Monthly Notices of the ROYAL ASTRONOMICAL SOCIETY

MNRAS **528,** 4053–4067 (2024) https://doi.org/10.1093/mnras/stae245
Advance Access publication 2024 January 23

The GPU phase folding and deep learning method for detecting exoplanet transits

Kaitlyn Wang,[1,2★] Jian Ge,[3★] Kevin Willis,[2] Kevin Wang[4] and Yinan Zhao[5]

[1]*The Harker School, 500 Saratoga Ave, San Jose, CA 95129, USA*
[2]*Science Talent Training Center, Gainesville, FL 32606, USA*
[3]*Division of Science and Technology for Optical Astronomy, Shanghai Astronomical Observatory, Chinese Academy of Sciences, Shanghai 200030, China*
[4]*Department of Computer Science, Princeton University, PO Box 430, Princeton, NJ 08544, USA*
[5]*Department of Astronomy, University of Geneva, Versoix 1290, Switzerland*

Accepted 2024 January 21. Received 2024 January 20; in original form 2023 December 4

ABSTRACT

This paper presents GPFC, a novel Graphics Processing Unit (GPU) Phase Folding and Convolutional Neural Network (CNN) system to detect exoplanets using the transit method. We devise a fast-folding algorithm parallelized on a GPU to amplify low signal-to-noise ratio transit signals, allowing a search at high precision and speed. A CNN trained on two million synthetic light curves reports a score indicating the likelihood of a planetary signal at each period. While the GPFC method has broad applicability across period ranges, this research specifically focuses on detecting ultrashort-period planets with orbital periods less than one day. GPFC improves on speed by three orders of magnitude over the predominant Box-fitting Least Squares (BLS) method. Our simulation results show GPFC achieves 97 per cent training accuracy, higher true positive rate at the same false positive rate of detection, and higher precision at the same recall rate when compared to BLS. GPFC recovers 100 per cent of known ultrashort-period planets in *Kepler* light curves from a blind search. These results highlight the promise of GPFC as an alternative approach to the traditional BLS algorithm for finding new transiting exoplanets in data taken with *Kepler* and other space transit missions such as *K*2, *TESS*, and future *PLATO* and Earth 2.0.

Key words: methods: data analysis – techniques: photometric – catalogues – surveys – planets and satellites: detection.

1 INTRODUCTION

Since the discovery of the first exoplanets (Wolszczan & Frail 1992; Major & Queloz 1995), more than 5000 exoplanets have been found and many thousands of candidates have yet to be confirmed. Compared with the other major exoplanet detection methods – radial velocity (Campbell, Walker & Yang 1988), direct imaging (Chauvin et al. 2004), and gravitational microlensing (Beaulieu et al. 2006) – the transit method (Charbonneau et al. 2000) has made the biggest contribution empowered by the large-scale transit surveys including *Kepler* (Borucki et al. 2010), *K*2 (Howell et al. 2014), *TESS* (Ricker et al. 2014), and beyond.

At a high level, planetary transit detection involves the following general steps. First, the light-curve data are pre-processed and detrended to remove stellar variability (Smith et al. 2012; Stumpe et al. 2012). After that, a variety of algorithms are used to search for periodic transit signals in the light curve. The Box-fitting Least Square (BLS) method, introduced by Kovács, Zucker & Mazeh (2002), has been the predominant method for transit searches in large data sets and is widely used in the ground- and space-based surveys. There have been various extensions and optimizations on the original BLS method (Cameron et al. 2006; Renner et al. 2008; Carter & Agol 2013; Boufleur et al. 2014; Ofir 2014; Hartman & Bakos 2016; Caceres et al. 2019). Panahi & Zucker (2021) optimized BLS for low-cadence surveys such as Gaia Collaboration (2016), and Shahaf et al. (2022) proposed an efficient periodicity detection algorithm by combining two long withstanding techniques – the fast-folding algorithm (FFA: Staelin 1969) and BLS. Following transit signal detection, a list of threshold crossing events (TCEs) is generated. Then, a vetting process is conducted to filter out a vast number of false positives in the TCEs caused by instrumental noise or astrophysical variability. Various machine-learning auto-vetting methods have been developed, including Robovetter (Thompson et al. 2018), Autovetter (McCauliff et al. 2015), and Astronet (Shallue & Vanderburg 2018). Astronet employed deep learning (Convolutional Neural Network, CNN) to vet *Kepler* candidates, and it was thereafter adapted to more surveys such as *K*2 (Dattilo et al. 2019), Next-Generation Transit Survey *(NGTS)* (Chaushev et al. 2019), Wide Angle Search for Planets *(WASP)* (Schanche et al. 2018), and *TESS* (Yu et al. 2019; Osborn et al. 2020; Olmschenk et al. 2021; Rao et al. 2021; Ofman et al. 2022).

Meanwhile, some researchers have been exploring a different approach which detects exoplanets directly from light curves via machine learning without the involvement of the BLS method (Pearson, Palafox & Griffith 2017; Zucker & Giryes 2018; Chintarungruangchai & Jiang 2019; Cui et al. 2021; Malik, Moster & Obermeier 2022a). Among these, some utilize the phase folding

★ E-mail: 24kaitlynw@students.harker.org (KW); jge@shao.ac.cn (JG)

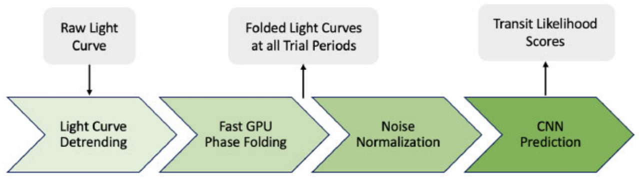


Figure 1. Fast GPFC Processing Pipeline. The GPFC approach initiates by ingesting a raw light curve and subjects it to detrending. Following this, the light curve is phase folded using a high-precision grid of trial periods. Then, the folded results are noise normalized and fed into the CNN, which produces a probability score indicative of the likelihood that the light curve contains a transit event.

technique whereas others do not. In general, the methods processing light curves without phase folding limit their sensitivity to transit signals with signal-to-noise ratios (SNR) above 10; for the methods that involve phase folding (Pearson et al. 2017; Yeh & Jiang 2020), since the resolution of the trial folding periods limits the accuracy of unknown transit detection, those efforts mainly focused on simulated data rather than applying their neural networks to search for new candidates. Practically, in order to detect shallow transit signals generated by small exoplanets, there must be a high resolution of trial folding periods, which converts to prohibitive computation time. This is the critical problem that motivates our novel method, the GPU (Graphics Processing Units) Phase Folding and Convolutional Neural Network (GPFC) method. With GPFC, we increase computational speed to achieve phase folding at high-resolution trial periods. In GPFC, we developed a scalable phase folding algorithm leveraging GPU's parallelism to process phase folding with high precision, together with a CNN to evaluate transit signals from the high-dimensional folded results. We demonstrate that the GPFC detection system is capable of searching exoplanets in large volume of *Kepler* data at three orders of magnitude higher speed than traditional BLS, and it also reports new exoplanet candidates undetected in previous research work.

The organization of this paper is as follows: Section 2 delineates the foundational principles of the GPFC method, detailing each component within its operational pipeline. Section 3 delves into a detailed description of the simulation tests we employed to compare the GPFC method and the classic BLS method. In Section 4, we demonstrate that the GPFC detection system can recover all of confirmed USPs in the *Kepler* Archive – a validation for its potential for new exoplanetary discoveries. Finally, in Section 5, we present a discussion of our method, comparison with various implementations of the BLS method, and outline prospective directions for future research.

2 METHODS

2.1 Overview of the GPFC method

The architecture of the GPFC method is shown in Fig. 1. The GPFC method comprises of a workflow of pre-processing, GPU parallelized phase folding, noise normalization, and CNN transit prediction. To begin with, the GPFC method takes a raw light curve as input, obtained from the Kepler survey in this research, but the method is generic and can be used with other survey data as well. Before analysis, the raw light curve undergoes pre-processing, including detrending and iterative removal of outliers. Next, the light curve is folded at various trial periods, tightly and evenly divided across a designated search range, using the GPU phase folding method. After phase folding, the GPFC method organizes data samples into bins based on their phase and calculates the average flux for each bin. In a typical 64K data point *Kepler* light curve, which is evenly distributed into 256 bins, each bin encompasses 256 data points. This approach shows good tolerance to gaps and irregular individual observations in the original *Kepler* light curve, given that each bin contains a sufficient number of data samples. The resulting folded and binned light curves are normalized to a specific noise level and then passed through a CNN module to evaluate whether any exoplanet transit exists in any of the trial periods.

As the focus of our research is to search ultrashort-period (USP) exoplanet candidates, we choose to search the period range of less than one day, although the GPFC method can be extrapolated to other period ranges too.

A significant challenge of the search for USP planets is the intensive computation needed for detecting signals with short periodicity. Since a priori knowledge of the potential transit period is not available, a vast number of trial periods need to be evaluated. As proposed by VanderPlas (2018), to ensure that a period scan does not miss signals in a periodogram, the total number of required sampling for the periodogram with a total observing time window of T can be calculated by equation (1),

$$N_{\text{sample}} = \frac{N_{\text{o}}T}{L}, \tag{1}$$

where L is the expected width of the signals, and N_{o} is an oversample factor. This formula can be applied to the *Kepler* survey to determine the sampling precision necessary for detecting USP signals within a *Kepler* light curve. As *Kepler* survey spans an observing window of around 4 yr, a typical light curve has total time span $T = 4 \times 365$ d. An analysis of the 43 confirmed USPs listed in the *Kepler* KOI (Kepler Objects of Interest) catalogue reveals that their transit durations lie between (0.03, 0.09) d, prompting us to adopt $L = 0.03$. Adhering to Nyquist's Theorem, we set $N_{\text{o}} = 2$ to satisfy the minimum sampling criteria. Consequently, to detect *Kepler* USP transits, $N_{\text{sample}} = \frac{2\times4\times365}{0.03} = 97,333$ samples are necessary in the USP period range, which translates to a trial grid precision of as fine as 0.7 s. Maintaining this high precision is essential, as even slight deviations from the transit period, on the order of seconds, can obscure the USP transit signal.

Based on the above calculation, we configure our phase folding program to uniformly sample 100 000 periods within a representative USP search period defined as $p \in_R [0.2, 1.0]$ d, resulting in a trial period granularity of 0.7 s. In tandem, the GPU phase folding algorithm categorizes the light curve to 256 bins, producing 100 000 folded light curves, culminating in 100 000 × 256 data points. If a light curve's folding period is the same as the planet's correct transit period, the light curve will manifest a clear transit, as depicted in Fig. 2. Otherwise, the phase misplaced transit signals will become indistinguishable, buried in their surrounding noise after folding.

Specifically, the noise attenuation achieved through phase folding is proportional to $\sqrt{N}$, where N represents the total count of the subsegments folded. This noise attenuation is evident when observing a characteristic *Kepler* light curve with a 4-yr observation window. Given an example USP search period $p \in_R [0.2, 1.0]$ d, N can be calculated as: $N = \frac{4\times365}{p}$, and $\frac{1}{\sqrt{N}} = \frac{1}{\sqrt{\frac{4\times365}{p}}} \in_R [0.01, 0.03]$. This indicates that the noise within the folded light curve diminishes to around 1 per cent ~ 3 per cent of its original magnitude.

