HIGH SCHOOL EDITION
Journal of Student Research
Volume 11 Issue 3 (2022)

A Novel Method of Transit Detection Using Parallel Processing and Machine Learning

Zarar Haider[1], Jian Ge[1], Kevin Willis[1], Yinan Zhao[1], Kevin Wang[1]

Trinity School, USA

ABSTRACT

Ultra-short-period (USP) planets are rare Earth-sized planets with the shortest possible orbital periods of all known planets. The study of this group is important for investigating planet formation and evolution processes. To date, only slightly over 100 USPs have been detected in Kepler photometry data of nearby FGKM dwarfs. However, traditional methods used in detecting planets and transit-like events are often biased, inefficient and time-consuming. For the first time, we introduce a GPU fast phase folding technique coupled with a Deep Convolutional Neural Network (DCNN) specifically used for searching for USP planets. The DCNN is trained on a set of 2,000,000 synthetic USP samples and performs exceedingly well in identifying both true and false positive transit signals, with a 99.5% validation accuracy over the given training set. With the computational power provided by GPU fast phase folding, our method, compared to the traditional Box Least Squares method, has shown to be ~1000 times faster in searching for transit signals in a photometric light curve with the same or better precision and recall rate. Furthermore, our method can also be applied to other interesting planet populations beyond USP planets. We used this method to search through all available KIDs in the Kepler database and were able to reproduce all existing USP planets with a 100% recovery rate. We also discuss how further adjustments can be made, making this system even more efficient and powerful, as well as how it can be applied to broader planet populations.

Introduction

The secular formation theory states that USP planets were initially formed on orbits > 1 day before interaction with other planets within their system launched them into eccentric, inclined orbits that eventually tidally shrunk to the current orbits of less than a day. Though this is the most popular formation theory, others predicate on similar observations of USP planets, specifically their low eccentricity, circular orbits and how tidal interactions influence their orbit and consequently period. This led to the “high-eccentricity migration” theory being proposed (Schlaufman et al. 2010). Further formation scenarios have materialized in order to explain the low eccentricity and observed mutual inclination of USP planets. Researchers Pu \& Lai (2019) pointed to the low-eccentricity tidal migration induced by secular planet-planet interactions in their findings that reproduced a USP population consistent with the multi-planet systems present in the Kepler mission.

The most commonly used method of exoplanet detection is transit photometry, with the majority of exoplanets discovered using it. Typically, teams of experts have to manually examine possible planet signals and vote on their final decisions. This process can be strenuous as it takes examiners up to a few days to eliminate the obvious false positives, on tens of thousands of candidates. Furthermore, human vetters may not always maintain a consistent set of criteria when judging potential planetary signals. Even an experienced team of vetters may sometimes disagree on the disposition of a possible planet signal, and dispositions given to the same object may vary depending on previous context relating to other TCEs viewed recently. Once this process

is concluded, to prepare the data for the standard Box Least Squares (BLS) method (Kovács et al. 2002), researchers then need to fit and normalize the time series to fill the gaps in the unevenly spaced time series in which there may be a lack of data. The amount of time and effort put into detecting candidates is extremely high, and the results aren't always completely accurate, resulting in possible exoplanets being missed out on. Therefore, the process in which planets are detected must be accelerated and be less dependent on direct human involvement in order to maximize the output and accuracy of exoplanet detection.

This is where astronomy and artificial intelligence work well together. Due to the nature of the lightcurve data and spectra, astronomy has always produced an abundance of data that will always take very long to completely vet and analyze. A machine learning algorithm that can automatically, quickly and accurately analyze hundreds of thousands of light curves is therefore imperative to the study of USP planets, short-period planets, and exoplanets as a whole, in order to further our understanding of the formation and composition. With the acceleration of astronomy and observation, more and more lightcurve data will be generated, and without a reliable, fast automatic program that can vet hundreds of thousands of images at once, the discovery of planets and the path to the exploration of the physical properties and formation theories of planets will be significantly deterred by the slow, relatively inaccurate traditional methods that are susceptible to error with an outpacing supply.

Recent advancements in computer science have led to the increasing usage of neural networks in scientific research, a phenomenon brought on by the scientific community's growing recognition of its powerful potential in studying big data. Designed to mimic the neuron structure of the human brain, neural networks are at the most basic level a multilayered construction of virtual nodes capable of learning complex pattern recognition and classification (LeCun et al. 2015).

