rightarrow 9824$	$B \times 9824$
FC1	Linear + BN + ReLU + Dropout(0.2)	$9824 \rightarrow 512$	$B \times 512$
FC2	Linear + BN + ReLU + Dropout(0.4)	$512 \rightarrow 128$	$B \times 128$
FC3	Linear	$128 \rightarrow 2$	$B \times 2$

Notes: Conv1D / Conv2D = convolution layer for 1D / 2D inputs; BN = batch normalization; AvgPool1D / AvgPool2D = average pooling for downsampling; SE = squeeze-and-excitation attention; Bottleneck1D = Bottleneck block; Channel / Spatial Attention = attention mechanisms for channels / spatial dimensions; Flatten = reshape feature map to vector; Concatenate = combine features from different branches; Linear = fully connected layer; ReLU = activation function; Dropout = regularization by random feature masking; B, L, H, W = batch size, sequence length, height, width; F1, F2 = flattened feature dimensions from each branch.