Before the folded results are fed into the CNN, we first scale each of the 100 000 folded results to a common noise standard deviation level, as neural networks are often sensitive to the noise level of the

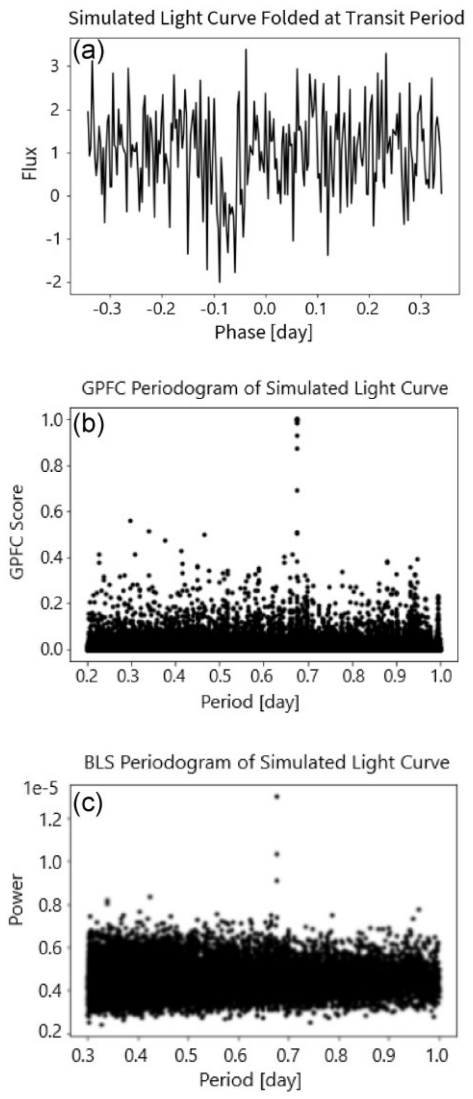


Figure 2. Example outputs of the GPFC and BLS methods on a simulated light curve. The light-curve phase folded at the instrumented transit period is shown in the top panel. The score versus trial period is illustrated for both GPFC (the middle panel) and BLS (the bottom panel), showing peak scores at the correct period.

input data. This normalization specifically entails scaling the folded flux values to a standard deviation of 1.0, while maintaining the mean of the folded results. This procedure aligns with the standard deviation of the synthetic data used in training the neural network. This normalization step standardizes the model prediction scores of the folded results among different targets so that CNN's predictions are comparable across targets.

We then feed the normalized folded results into our CNN, which discerns whether each of the 100 000 folded results contains a transit signal. When such a transit signal exists, we will observe a peak in the model score at the detected orbital period (and we may also see weaker peaks at the harmonics of the orbital period). Because stronger signals result in higher CNN prediction scores, we sort the folded light curves based on the height and width of the peak and choose the best one, which is equivalent to the most probable transit signal among the 100 000 folded results. Fig. 2(a) shows flux values of a simulated folded light curve that has artificial transits at a period of 0.6852 d. As demonstrated in Fig. 2(b) is a graph of CNN score versus trial period for GPFC. Here, GPFC reports scores at all trails periods, revealing a peak score at the correct transit period. For comparison, Fig. 2(c) illustrates a corresponding graph of power versus trial period for BLS. This reference will be explored in greater detail in Section 3.1.

2.2 Pre-processing

The *Kepler* light curves used in this research are produced by the *Kepler* Science Processing Pipeline (Jenkins et al. 2010), with each light curve consisting of integrated flux measurements with a cadence of 1766 s (∼29.4 min) intervals spanning up to four years, in the range of 30 000–70 000 epochs. Before they are analysed by the GPFC system, these *Kepler* light curves are pre-processed in a manner similar to the fitting process illustrated in Vanderburg & Johnson (2014). First, each light curve is divided into multiple segments based on the time and flux gaps observed within the light curve. As we scan through the light curve, segmentation is triggered if the time interval between two adjacent data points exceeds a predefined time gap threshold, or if the flux value difference between the two adjacent data points exceeds a predefined flux gap threshold. In this study, we set the time gap threshold at 30 times the *Kepler* data sampling interval (0.020428 d), equivalent to approximately 14.7 h. This threshold is selected to effectively segment the light curve during relatively long pauses in *Kepler* observations. Additionally, we set the flux gap threshold at five times the standard deviation of the light curve's flux values. This segmentation strategy is intended to enhance the accuracy of the piecewise fitting process. For each segment, spline fitting was then performed with all known transits masked to ensure the preservation of transit signals. Throughout this fitting process, outliers exceeding 3σ were iteratively removed, and spline smoothing parameters were fine-tuned to minimize the Bayesian Information Criterion. Next, after dividing the light curve by the best-fitting spline, we clipped 30 data points from each end of the segment where the spline fits may be skewed. Subsequently, we stitched these segments together, producing a fully detrended light curve. This procedure, illustrated in Fig. 3, effectively detrends the raw light curve and removes low-frequency stellar variability. The pre-processed light curves are subsequently fed into the GPU phase folding module.

2.3 Fast GPU phase folding

Phase folding increases SNR of a light curve which maintains periodic signals while reducing non-periodic noises. It is a critical technique for searching small transiting planets which generate weak and shallow transit signals in a noisy background. At the same time, it dramatically reduces the dimensionality of the data making it feasible for further inference with a CNN. To find small USP exoplanets potentially overlooked by previous methods, the high precision of the folds is essential because an offset of a few seconds may obfuscate a narrow transit signal. On the other hand, however, folding at a large number of trial periods per light curve is computationally untenable with traditional methods. To attain both folding precision on our trial period grid and practical computational speed, we present the GPU phase folding algorithm which utilizes GPU technology to scale the heavy folding workload with high parallelism and great computational speed. In this research, we used the Nvidia GeForce RTX 3090 Ti, a standard commercial GPU card, to implement our GPU phase folding algorithm.

A typical phase folding process (e.g. Shallue & Vanderburg 2018) consists of three steps: (1) computing modular residuals using the

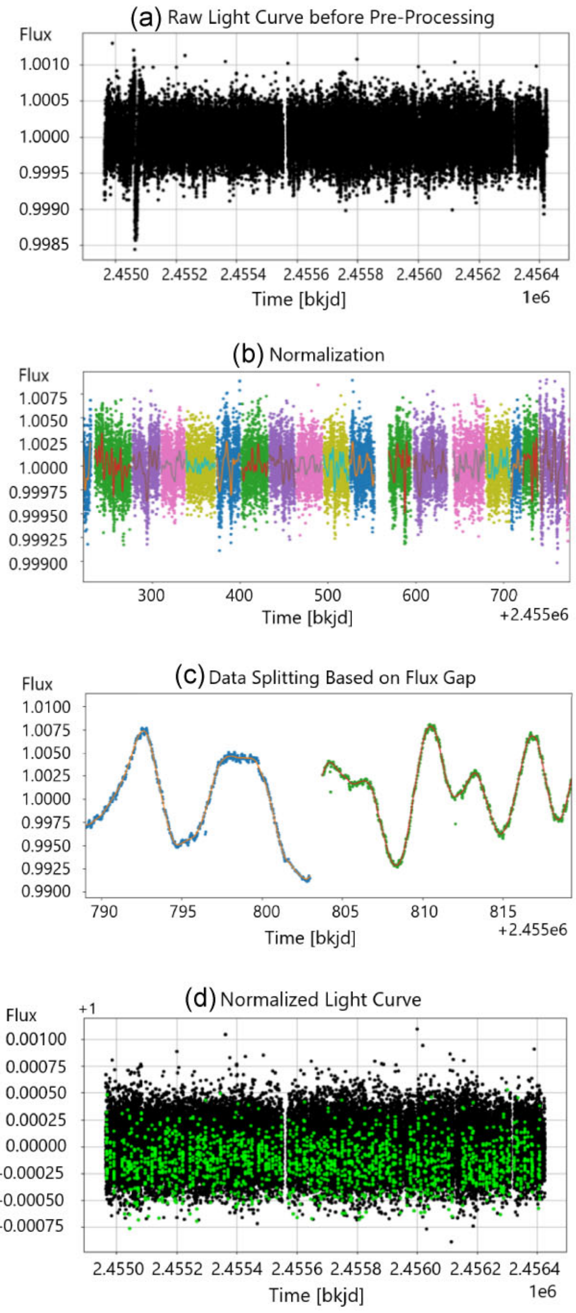


Figure 3. The data pre-processing module: (a) a raw *Kepler* light curve before pre-processing, (b) pre-processing steps conducted: masking of known transits, segmenting the light curve to multiple data sections, and cropping edges, (c) splitting by flux gap criteria and continuum fitting in each segmented section, and (d) the final normalized light curve after pre-processing, where data points in transit windows are plotted in green.

timestamp of each light-curve data point modded by a chosen folding period, (2) sorting the flux data points by their modular residuals, and (3) allocating the flux data points equally across a set number of bins. However, this phase folding method does not fully leverage GPU parallelism, as the GPU library does not support the associated sorting in parallel. On the other hand, since flux data points are binned and averaged immediately post-sorting, the order of the flux data points within each bin is irrelevant and sorting becomes unnecessary.

Based on this analysis, we devised a slightly different phase folding process that can be fully parallelized by GPU. At its core, the objective of phase folding is to group and average flux data points with similar timestamp modular residuals. However, directly mapping of these data points into a pre-determined set of 256 bins is problematic, because certain bins may end up sparsely populated or even entirely empty, while other bins could be densely populated. Such uneven distributions lead to under-populated bins generating noisy average flux values, mainly because these bins become excessively susceptible to anomalies or outliers present in the original flux data points.

Thus we devised a two-tier mapping approach. Our algorithm introduces an intermediary phase with a significantly higher bin count, specifically, 4096 bins. In this phase, the flux data points are mapped to these 4096 bins by the modular residuals of their timestamps. Following this, a merging process is initiated to consolidate these 4096 bins down to the intended 256 bins. The criteria of this merging is to ensure that each of the 256 bins contains an identical number of flux data points. As an example, given a light curve comprising 65 536 data points, each of the 256 bins will accommodate 65536/256 = 256 data points. Note that in our two-tier mapping approach, it is permissible for some of the initial 4096 bins to have a lower number of samples or even be empty. This is because the second-tier mapping effectively consolidates these intermediate bins into the final set of 256 bins, and then calculates the average flux for each.

In summary, the initial phase of mapping to the 4096 bins acts as a near-perfect emulation of sorting. Subsequent rebinning to the 256 bins produces homogeneous noise in the binned data to minimize false signals. Noise reduction is further achieved by averaging the flux values within each bin. Note that this rebinning of the 256 bins might introduce minor time step differences on certain occasions, but it does not lead to any missed signal detection. This was verified using two methods: first, through visual inspection of the folded light curves, we confirmed that the outcomes from the two-tier mapping closely resemble those from the traditional sorting-based method. Secondly, by testing known USP planets in the *Kepler* catalogue, we confirmed that our two-tier mapping approach successfully detects transits at the same periods as identified by the traditional method.

We compared our method with the traditional sorting-based phase folding method and confirmed that our approach exhibits no compromise on accuracy while providing a significantly faster performance by leveraging GPU parallelization.