Past responses to this problem of human vetting using deep learning have included work done by the Autovetter project (McCauliff et al. 2015) which used a random forest classifier in order to distinguish TCEs based on promising characteristics in line with Kepler pipeline statistics. Furthermore, Mislis et al. (2016). Thompson et al. (2015) and Armstrong et al. (2017) all utilized unsupervised machine learning on light curves with similar shapes, classifying them using labels from preexisting Kepler TCEs.

The most recent application of neural networks to exoplanet detection however has been by the Shallue \& Vanderburg Google Research team in their 2018 paper, and subsequent follow up research, wherein they trained a deep neural network up to 98.8% accuracy on existing TCEs provided by the Kepler database, and identified two new exoplanets as a result (Shallue, Vanderburg et al. 2018).

Our new unique method of transit detection begins with the fundamental lightcurve itself. We first normalize and filter the lightcurve according to a specialized spline fit that maintains an even amount of data on either side of the fit as well as sigma and end clipping. Given this preprocessed data, generally the next step is to fold and bin the data along trial periods, utilizing the typical manual BLS method in which a rectangular box fit tries various transit durations and depths in trying to locate the transit for an individual lightcurve. More broadly however, we recognize that the key principle in the transit method is to find the period, afterwhich the signal itself is strengthened as the noise is averaged out by the square root of n folds. In this approach then, we draw on this fundamental idea utilizing GPU Fast Folding to not only vastly accelerate the process, but amplify the transit signal as we repeatedly fold on the correct period thousands of times, significantly reducing noise and reinforcing the signal. This process is reproduced simultaneously on numerous parallel cores, enabling me to initiate this fast folding on thousands of lightcurves in minimal time. This is immensely helpful in generating training data for the neural network as the current population of USP planets is not sufficient for a Deep Convolutional Neural Network. Therefore, we create our own vast artificial dataset modeled after the existing USP planet population of varying parameters and train the neural network comprehensively, to the point that it can recover all existing USP planets with ease, in a fraction of the time normally needed. The DCNN itself eliminates the need for manual vetting and allows us to engage in a more accurate and efficient USP candidate search

process where each TCE is assigned a confidence score on whether or not it is a true signal based on the patterns observed in the vast training set.

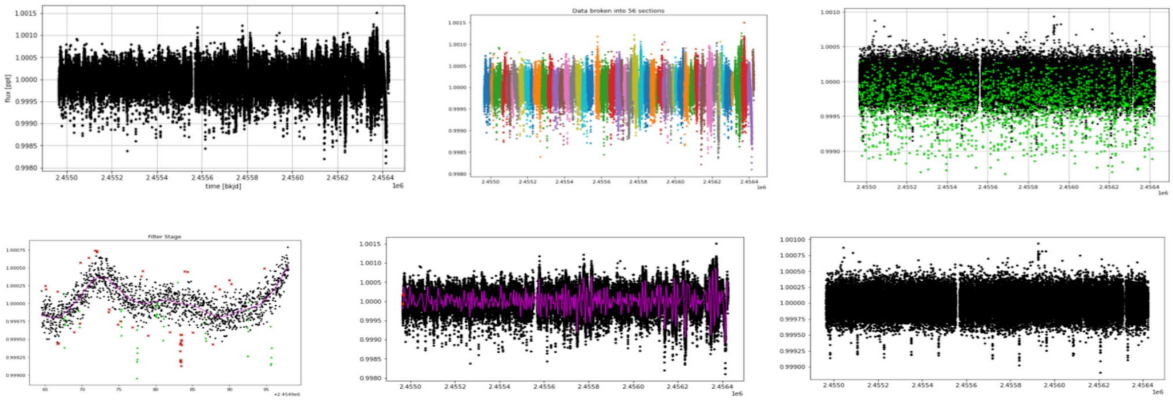

Figure 1. From left to right, our process for preparing the lightcurve, using KIC9002278 as an example. The input is the original lightcurve data. We then mask out all known transits in the planet system, fitting and filtering each sectioned off piece of the data for outliers, and apply a spline fit. This gives us a fully normalized lightcurve.

Though this method can be tested on any type of exoplanet, we chose to test on USP planets. In investigating the process of planet formation, the more comprehensive and fleshed out the existing pool of a planet population is, the more accurate predictions we can make. This is one reason why the study of USP planets, a rare classification of planets, is important and impactful. They enable us to further understand the formation and orbital evolution of USP planets, as well as secular and star-planet interactions and atmospheric erosion. Additionally, there are practical advantages to studying USP planets as they are easier to detect than planets of the same size in wider orbits, provide more useful information in the context of planet formation, and their masses and general planetary characteristics, the most basic inputs into the neural network, are easier to measure. They are sometimes hot enough to emit a detectable glow too, enabling observations to determine their surface temperature and reflectivity, which is usually impossible for wider-orbiting planets, endowing us with further information on an incomplete yet useful planet population.

In our paper, we present novel, more efficient, and more accurate methods of exoplanet detection using a deep convolutional neural network, which can also be applied to further planet populations beyond USP planets. In Section 2, we describe the streamlined routine of preprocessing, preparing data and the neural network itself that we have used in conjunction with machine learning and GPU parallel processing, which has dramatically sped up the processing and analyzing of hundreds of Kepler light curves. Then in Section 3, through the construction of an artificial data set of TCEs, we describe the training and testing of the neural network on our own data constructed from scratch. Finally in Section 4, we discuss the results stemming from the neural network search, the confirmation routine, stellar parameters from transit fitting and how the reduction of the total processing time from three to four days to only a couple of hours, enables vast advancements in exoplanet research and detection. We also discuss how further adjustments can be made, making this system even more efficient and powerful, and what further research can be conducted.

Preprocessing

Using data originating from the Kepler mission, each quarter represents approximately 90 days worth of data containing roughly 3,900 points evenly spaced in approximately 29.9-minute intervals, and with around 17 or fewer quarters of data available for each target. The resulting light curves have approximately 60,000 data points representing up to around four years of data. It is essential to further prepare and process the light curves to increase the likelihood of successfully recovering any USP planet transits. Even though they may occur often in this large sample of data, it is important that the neural network can recognize every variant type conceivable (Nikolaou et al 2020).

Given the input of the Kepler mission data, we must take measures to correct imperfections in the lightcurves themselves, which include both time and flux gaps, which means we choose not to use universal spline fitting. Rather we estimate both time and flux gap thresholds which allows me to properly identify and remedy missing data. We also utilize end clipping in order to clip off data points at the left and right of each section of data, removing data that confuses the spline fit, which is then implemented using a piecewise function. This enables me to create as accurate a fit as possible as the lightcurve is broken into individual sections of data and a fit is generated for each respective section.

First, we search for the KOI in the Kepler directory, a master list of possible stars with exoplanets, collecting data for all quarters of the light curve. At this point, we do not bin the lightcurve unless its size is over 50,000 data points. If the lightcurve does exceed this threshold however, we bin the data in order to expedite the process and use less memory since binning produces a smaller dataset that is still representative of the lightcurve. We then remove the NaNs from the data set and simply plot the raw data output, which includes every known planet in the system and serves as a preview of the filtered lightcurve.

Given this data, measures must be taken to minimize imperfections in the light curves. For example, light curves are split in locations where over 1.5 hours of time data is missing. By splitting the light curve at these time gaps, we avoid the complications of missing data when attempting to fit the data.

Our light curves are detrended using a smoothing spline fit. One of the key parameters that must be solved in this fitting process is the smoothing parameter. When it is set to its upper bound, the fit is a simple spline fit which overfits the data. Conversely, at its lower bound, the fit is a completely linear underfit. Finding a balance between the two is essential in producing an adequate fit to the data.

After this, we estimate the time gap threshold, which allows us to separate the data into distinct sections wherever there are time gaps in the data which are periods of time in which data was not taken. For example, if the average time between each measurement is 0.1 days, then any data missing for over 0.1 days is a gap. This process is required because the fitting routine often has troubles with the edges of data when attempting to create a smoothing fit. Therefore, if you don't split the data into their own respective sections, the smoothing spline filter will not properly fit between time spans which causes major issues in preprocessing.

Once data is properly split based on time gap threshold and end clipped, we then move to estimate a flux gap threshold. Although it is relatively easy to find time gaps within the data, oftentimes the gap is so small that the time gap threshold filter can't find it entirely, which is where the flux gap threshold functions. The split light curve is then filtered based on its standard deviation, then fitted with an optimally smoothed spline fit.

In the preprocessing routine the smoothing spline variable is essential. When it is set to its upper limit of 1, the fit is a simple spline fit which overfits the data, piercing every data point of the lightcurve. Conversely, at its lower limit of 0, the fit is a completely linear underfit. Finding a balance between the two is essential in good preprocessed data.