Thus, as illustrated in Fig. 4, the resulting GPU-optimized phase folding process – which is a mathematical near-equivalent to typical phase folding – is composed of the following five steps. First, we calculate the modular residuals of the timestamp of each light-curve data point modded by a given folding period. Secondly, we map the modular residuals to an intermediary 4096 bins. Thirdly, for each of the 4096 bins, calculate the number of flux data points and the sum of the flux values in that bin. Fourthly, scan the 4096 bins from left to right, combine the adjacent bins, and split bins on the boundary as needed, to generate a final layout of 256 bins. This step reallocates the flux data points evenly from the intermediate 4096 to 256 bins, with each of the 256 bins containing the same number of flux data points. This step needs to be done sequentially but it is fast because it only needs to convert from 4096 to 256 bins instead of converting from 64k light-curve data points to 256 bins. Lastly, we calculate the average flux values for each of the 256 bins, dividing the sum of flux values by the number of data points for each bin.

For each *Kepler* light curve, the GPU phase folding algorithm applies this phase folding process at each of the 100 000 trial periods.

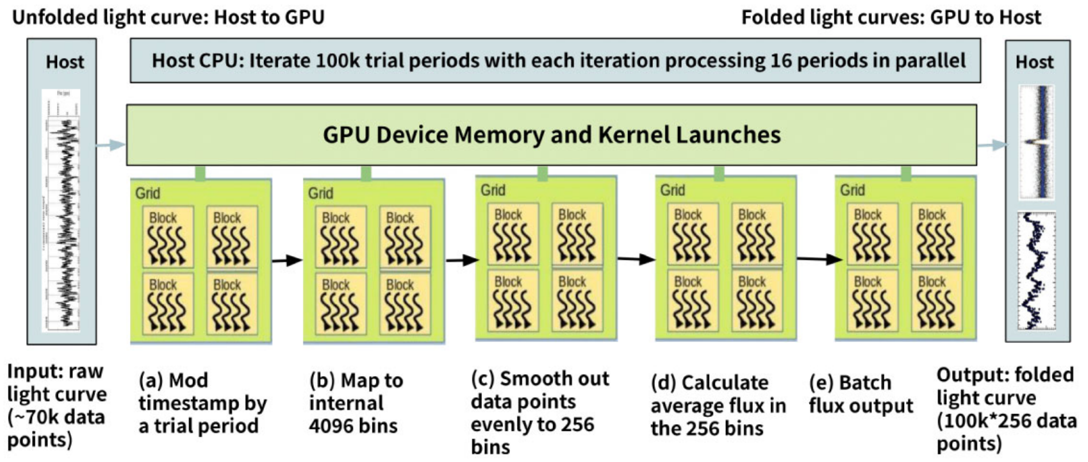

Unfolded light curve: Host to GPU
Folded light curves: GPU to Host
Host
Host CPU: Iterate 100k trial periods with each iteration processing 16 periods in parallel
Host
GPU Device Memory and Kernel Launches
Grid
Block
Input: raw light curve (~70k data points)
(a) Mod timestamp by a trial period
(b) Map to internal 4096 bins
(c) Smooth out data points evenly to 256 bins
(d) Calculate average flux in the 256 bins
(e) Batch flux output
Output: folded light curve (100k*256 data points)

Figure 4. The GPU phase folding module. The GPU phase folding algorithm consists of five steps, each of which is optimized through parallel computing. First, (a) the timestamps of the light-curve data points are modded by the trial period. Then, (b) the full time span of the trial period is evenly split into 4096 equally spaced bins, and the data points are mapped into the bins based on the value of their time residuals from (a). Next, (c) the 4096 bins are rebinned into 256 bins, such that each of the 256 bins contains an equal number of data points. The flux values of the data points in each bin are averaged in (d) and filled into its final form, 100 000 × 256 data points, in (e). The initial mapping to the 4096 bins in (b) serves as a near-perfect emulation of sorting, and the subsequent mapping to the 256 bins and averaging in (c) and (d) create homogeneous noises in the binned data to minimize false signals and achieve noise reduction. The algorithm takes advantage of the GPU's blocks and threads structure such that multiple (p) trial periods are folded at the same time. (p was 16 with the GPU we used but it can be more with an advanced GPU with additional memory and thread parallelism.) In other words, for each of the steps from (a) to (e), there are p of that process executing at the same time.

The algorithm divides the 100 000 trial periods into batches of 16, processing each batch in parallel until all periods are completed. With the Nvidia GeForce RTX 3090 Ti card that is used in our research, we let the GPU kernel execute 16 periods simultaneously, which uses close to the entire 11 GB GPU device memory on the card. The final output of the GPU phase folding algorithm is a data array with dimensions 100 000 × 256, which represents the folded result at each of the 100 000 evenly spaced trial periods. The GPU phase folding algorithm takes ∼5 s to finish 100 000 folds for a typical *Kepler* light curve.

2.4 GPU phase folding performance tuning

Because the performance of the GPU algorithm is key to our GPFC system, we exploited various techniques to maximize parallelism and optimize its performance. Although we have briefly touched on some of the performance tuning, below is a description of the techniques used.

(i) *Maximization of GPU kernel's parallelism in the phase folding algorithm.* Our fast GPU phase folding algorithm was composed of a series of GPU kernel function launches. Nvidia GeForce RTX 3090 Ti GPU card supports a maximum of 65535 × 65535 × 65535 block-n-threads parallelism. We used the first dimension to parallelize across the 100 000 periods, with 16 phase folds conducted simultaneously. For each of the functions of which the GPU phase folding algorithm consists, we assigned the other two dimensions on the rawlight-curve data points such that they are all processed by the GPU kernel simultaneously with its thread pool.

(ii) *Minimization of memory transfer overhead between CPU and GPU.* Although GPU-based executions are fast, the data transfer between CPU and GPU is often an expensive step. To reduce memory transfer overhead, we employed a zero-memory-copy technique, and batched small memory transfers to maximize memory bandwidth utilization. We also minimized the GPU memory read–write time by maximizing the amount of parallel memory accesses. These memory optimizations reduced memory transfer time from 10 to 0.25 s in our algorithm.

(iii) *Utilization of atomic read-write operations to speed up thread serialization.* With significant concurrent execution of numerous GPU kernel threads, it is imperative to implement thread serialization when altering data stored at a shared memory location. To address this requirement, we leveraged the inherent atomic operation hardware support of the NVIDIA GeForce RTX 3090 Ti card as our chosen synchronization tool. Notably, the utilization of atomic operations resulted in a 12-fold speed increase in comparison to the conventional locking mechanism.

Table 1 shows a breakdown of the final runtime of our GPU phase folding algorithm. As seen in this example, the program spent most of the time on memory read/write operations within the GPU device (3.66 s), then on calculations within the GPU device (0.66 s), followed by the batch memory transfer from the GPU card to the CPU host (0.25 s), then by memory allocation and free operations, mostly spent on the GPU device (0.18 s). Lastly, initializing the program on the host CPU took a small time slice of 0.042 s. The total execution time amounted to 5.00 s. Note that this measured duration pertains to the folding and binning of a light curve consisting of 70 000 epochs. The execution time is subject to slight variations depending on the length of the *Kepler* light curves, with a marginal reduction in duration for shorter light curves. Roughly speaking, the maximum running time for processing *Kepler* light curves is around 5 s.

2.5 Deep neural network

After the 100 000 folds of a light curve are noise-normalized, as described in Introduction, they are ready to be fed into the CNN to discern whether they contain a transit signal at any of the trial periods.

The design of our CNN architecture consists of 19 total layers as illustrated in Fig. 5. To start with, the initial 1D input data undergoes reshaping into a 2D tensor, preparing it for subsequent

Table 1. GPU phase folding performance profiling breakdown. In this example, the longest time of the GPU program was spent on memory read and write operations within the GPU device (3.66 s). The second longest time was spent on calculations in GPU kernel functions (0.66 s), followed by the batch memory transfer from the GPU device to the CPU host (0.25 s), then by memory allocation and free operations, mostly spent on the GPU device (0.18 s). Additionally, initializing the program on the host CPU took a small time slice of 0.042 s. The total execution time amounted to 5.00 s.

GPU phase folding performance profiling	
Runtime breakdown	Time (s)
Memory reads and writes on GPU device	3.66
Running GPU Kernel functions	0.66
Memory transfer from GPU to host	0.25
Total memory allocation/free (host and GPU)	0.18
Memory allocation on GPU (heap)	0.17
Memory allocation on host (stack)	0.0097
Memory free on GPU (heap)	0.00058
Read input data files on host	0.11
Write output to disc on host	0.098
Initialization on host CPU	0.042
Total time	5.00

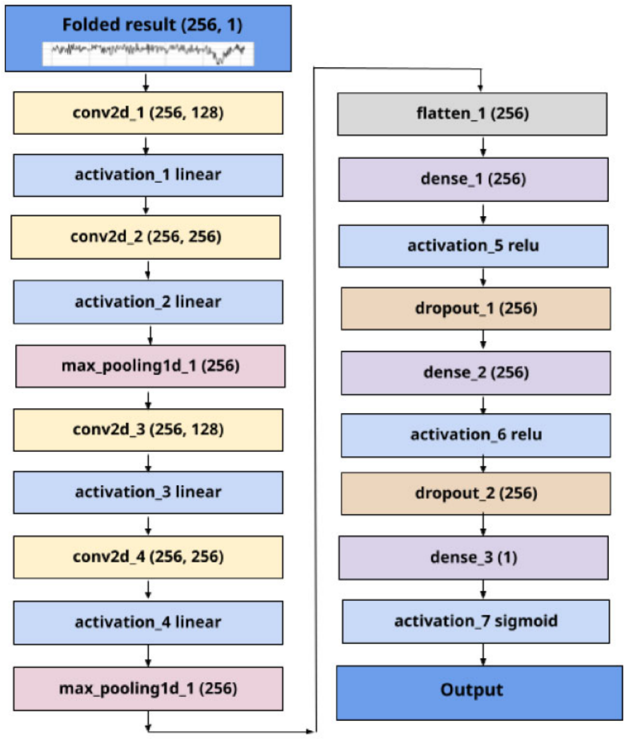


Figure 5. The CNN module. As input, the CNN takes a single noise normalized 256-length light -urve fold and the CNN outputs a confidence score that the folded input contains a transit signal.

2D convolutional operations. Taking into account the periodic nature of the data, we design a model incorporating a circular convolution layer, a convolution with edge wrapping sourced by Schubert et al. (2019). The initial circular convolution layer employs 128 filters, followed by another with 256 filters. Subsequently, the 2D tensor is reverted to its original 1D form, and a global-max-pooling layer is introduced to retain the most significant values. Following this, the same sequence of operations is repeated, this time utilizing larger kernel sizes to capture broader spatial patterns within the data. Afterwards, the data are flattened and passed through fully connected layers with the rectified linear unit (ReLU) activation function. Dropout layers are incorporated to mitigate overfitting during the training process. Ultimately, a dense layer with the sigmoid activation function furnishes the model's prediction score. The output of the model is a probability score representing the likelihood that the inputted data contains a planetary transit signal. We construct our model on the top of the open source TensorFlow library (Abadi et al. 2016), and we use the Adam optimization algorithm (Kingma & Ba 2015) to minimize the cross-entropy error function. We train the neural network with a learning rate of 10^{-6} and a batch size of 32 across 90 epochs. We train the neural network with a learning rate of 10^{-6} and a batch size of 32 across 90 epochs. The final parameters were selected based on approaches used in similar applications, along with experimentation, validation, and testing. For future work, hyperparameter tuning methods such as grid search or random search can be employed to systematically explore further optimizations.

2.6 Light-curve simulation

The SNR of a planetary transit detection in a given light curve can, in the simplest case, be approximated by formula (2) (von Braun & Ciardi 2007),

$$\alpha = \frac{d}{\sigma}\sqrt{n\frac{L}{p}}, \tag{2}$$

where d is the transit depth, σ represents the photometric measurement uncertainty in relative flux per data point, assuming it is the same for all data points. p is the transit period, and L is the transit duration. n equals the total number of data points in observation, therefore, equivalently, $n\frac{L}{p}$ equals the number of data points observed during transits. The assumption of this equation is that there exists only white noise and no statistically correlated (red) noise.

To create simulated light curves as realistic as possible, we studied the statistics of the parameters of real *Kepler* light curves and used them as guidance for the simulation. Since our objective is to simulate light curves generated by USP exoplanets, we based our simulation on the parameter distributions observed in the 43 confirmed USPs listed in the KOIs in the Kepler Input Catalog (KIC), as depicted in Fig. 6.

Specifically, we measured the following parameters: orbital period, transit duration, transit duration over orbital period, and the relative flux 1σ uncertainty in the detrended light curves. From the confirmed USPs, we observe that the parameter distributions are as follows: orbital period $p \in_R (0.3, 1.0)$ d, standard deviation of the noise $\sigma \in_R (0.0001, 0.0005)$, transit duration $L \in_R (0.03, 0.09)$ d, and transit duration over orbital period $\frac{L}{p} \in_R (0.04, 0.12)$. These distributions are listed in Table 2.

We simulate a transit signal with a trapezoidal model as shown in Fig. 7. The trapezoidal parameters are determined based on the real USP parameter distributions gathered above. The trapezoid ratio between the short base and the long base is a random number $r \in_R (0, 1.0)$. The transit duration is measured at the half-depth of the trapezoid. Given a simulation SNR target value α, the transit signal depth d can be derived from formula (2).

2.7 Synthetic data set for neural network training

As the 43 confirmed USPs in the *Kepler* survey do not constitute a sufficiently large data set for effective CNN training, we created an extensive training data set, comprising two million synthetic light curves divided into one million transit and one million non-transit

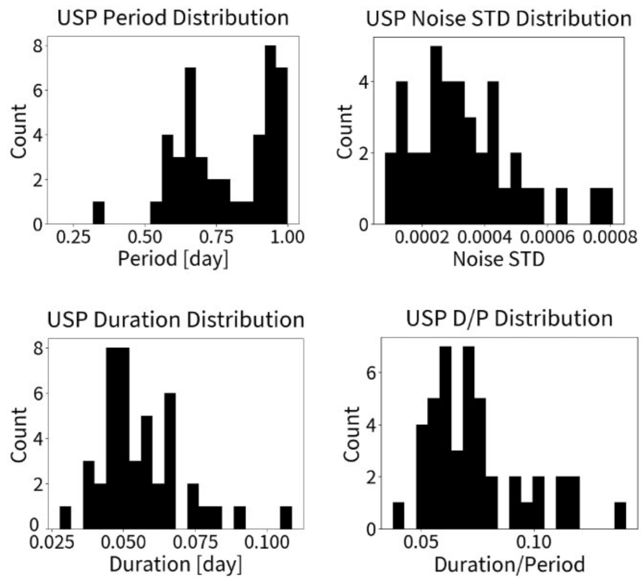


Figure 6. The parameter distributions for the 43 confirmed USPs from the KOI catalogue. (a) Period, (b) noise, (c) duration, and (d) duration/period. These parameters are used as a guide for our simulation of light curves.

Table 2. USP parameter distribution gathered from the 43 confirmed USPs in the *Kepler* KOIs.

USP transit parameter distribution in *Kepler*	
Orbital period (d)	$p \in_R (0.3, 1.0)$
Transit duration (d)	$L \in_R (0.03, 0.09)$
Transit duration to period ratio	$\frac{L}{p} \in_R (0.04, 0.12)$
Standard deviation of noise	$\sigma \in_R (0.0001, 0.0005)$

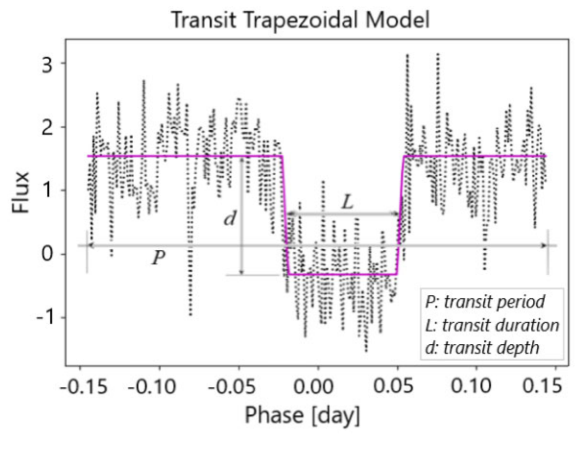


Figure 7. Trapezoidal model. A trapezoidal shape is incorporated into Gaussian noise to simulate planetary transit signals within a light curve. The input parameters for the trapezoid are determined based on the actual USP parameter distributions extracted from *Kepler* KOIs. P and L denote the orbital period and the transit duration, respectively. The transit depth, denoted as d, is computed given a specific SNR, while the ratio between the trapezoid's short and long base is randomly chosen $r \in_R (0, 1.0)$.

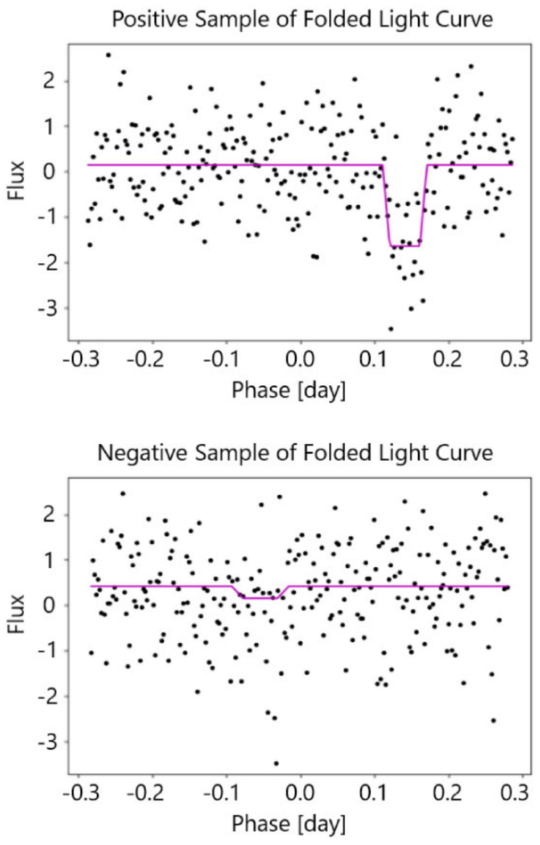


Figure 8. Simulated folded light curves for CNN training. The upper plot demonstrates a transit sample generated by injecting a trapezoidal model into Gaussian noise with period $p = 0.569975$, duration $L = 0.0593157$, and SNR $\alpha = 9.2$. The lower plot demonstrates a non-transit sample generated by injecting a low SNR trapezoidal signal to Gaussian noise with period $p = 0.386515$, duration $L = 0.0528367$, and SNR $\alpha = 1.6$.

samples. Each sample set consists of vectors with a length of 256, simulating folded light curves that the CNN is designed to process.

In a foundational model, Gaussian noise can be employed to simulate non-transit samples, while the injection of a transit signal into Gaussian noise yields transit samples. However, in practice, Gaussian fluctuations can occur especially when light-curve segments are repetitively stacked or folded at a specific period. These fluctuations can sometimes resemble transits with low SNRs. Therefore, in addition to pure Gaussian noise, we intentionally introduce non-transit samples by injecting transits with low SNRs. This method optimizes the CNN's capability to differentiate between significant Gaussian fluctuations and genuine transits. Empirically, we chose a cut-off SNR of 5; training data produced with a corresponding SNR greater than 5 are labelled as positive, while those with an SNR below 5 are labelled as negative. We also confirmed that our CNN trained with an SNR cut-off of 5 misclassifies Gaussian noise as positive signals in fewer than 0.001 per cent of instances.

To create transit samples, we inject a trapezoidal model into randomly generated Gaussian noise of length 256. And we generate transit samples in the range of SNR $\alpha \in_R [6, 10]$. Fig. 8 demonstrates the generated transit and non-transit samples, respectively.

Our synthetic data set represents a simplified model of the complex noises found in actual light curves, which include correlated noise from various sources as well as Gaussian noise. This data set is confined to scenarios involving planet transits and non-transits, omit-

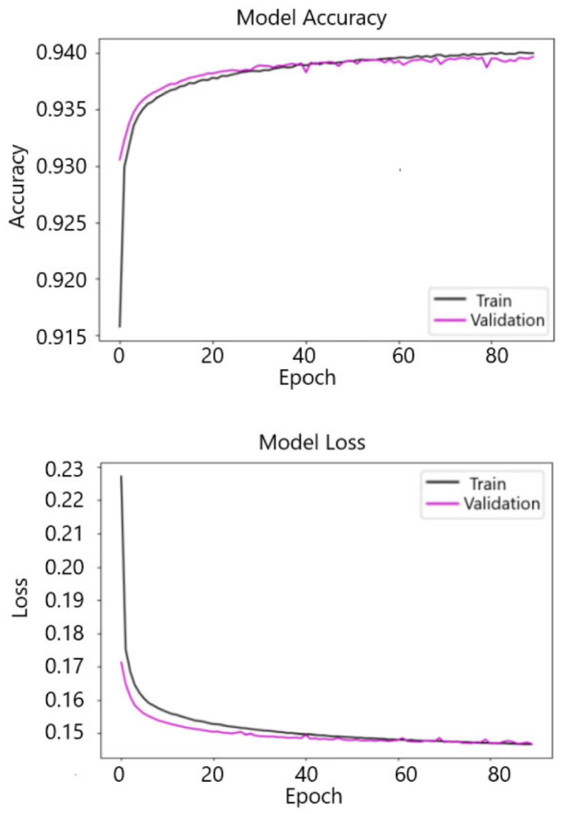


Figure 9. The training curves (accuracy and loss) for the CNN model on the training and validation sets, respectively, across 90 epochs.

ting signals characteristic of other common false positive situations. However, considering that the primary goal of this simulation test is to assess the timing performance of our method, this simplified model is adequate for our purposes. We used the same data set for BLS comparison, ensuring fairness in evaluating relative differences. Nevertheless, it is worth noting that the absolute performance indicated by this model is likely more optimistic than what would be observed with real light-curve scenarios.

By following this process, we generate a data set comprising one million transit and one million non-transit samples, which are subsequently randomly partitioned into training (80 per cent), validation (10 per cent), and testing (10 per cent) sets. Following the neural network training, our best model achieves an accuracy of 94.0 per cent. For a detailed view of the training progress, please refer to Fig. 9, which presents the complete training curve across the 90 epochs.