To find that balance, we check against two criteria. The first criteria entails calculating how many data points fall below and above the spline fit. The fit should have approximately equal amounts of data above and below the spline fit in every section. The second criteria uses a sigma value, calculated using the distance

between each data point, as the approximation for the noise of a section. After each iteration in the fitting process, we divide the light curve by the fit. We then take the RMS of that result and compare it to the approximate noise. If they are similar in magnitude, then we know that the preprocessing was comprehensive and that we have an accurate spline fit (Dattilo et al. 2019). After detrending is complete, we end clip, removing 30 data points at the left and right edges of each section of data. This eliminates data that the spline is unable to fit well. We then mask out all known transits in the system and concatenate all detrended sections into one fully normalized light curve.

GPU Folding and Artificial Data

Training a neural network requires a large sample of data containing the desired feature and having the network infer patterns for recognizing the feature. The more samples there are to train on, the more the network can learn and improve – training sets often contain thousands of samples, if not hundreds of thousands or more. However, for USP planets, their small existing population does not provide enough samples to make a training set of real data.

One solution is to simulate transits with a simple shape to closely approximate the properties of real USP transits. This would ensure a sufficiently large sample of signals that can be used to train the neural network. We utilize this method in making over 2,000,000 synthetic light curves for training, which is split into halves of transits and pure noise in order to ensure that the neural network is not biased to either data set.
We first load a catalog of Kepler Objects of Interest. This provides targets with exoplanets that were confirmed and their parameters of transit duration, transit depth, stellar mass, stellar radius and orbital period (Thompson et al. 2018).

We initially construct positive samples with a transit generated through a trapezoid shape. This transit is based on the ranges of transit duration, transit depth, and radius ratio of planet to star of previously discovered USP planets (Armstrong et al. 2017). We construct these transits through uniform distributions of these parameters to avoid biases in the model (Fig. 5). The transit shape itself is constructed by marking the four vertices of a trapezoid. The trapezoid is then injected into a range of times representing the period time window, at a random location to account for the fact that a transit could be present anywhere in the window. Gaussian noise is applied to the light curves using the user-specified SNR, simulating real noise that plagues real light curves. The ratio between the top and bottom of the trapezoidal transit is randomized within a representative range in order to guarantee a wide array of possible USP transit shapes.

In the false samples, the light curve contains only noise and no transit signal. An additional step is taken in order to ensure that the transit signal is always 2 standard deviations greater than the noise level of the data. As the neural network would eventually have to detect real transit signals, the artificial data generated accounts for the entire range of possible properties that a real USP transit could have.

One quick check for the quality of generated artificial data for the training set is to visually compare real and synthetic signals. As shown in Figures 3 and 4, the real and artificial signals are indistinguishable by eye, both showing very similar features. Comparing the distributions of transit durations and depths for synthetic and real transits side-by-side, the range of durations and depths in the artificial data fully covers the range of durations and depths found in the real data, so it appears to be a fairly representative and accurate simulation, ready to input into the neural network.

We also find that the new procedure of windowed fastfolding utilizing a GPU significantly reduces the time needed to prepare light curves for input into a neural network model. The most commonly used method for modeling and calculating the parameters of these transits is with the Box Least Squares (BLS) algorithm (Kovács et al. 2002). With BLS, an estimate for the period and transit reference time can be determined, but the method has its limitations.The major disadvantage of BLS is that it requires searching a huge combination of

periods and t0s which is extremely time consuming. In contrast, the GPU Folding method only requires a desired period range and period stepping parameter. Another major weakness of the BLS algorithm is that shallow or weaker transits often do not produce enough power in a power spectrum to appear as peaks and thus be detected as potential candidates. This eliminates a large population of possible USP planets that may be overlooked by BLS. (Hinners et al. 2018).

Our procedure can be deployed in consumer hardware while maintaining its ability to scale to bigger datasets. Initially, data is copied over to the GPU for a parallel processing of the light curves. Given a range of periods [p1, p2] to attempt, and a period stepping parameter, x, a given light curve will be folded n times. Consequently, noise is reduced by the square root of n folds. Folding is done at periods starting at p1 and incrementing by p2-p1x each time.

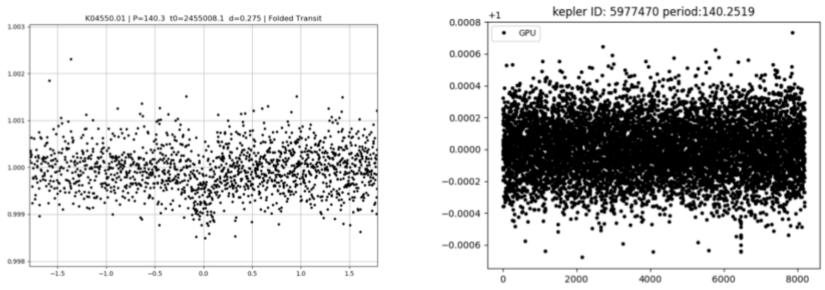

Figure 2. Fastfolding result (left) compared with a manual folding result (right)

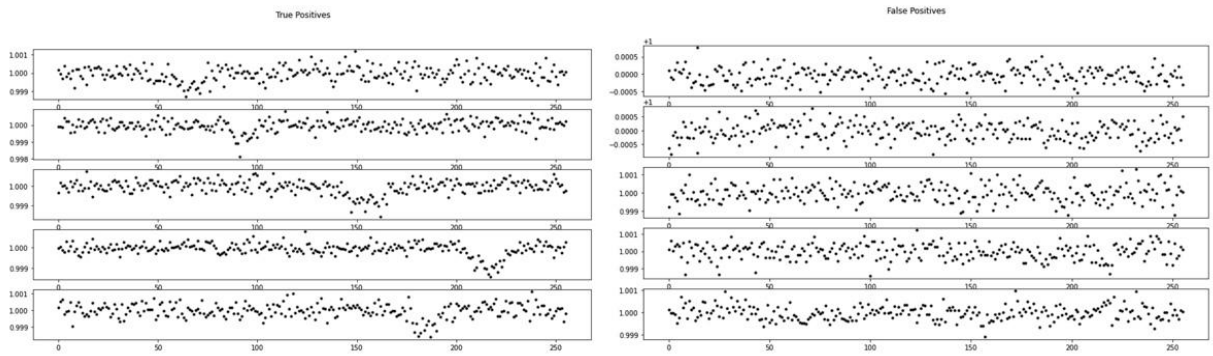

Figure 3. An artificially generated true lightcurve signal side by side with real true lightcurve signals.

The output for each period, p, is a window of fixed size n, where each point represents a binned time of width pn. In other words, GPU fastfolding is a method to brute force search a desired range of periods. A transit feature is apparent in the folded result, shown in the top plot. Compared to the manually folded result in the bottom plot, the folded light curve noise is reduced, despite the same binning being used in both procedures. This is because the transits overlapped at the same location multiple times during the folding process resulting in a higher SNR (Fig 2).

GPU fastfolding allows for a large number of folds to be quickly generated and scored by the neural network. This improves the chance of discovering transit candidates because we are able to search thousands more light curve folds per second than traditional methods.

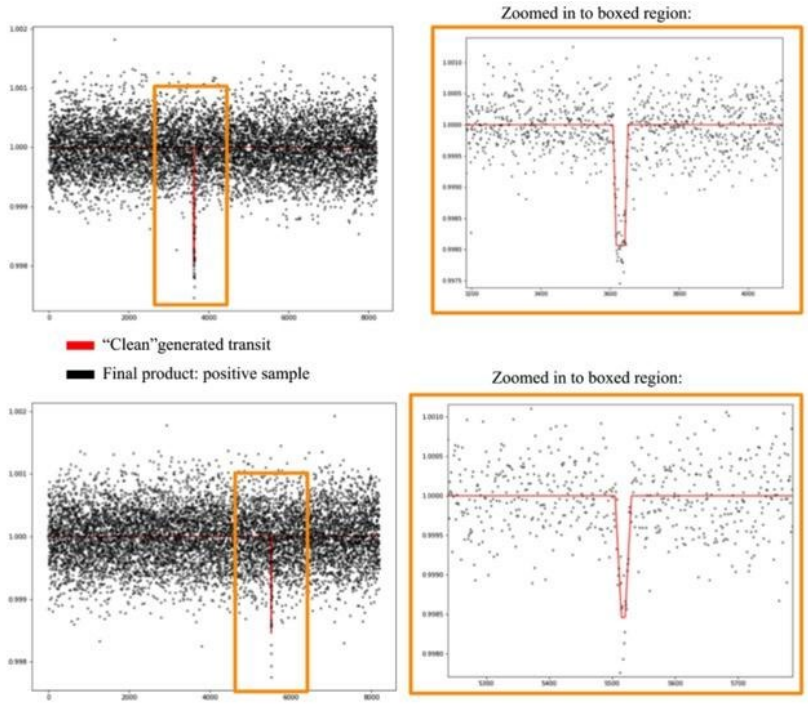

Figure 4. Artificial transit signals generated using the trapezoid method.