3 COMPARISON

In this section, we conduct a comparative analysis between the GPFC and the BLS methods, using simulated unfolded *Kepler* light curves to evaluate their characteristics and performance metrics.

To closely replicate real *Kepler* light curves, all the light curves we generated have a length of $n = 65\,536$ epochs and a time interval of 0.0204 d (∼29.4 min) between consecutive data points.

We use Gaussian noise to simulate light curves that do not contain transits. Conversely, for simulating planetary transits within the light curve, we inject a series of trapezoidal shapes into Gaussian noise. These trapezoids are parametrized based on the provided values for orbital period, transit duration, transit epoch, and SNR.

3.1 The Box-fitting Least Squares periodogram

To make a comparison with our GPFC method, we employ the widely recognized BLS algorithm developed by Kovács et al. (2002). BLS is a periodogram that phase folds a light curve and fits a transit-like rectangular box, aiming to minimize the fitting squared error. BLS searches for the transit signals on the frequency space, and provides a periodogram power value for each trial frequency. To evaluate the likelihood of a transit-like signal existing in a particular light curve, a signal detection efficiency (SDE) score is computed, as illustrated in the following equation (Kovács et al. 2002):

$$\mathrm{SDE} = \frac{P(f_{\max}) - \langle P \rangle}{\mathrm{sd}\langle P \rangle}, \tag{3}$$

where P is the BLS periodogram function, $f_{\max}$ is the frequency at which the highest power occurs, $\langle P \rangle$ is the mean of the periodogram powers, and $\mathrm{sd}\langle P \rangle$ is the standard deviation of the periodogram powers.

In short, given all the periodogram powers across the frequency space, the SDE value represents the number of standard deviations above the mean of the maximum power. The BLS method also takes parameters including the number of bins, the frequency range, and the range of the fractional duration over the period. In our testing, we set the number of bins used by BLS to 256. The specifications for the frequency range and the range of fractional duration over the period are derived from the distributions illustrated in Fig. 6, adopting the same criteria as used for the GPFC method.

3.2 Accuracy of GPFC versus BLS

Next, we analyse the capability of the two methods to detect exoplanet signals in light curves. We posit that the underlying mechanisms that GPFC employs makes it a more powerful detection system because (1) our model evaluates light curves with a deep neural network, which, if trained properly, has the capability for stronger spatial awareness in recognizing transit-like signals than a least-squares approach, and (2) our neural network is trained on a trapezoidal model with white noise, which is a closer representation of a true transit shape than a box-car function.

To assess the efficacy of the GPFC approach relative to the BLS method, we evaluate their capabilities to distinguish between true and false signals that we simulated across a range of SNRs. For BLS, this capability is reflected by the distinction in the SDE scores assigned to transit versus non-transit light curves; for GPFC, it is represented by the disparity in maximum model scores between transit and non-transit light curves. For each method, by selecting various thresholds to demarcate transit from non-transit predictions, we can construct a Receiving Operating Characteristic (ROC) curve. The more powerful a given classifier is, the more its ROC curve will be higher and toward the left (Fig. 10), denoting a larger true positive rate (TPR) and lower false positive rate (FPR) across different cut-offs. Thus, we can use the area under the curve (AUC) of the ROC as an overall representation of how accurately a model distinguishes true and false signals. Fig. 11 illustrates that the AUC of GPFC, which is employed with 100 000 trial periods, outperforms that of BLS, demonstrating that GPFC is a stronger general classifier of transit signals than BLS. The precision–recall results in Fig. 12 similarly show that GPFC exhibits higher performance ability in the trade-off between TPR and FPR.

While AUC represents overall ability to distinguish across the TPR versus FPR trade-off, in practical research it is often the case that either the TPR or the FPR is given prioritized importance as a

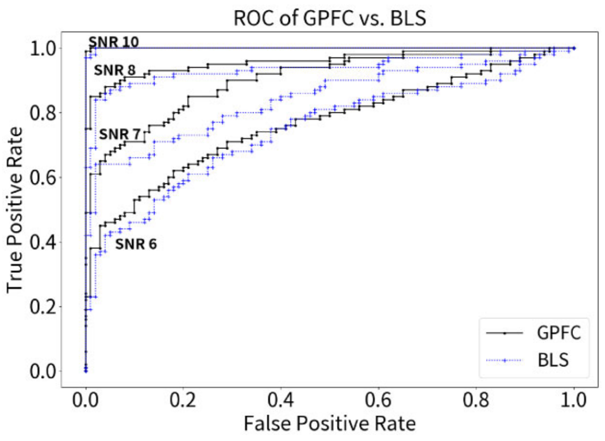


Figure 10. The ROC curves for GPFC and BLS, evaluated at 100 000 frequencies, span SNRs of 6, 7, 8, and 10. As the threshold for classifying transit and non-transit outputs varies, there is a trade-off between the TPR and the FPR, shown by the ROC curve. The GPFC method shows stronger ability to distinguish transit and non-transit light curves for SNR below 10.

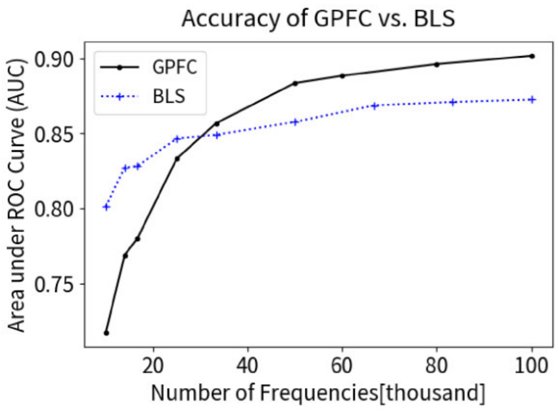


Figure 11. The accuracy of GPFC and BLS as the number of trial frequencies vary. Here, accuracy refers to the ability to distinguish transit and non-transit light curves, as represented by the area under the ROC curve.

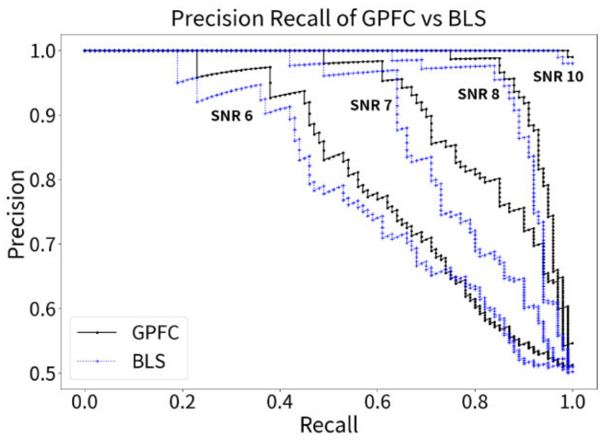


Figure 12. The Precision–Recall curves for GPFC and BLS, evaluated at 100 000 frequencies over SNRs 6, 7, 8, and 10 are shown. The GPFC method achieves higher precision than BLS at the same recall in the curves with SNR below 10.

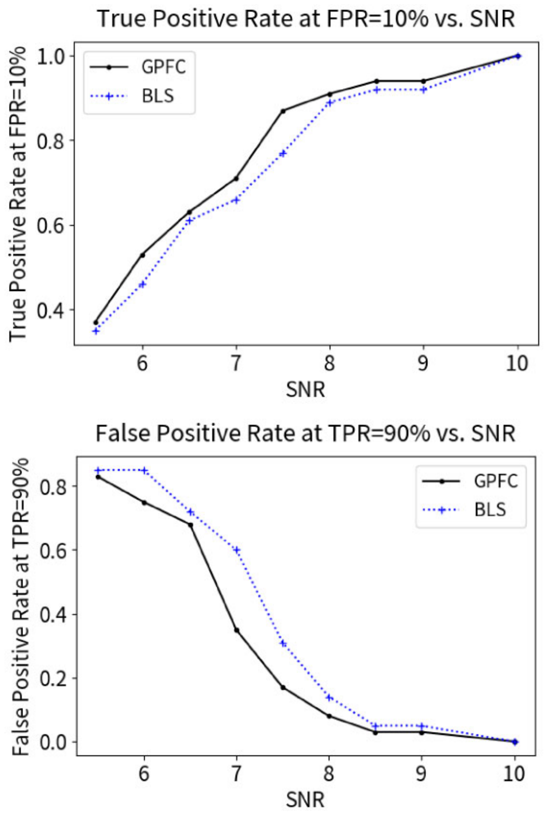


Figure 13. Performance comparison of GPFC and BLS in two scenarios. (a) At the same FPR of 10 per cent, the recall is higher for GFPC than BLS for SNR between 5.5 to 10. (b) At the same high recall such as 90 per cent, the FPR is lower for GFPC than BLS for SNR between 5.5 to 10. GPFC outperforms BLS in terms of both metrics: TPR when FPR is at 10 per cent, and FPR at TPR of 90 per cent.

result corresponding to the specific use-case. Therefore, we illustrate a second comparison between GFPC and BLS in Fig. 13, splitting off these two types of situations. When the FPR is prioritized (10 per cent FPR), the resulting TPR is shown in the top panel of Fig. 13, whereas when the TPR is prioritized (90 per cent TPR), the resulting FPR is shown in the bottom panel of Fig. 13. GPFC shows a distinct advantage over BLS in both metrics: the TPR at 10 per cent FPR, and the FPR at 90 per cent TPR.

3.3 Speed of GPFC versus BLS

The GPU phase folding algorithm takes ∼5 s per light curve with length 65 536, and the CNN takes ∼6 s to evaluate the 100 000 folded results. Thus, to process one light curve with the GPFC method takes ∼11 s. Note that when using the GPFC method to process a large number of light curves, the speed is reduced to 6 s per light curve because the CNN and GPU phase folding algorithm can be run simultaneously – as the GPU phase folds a light curve, the CNN can process the previously folded light curves in parallel. With the 6 s per light-curve speed, the entire catalogue of the 150 000 *Kepler* main-sequence stars can be searched in just over 10 d. On the other hand, as Kovács et al. (2002) mention, the performance of the BLS periodogram varies based on the number of frequencies scanned in its frequency space. The runtime for BLS with the same search accuracy on the same light curves would be three orders of magnitude higher. To speed up the runtime comparison, we further assessed a fast BLS

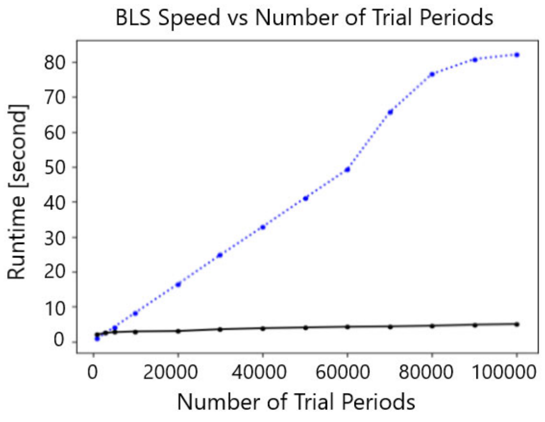


Figure 14. The speed of BLS versus the number of trial periods searched in the USP period range of [0.2, 1] d. The dotted line shows a roughly linear relationship between the runtime of BLS for one light curve as the number of trial periods searched varies. The fast CYTHON-accelerated BLS implementation provided by ASTROPY is used for runtime comparison, which takes 80 s of runtime at 100 000 trial periods. On the other hand, the GPFC method processes one light curve in 6 s at 100 000 trial periods, which is 15 times faster.

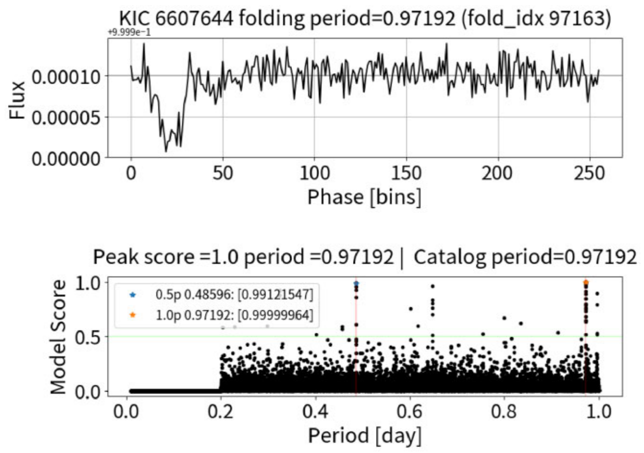


Figure 15. An example of a real *Kepler* USP (KID 6607644) recovered by the GPFC method. The phase fold at the detected transit period of 0.971923 d is shown in the upper plot. In tandem, the model score versus trial period plot (the lower plot) shows a clear peak at period 0.971923 d with model score of 0.99999964. This peak period accurately matches the known transit period of 0.971916 d in the *Kepler* catalogue (error 7.362×10^{-6}). The model scores also reveal corresponding peaks at the harmonics of the transit period. For instance, at the half-period 0.48596 d, the model reaches a high score of 0.99121547.

implementation provided by ASTROPY (docs.astropy.org), which is accelerated by CYTHON. Fig. 14 shows the ASTROPY implementation have a runtime of ∼80 s at frequency of 100 000, which is 15 times slower than GPFC.

The performance advantage of GPFC over BLS is attributed to a few factors built in the design of the GPFC system. To search for a flux 'dip' in a light curve, the BLS algorithm constructs a three-level nested loop to exploit an exhaustive scan over all possible transit start positions, transit durations, and all trial frequencies. By using a CNN to vet light curves, we eliminate the two loops of exhaustive search over transit start positions and transit durations, leveraging CNN's advantage to recognize objects that contain spatial structure. For the loop over trial periods, the GPU-based folding algorithm, the GPU's capability for parallelism allows it to phase fold multiple frequencies at the same time, greatly boosting performance.

4 GPFC APPLIED TO REAL *KEPLER* LIGHT CURVES

4.1 *Kepler* light-curve processing

Next, we confirm that GPFC works on real *Kepler* light curves. For this study, we downloaded the light curves from the Q1–Q17 *Kepler* Data Release 25 (DR25), made available by the *Kepler* mission through the Mikulski Archive for Space Telescopes. We processed the downloaded light curves with the pre-processing method described in Section 2.2.

We also used the set of KOIs available on the NASA Exoplanet Archive as of 2023 June 3. There are 9564 total dispositioned KOIs with Data Validation (DV) light curves available. These include 2350 confirmed planets, 2366 candidates, and 4848 false positives. Transit parameters (period, epoch, and duration) taken from the KIC catalogue are used for testing and verification. We downloaded all 9564 KOIs in the KIC.

As a start, we chose a subset of the KOIs which contains only target stars whose transit events are all labelled as 'CONFIRMED' planetary transits. This filters out all light curves whose target star contains a transit event with *Kepler* disposition of 'FALSE POSITIVE' or 'CANDIDATE'. This process is needed to ensure that none of the target stars we include are affected by undesirable false positive interference, or contain secondary eclipses of eclipsing binaries. The metadata for secondary eclipses are not recorded in the KOIs and therefore cannot be fully masked when needed.

From this KOI subset, we prepared two sets of light curves to test the GPFC system: one transit and one non-transit data sets. The transit data set was composed of light curves of all target stars that contain transits of a confirmed USP planet. For each such star, the light curve was conditioned by keeping the USP transits intact while removing the transits from any other planets. The non-transit data set consisted of light curves of all target stars with all of their transit events masked.

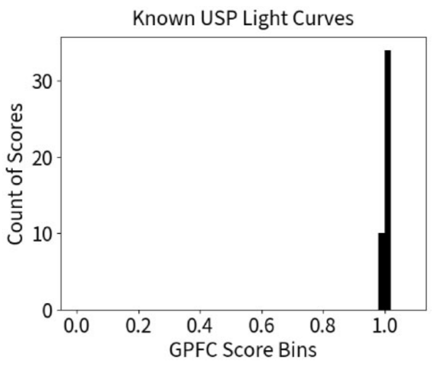
Through this process, we obtained 43 light curves with verifiable USP transits for the transit data set, and 1437 light curves devoid of any transit events for the non-transit data set.

4.2 Recovery of all confirmed *Kepler* ultrashort-period exoplanets

By applying the GPFC method to the transit data set, it has been demonstrated that GPFC accurately identifies all of the 43 confirmed USPs. The method recovers these exoplanets at the periods within 0.004 per cent of the recorded value in the KOIs as shown in Figs A1 and A2 in Appendix A. Additionally, all confirmed USPs are recovered with a score of 0.99 or higher, further reinforcing the validity of the GPFC method. This crucial validation lays the foundation for using GPFC to uncover new exoplanets in the *Kepler* database, which will be elaborated in our follow-up paper (Wang et al. 2024, submitted).

Fig. 15 demonstrates an example USP that the GPFC method recovered with a high probability score.

In Fig. 16, the upper plot shows the distribution of the GPFC model scores for the 43 confirmed USPs, and the lower plot shows the score distribution for all of the non-transit light curves. The score for each light curve is determined by identifying the optimal



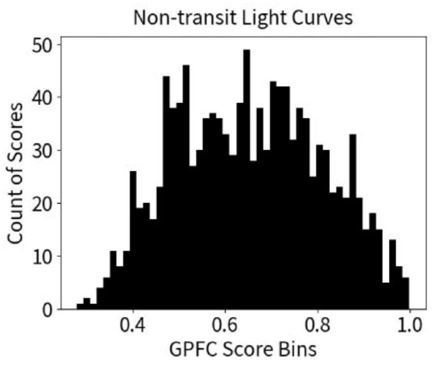


Figure 16. The performance of GPFC on real *Kepler* light curves. As shown in the upper panel, GPFC distinguishes all confirmed USPs with scores above 0.99. The lower panel represents GPFC's scores on real *Kepler* KOI with known transits masked. We use the *Kepler* KOI with only confirmed planets to avoid the secondary eclipses of eclipsing binaries. For the 1437 *Kepler* KOIs with known transits masked, signals that received high scores were further evaluated as potential candidates. We set a cut-off CNN score threshold of 0.8 for this test, aligning with a 96 per cent TPR as indicated in our simulation test.

peak among the 100 000 scores that exceed a height of 3σ and have a width of more than three data points. We see that most of the non-transit light curves received scores below 0.9. The candidates on non-transit light curves which the model identified with scores above 0.8 identified in non-transit light curves will be further examined in our follow-up investigation. Our simulation results indicated that CNN scores of 0.76, 0.8, and 0.9 correspond to TPRs of 90 per cent, 96 per cent, and 98 per cent, respectively. Based on these findings, we selected 0.8 as the threshold for this test. For the validation of real candidates, we implemented a pipeline comprising a series of rigorous verification tests. This validation procedure led to the identification of five promising candidates, which will be elaborately discussed in our upcoming follow-up paper.

Fig. 17 illustrates the high-probability (probability greater than 0.9) false positives detected by the GPFC method. In our analysis of 1437 *Kepler* light curves, we identified several false positives: 10 due to sinusoidal signals, exemplified by KIC 9 642 292 in plot (a), and 32 resulting from longer-term stellar variation, as illustrated by KIC 10 717 220 in plot (b). Additionally, two cases showed significant noise from unrelated sources, represented by KIC 5 621 125 in plot (c). These 44 false positives were manually vetted and subsequently discarded. Looking ahead, we anticipate that such false positives could be automatically filtered out in future work through improved pre-processing procedures, eliminating the need for manual intervention.

In contrast, plot (d) (KIC 9963524) and plot (e) (KIC 5640085) present transit-shaped signals; however, given the noisy nature of the light curves, these detected signals are subjected to further false alarm probability (FAP) checks in the verification process. This process, which operates without human intervention, aims to eliminate most of the false positives originating from extraneous sources (such as exemplified in plots d and e). The details of this verification process will be discussed in our forthcoming discovery paper.

5 DISCUSSION AND CONCLUSION

5.1 Real exoplanet discovery

For the purpose of validating the GPFC approach with actual *Kepler* light curves, we confined our work to target stars exclusively associated with confirmed planetary transit events. As a subsequent step, we intend to employ GPFC for real exoplanet discovery, encompassing the entire *Kepler* catalogue.

The design of the GPFC system is inherently tailored to detect shallow and narrow transit signals, combining high-precision phase folding with deep learning techniques. Consequently, it holds potential for uncovering smaller exoplanets. Our preliminary runs of GPFC on real *Kepler* survey data indicate its capability to quickly process vast data sets. Moreover, it has exhibited proficiency in identifying exoplanet transit signals with lower SNRs, which might have previously gone unnoticed. Our team plans to continue exploring this avenue and will provide updates on our findings in an incoming publication (Wang et al. 2024, submitted).

5.2 GPFC versus the fBLS method

Shahaf et al. (2022) introduced a fast model of BLS, termed fBLS, which incorporates an FFA and a transit detection approach optimized for runtime. This fBLS system reported the discovery of six candidates: KIC 6293500, 9217391, 9835433, 2718885, 6359893, and 11187332. To further validate the accuracy of GPFC, we applied GPFC to these six candidates and all of them were accurately identified with periods precisely matching those reported by fBLS. Both the GPFC and fBLS methods have similar runtimes of approximately 6 s analyse a typical light curve comprising 65 536 data points.

In the folding phase, GPFC is designed to directly process the original *Kepler* light curves, which are irregularly sampled with sporadic time gaps. This is achieved by folding based on the original measurement time modulo the trial period. In contrast, due to the requirement of uniform sampling by the FFA, the fBLS approach needs to initiate its analysis with a 'brute-force' folding procedure to address the irregular samples. Concurrently, the fBLS algorithm folds the light curve using approximately 250 000 trial periods within the interval [0.2, 1.0] d. Our simulation tests, as evidenced by the ROC curve, indicated that GPFC, when folded with 250k periods, exhibited performance similar to that observed when folding with 100k periods (AUC values are 0.901 and 0.902, respectively), so we opted to retain our configuration at 100k trial periods.

During the transit detection phase, GPFC uses 256 flux bins to represent the folded light curve in contrast to the 40–80 bins used by fBLS. Whereas fBLS and other optimized BLS algorithms iterate through a pre-selected set of transit start and duration combinations, the GPFC method differentiates itself by incorporating a deep learning module (CNN), to pinpoint the transit at any location and with any duration. Given that the light curves used to train the CNN incorporate randomized transit phases and durations, the

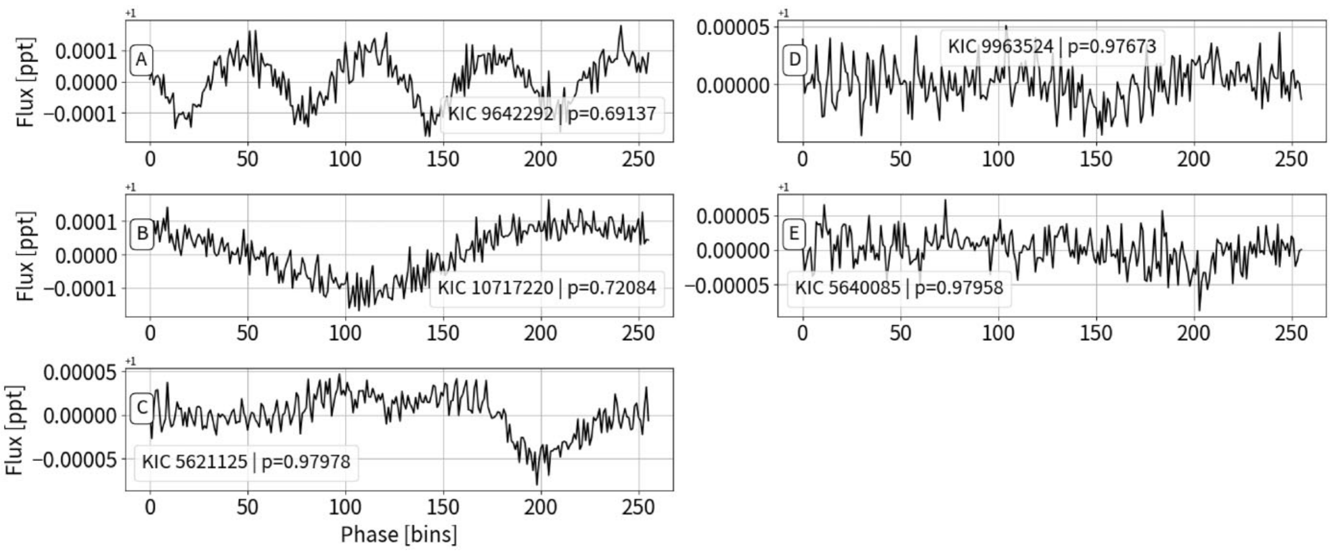


Figure 17. High-probability (greater than 0.9) false positives identified by the GPFC method. The GPFC method identified various types of high-probability false positives. Plot (a) illustrates an example of false positives caused by sinusoidal signals, while plot (b) presents those due to longer term stellar variations. These can be effectively filtered out in an enhanced pre-processing process. Plot (c) exhibits noise from extraneous sources. Conversely, plots (d) and (e) display transit-shaped signals; however, due to the noisy nature of the light curves, the detected transit-like signals require further FAP check in our automated verification process. The specifics of this verification will be detailed in our forthcoming discovery paper (Wang et al. 2024, submitted).

CNN naturally recognizes an expansive spectrum of transits without compromising processing speed.

5.3 GPFC versus a GPU-BLS method

We noted the presence of a GPU-accelerated BLS software on GitHub, termed cuvarbase BLS. Cuvarbase BLS is configured to automatically assign trial frequencies and features a default regular running mode. Moreover, it provides an additional use_fast mode designed for expedited processing, albeit with somewhat reduced functionality. We conducted a performance comparison between GPFC and cuvarbase BLS, using a typical *Kepler* light curve comprising 70k data points. In our evaluation, cuvarbase BLS processed the data in 30 s using its default mode and 1.05 s with the use_fast mode. In contrast, GPFC accomplished the same task in 6 s. The speeds of GPFC and GPU-BLS are comparable, while GPFC demonstrates higher accuracy, with AUC 7 per cent higher than that of GPU-BLS, which is consistent with comparisons to the traditional BLS method.

5.4 Further performance boosting

Although the GPFC method is notably faster than the traditional BLS method, there is still room to optimize its speed without compromising accuracy. As indicated in Fig. 11, the performance enhancement of GPFC starts to plateau at around 60k trial periods. In our research, we consistently used GPFC with 100k periods because our speed is not an issue in most cases. However, for specific scenarios, like aiming to search through the entire *Kepler* catalogue in less than 10 d, achieving a speed faster than the current 6 s might be beneficial. In such contexts, by switching to 60k trial periods, the computation time could be reduced to approximately 2.5 s.

Another approach to enhance the speed of the GPU phase folding is to further parallelize the algorithm. Due to the memory constraints of our GPU, which has a capacity of only 11 GB, we are limited to evaluating a maximum of 16 trial periods. By leveraging a GPU with a larger memory capacity or one supporting higher parallelism for atomic read–write operations, the runtime of our GPFC system will be improved even further.

5.5 Extending the applications of GPFC

While our primary focus in this research has been on USP exoplanets within the *Kepler* survey, the GPFC method is inherently generic. The GPU phase folding algorithm can function with any set of trial periods, Moreover, the CNN is amenable to training on any custom-configured data set. Consequently, the GPFC method is readily adaptable for searching transits across a wider range of periods. Within the period range of 1–10 d, the application of the GPFC method is expected to be straightforward and similar to current practices. For period ranges that exceed 10 d, the grid search space significantly widens, and it would be beneficial to employ additional strategies to preliminarily identify potential regions of interest.

For longer periods, we can conduct a *Kepler* catalogue analysis similar to that used for the USP, to examine the distribution of planet radii and transit durations. This allows us to ascertain the required $N_{\rm sample}$ for equation (1). For larger planets characterized by longer transit durations, $N_{\rm sample}$ can be smaller, suggesting the viability of a less precise search grid. Such an adjustment is anticipated to decrease the runtime, as depicted in Fig. 14. Larger planets usually produce bigger transit signals with higher SNRs, whereas longer periods result in a reduced number of folds in the light curve, subsequently lowering the noise reduction effect achieved through phase folding. Further predictions of our method's performance across these varying scenarios can be made by a thorough analysis of the *Kepler* database in the target period ranges.

Furthermore, another natural progression would be to enhance the final CNN model by incorporating additional metadata, as demonstrated in various CNN models (Malik et al. 2022b). This augmentation is expected to boost performance and potentially reduce the occurrence of false positives.

Moreover, the adaptability of GPFC extends to light curves from a range of other transit surveys, including *K*2, *TESS*, and upcoming missions like *Plato* (as detailed in Rauer et al. 2014, 2022) and the

Earth 2.0 (ET) mission (Ge et al. 2022a,b,c), and beyond. We expect that the pre-processing step of the GPFC method will need to be tailored for each specific survey to suit the unique characteristics of its light curves. Apart from this customization, the GPFC method is broadly applicable across different surveys.

5.6 Conclusion

This paper introduces the GPFC transit signal detection method, examining its ability to detect periodic transits in stellar photometric time-series. GPFC innovatively merges a GPU phase folding algorithm with a CNN, aiming to identify transit signals from small exoplanets directly from original raw light curves without resorting to the traditional BLS transit detection method.

Using simulated light curves based on *Kepler* parameter distributions, we compared the accuracy and performance of the GPFC method against the BLS method. Our focus is primarily on scenarios with low SNRs and we see that for SNR < 10, GPFC has an advantage in performance over BLS.

In terms of computational speed, GPFC holds a significant advantage. To match the performance levels set by GPFC, BLS would need to utilize at least 20 000 frequencies. This requirement makes BLS exponentially slower – approximately three orders of magnitude – compared to GPFC. When operating with 100 000 frequencies and using a fast version of BLS provided by ASTROPY, accelerated by CYTHON, BLS is still roughly 15 times slower than GPFC.

Lastly, when the GPFC method was applied to known *Kepler* exoplanets, it successfully recovered all confirmed planets within the period search range of [0.2, 1.0] d, as covered in this paper. It also assigned low scores to *Kepler* light curves where all transits were masked. During our tests, GPFC consistently identified new, small transit signals in *Kepler* light curves rapidly, showing its potential applications in large-scale transit surveys.

ACKNOWLEDGEMENTS

JG acknowledges the support from the Strategic Priority Program on Space Science of Chinese Academy of Sciences under grant no. XDA15020600. This research has made use of NASA's Astrophysics Data System and the NASA Exoplanet Archive, operated by the California Institute of Technology, under contract with NASA under the Exoplanet Exploration Program. This paper includes data collected by the *Kepler* mission. Funding for the *Kepler* mission is provided by the NASA Science Mission Directorate.

DATA AVAILABILITY

The *Kepler* light curves used in this study can be accessed from the NASA Exoplanet Archive (https://exoplanetarchive.ipac.caltech.edu/cgi-bin/TblView/nph-tblView?app=ExoTbls&config=cumulative).

REFERENCES

Abadi M. et al., 2016, preprint (arXiv:1603.04467)
Beaulieu J.-P. et al., 2006, Nature, 439, 437
Borucki W. et al., 2010, Science (New York, NY), 327, 977
Boufleur R. C., Emilio M., Pacheco E. J., de La Reza J. R., da Rocha J. C., 2014, in Nader H., ed., Proc. IAU Symp. 293, Formation, Detection, and Characterization of Extrasolar Habitable Planets. Kluwer, Dordrecht, p. 410
Caceres G. A., Feigelson E. D., Babu G. J., Bahamonde N., Christen A., Bertin K., Meza C., Curé M., 2019, AJ, 158, 57
Cameron A. C. et al., 2006, MNRAS, 373, 799
Campbell B., Walker G., Yang S., 1988, ApJ, 331, 902
Carter J. A., Agol E., 2013, ApJ, 765, 132
Charbonneau D., Brown T. M., Latham D. W., Mayor M., 2000, ApJ, 529, L45
Chaushev A. et al., 2019, MNRAS, 488, 5232
Chauvin G., Lagrange A.-M., Dumas C., Zuckerman B., Mouillet D., Song I., Beuzit J.-L., Lowrance P., 2004, A&A, 425, L29
Chintarungruangchai P., Jiang I.-G., 2019, PASP, 131, 064502
Cui K., Liu J., Feng F., Liu J., 2021, AJ, 163, 23
Dattilo A. et al., 2019, AJ, 157, 169
Gaia Collaboration, 2016, A&A, 595, A1
Ge J. et al., 2022a, preprint (arXiv:2206.06693)
Ge J. et al., 2022b, Proc. SPIE, 12180, 13
Ge J., Zhang H., Deng H., Howell S., 2022c, The Innovation (Camb), 3(4), 100271
Hartman J. D., Bakos G., 2016, Astronomy & Computing, 17, 1
Howell S. B. et al., 2014, PASP, 126, 398
Jenkins J. M. et al., 2010, ApJ, 713, L87
Kingma D. P., Ba J., 2015, in Bengio Y., LeCun Y., eds, 3rd International Conference on Learning Representations, ICLR, 2015. San Diego, CA, USA
Kovács G., Zucker S., Mazeh T., 2002, A&A, 391, 369
Major M., Queloz D., 1995, Nature, 378, 355
Malik A., Moster B. P., Obermeier C., 2022a, MNRAS, 513, 5505
Malik S. A., Eisner N. L., Lintott C. J., Gal Y., 2022b, in Baydin A. G. , et al., eds, Discovering Long-period Exoplanets using Deep Learning with Citizen Science Labels, NeurIPS 2022: Machine Learning and the Physical Siences Workshop. New Orleans, LA
McCauliff S. D. et al., 2015, ApJ, 806, 6
Ofir A., 2014, A&A, 561, A138
Ofman L., Averbuch A., Shliselberg A., Benaun I., Segev D., Rissman A., 2022, New Astron., 91, 101693
Olmschenk G. et al., 2021, AJ, 161, 273
Osborn H. P. et al., 2020, A&A, 633, A53
Panahi A., Zucker S., 2021, PASP, 133, 024502
Pearson K. A., Palafox L., Griffith C. A., 2017, MNRAS, 474, 478
Rao S., Mahabal A., Rao N., Raghavendra C., 2021, MNRAS, 502, 2845
Rauer H. et al., 2014, Exp. Astron., 38, 249
Rauer H. et al., 2022, in Kereszturi S., Ivanovski & A., eds, European Planetary Science Congress. EPSC2022, Copernicus GmbH at Palacio de Congresos de Granada, Spain, p. 453
Renner S., Rauer H., Erikson A., Hedelt P., Kabath P., Titz R., Voss H., 2008, A&A, 492, 617
Ricker G. R. et al., 2014, J. Astron. Telesc. Instrum. Syst., 1, 014003
Schanche N. et al., 2018, MNRAS, 483, 5534
Schubert S., Neubert P., Pöschmann J., Protzel P., 2019, in 2019 IEEE Intelligent Vehicles Symposium (IV), Circular Convolutional Neural Networks for Panoramic Images and Laser Data. IEEE, New York, p. 653
Shahaf S., Zackay B., Mazeh T., Faigler S., Ivashtenko O., 2022, MNRAS, 513, 2732
Shallue C. J., Vanderburg A., 2018, AJ, 155, 94
Smith J. C. et al., 2012, PASP, 124, 1000
Staelin D., 1969, Proc. IEEE, 57, 724
Stumpe M. C. et al., 2012, PASP, 124, 985
Thompson S. E. et al., 2018, ApJS, 235, 38
Vanderburg A., Johnson J. A., 2014, PASP, 126, 948
VanderPlas J. T., 2018, ApJS, 236, 16
von Braun K., Ciardi D. R., 2007, in Sun Y.-S., Ferraz-Mello S., Zhou J.-L., eds, Proc. IAU Symp. 249, Observational Window Functions in Planet Transit Searches. Kluwer, Dordrecht, p. 93
Wolszczan A., Frail D., 1992, Nature, 355, 145
Yeh L.-C., Jiang I.-G., 2020, PASP, 133, 014401
Yu L. et al., 2019, AJ, 158, 25
Zucker S., Giryes R., 2018, AJ, 155, 147

APPENDIX A: ADDITIONAL FIGURES

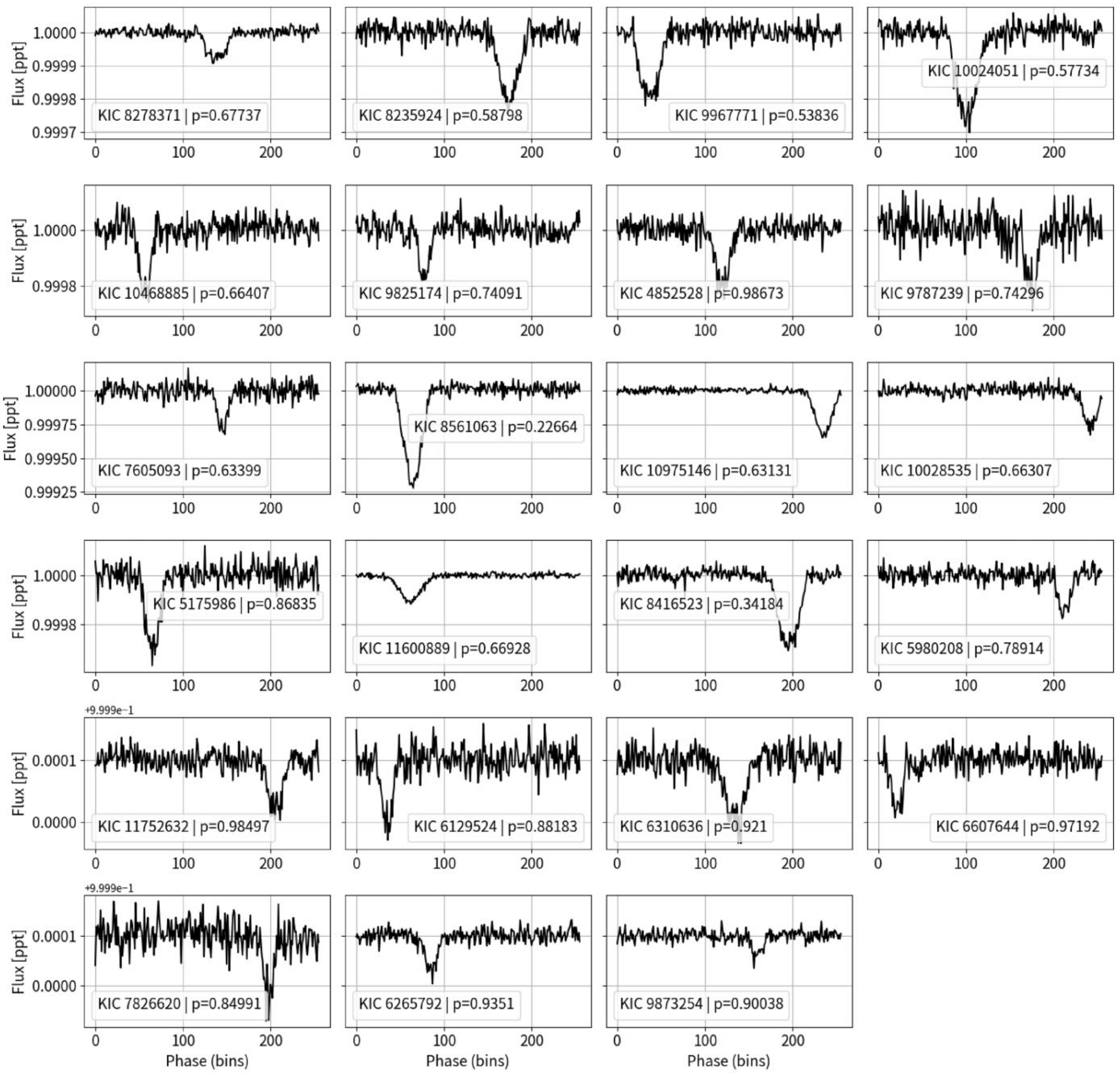


Figure A1. GPFC recovery of confirmed USPs. The GPFC method identified all of the 43 confirmed USPs in *Kepler* with each of them given a CNN score above 0.99. All of the detected periods are the exactly same as the period recorded in the KIC catalogue. Each folded light curve exhibits a clear transit signal. This blind search test proved the validity of the GPFC method in terms of USP transit detection.

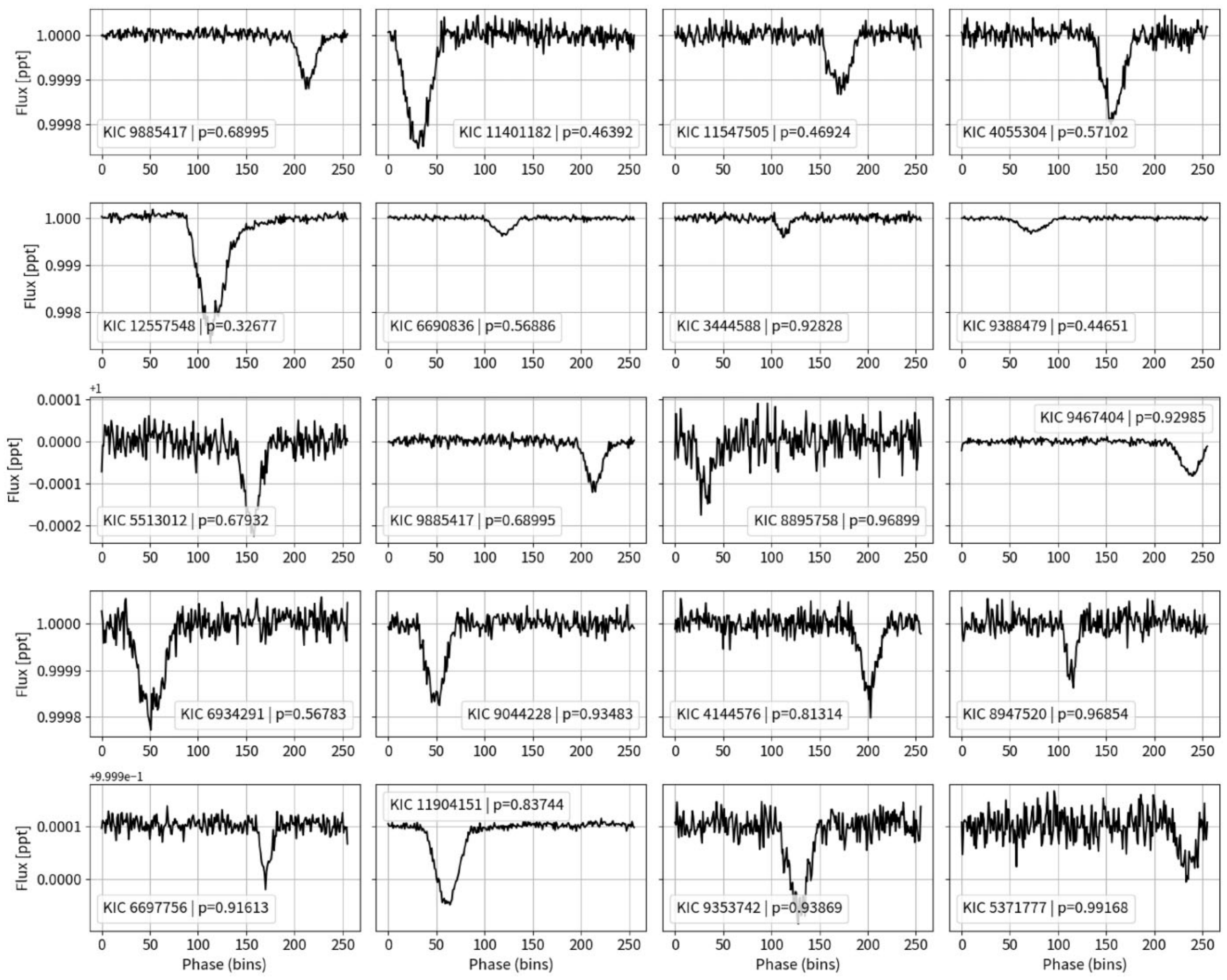


Figure A2. (Continued) GPFC recovery of confirmed USPs. The GPFC method identified all of the 43 confirmed USPs in *Kepler* with each of them given a CNN score above 0.99. All of the detected periods are the exactly same as the period recorded in the KIC catalogue. Each folded light curve exhibits a clear transit signal. This blind search test proved the validity of the GPFC method in terms of USP transit detection.

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