Deep Convolutional Neural Network

Structure

We utilize a DCNN, a specific type of flexible deep neural network uniquely suited for image and pattern recognition. In most neural networks, nodes are organized into "layers" of hundreds or thousands, with each individual node connecting to separate nodes in both the previous and next layers in order to receive and send data further down the chain of layers. We employ deep learning primarily as a type of representation learning, using multiple computational layers to build more and more detailed features, until eventually reaching a final feature layer that can differentiate between complex and varied objects (LeCun et al. 1998). The main distinguishing factor of DCNNs is the convolutional layer, which applies a specific matrix operation known as convolution in order to superimpose a smaller matrix known as the kernel onto the larger data matrix. Depending on the kernel, this convolution can transform the data in many ways, most notably by highlighting edges, points, gradients, or other spatial features to form a "feature map" useful for pattern recognition.

The initial neural network model consisted of eight convolutional layers, eight batch normalization layers, ten ReLU activation functions, four max-pooling layers, one flattening layer, three dense or fully connected layers, two dropout layers, and an output. Following standard neural network procedure, we divided the data we had compiled into two sets: one for training and one for validation. After we achieve sufficient training and validation accuracy, we test the neural network on a testing set to gauge its performance on data it has not

yet seen. A low validation score is a very good indication that the neural network is "overfitting," essentially just memorizing the specific characteristics of the training data rather than identifying general patterns which could be applicable in categorizing other data. Overfitting constitutes a major challenge in working with neural networks, because it renders the network useless when dealing with data which it has not been directly trained on, and so it was imperative to eliminate it in our project.

Individual connections between nodes of the neural network contain unique, learnable parameters called weights and biases, which determine the strength of the connection between the nodes and constantly adjust as the overall system improves in accuracy. Much like in the human brain, an individual node, or neuron, is incapable of performing complex tasks, but when many are connected in a dense, constantly evolving system, they gain the capacity to identify complicated patterns, judge their own performance, and effectively "learn" from their errors to improve in accuracy, in certain cases much more effectively than their human counterparts. The aforementioned layers form the computational power of the neural networks, however differing layers provide differing functions and the combination and interplay of these layers is essential in constructing an optimal DCNN.

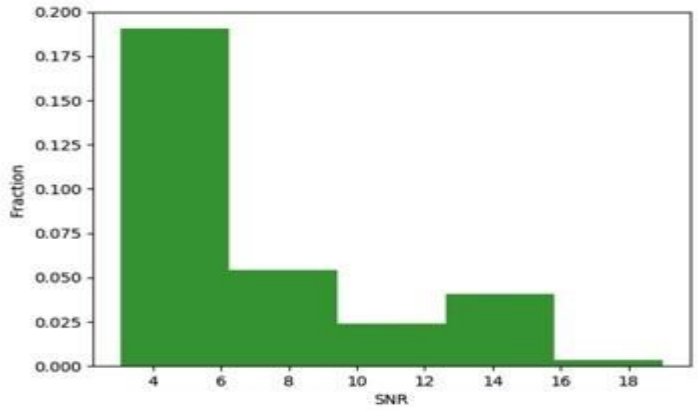

Figure 5. USP Planet Valley of SNR < 3.0. Y-axis indicates the percent of total USP planet discoveries, X-axis represents the SNR level of each USP planet. The number of USP planets drops to 0 once the SNR becomes lower than 3.

A hidden layer can contain multiple convolutional layers, each of which is followed by a batch normalization layer meant to further downscale data and regularize the output from the convolutional layer. The normalization layer is then immediately followed by an activation layer, a step critical to convolution because the activation introduces non-linearity to the network's output.

Pooling layers, typically found in conjunction with convolutional and activation layers, function primarily to incrementally reduce the dimensions of the large amount of raw data which is typically being input. Different types of pooling functions serve different secondary purposes on top of the primary function of dimensionality reduction. We use the most common pooling method, max-pooling, which is especially useful for extracting important features in data such as edges and shapes in an image.

The neural network also utilizes a flatten layer, which reduces multi-dimensional data down to one dimension in preparation for subsequent dense layers. More recently, data scientists have begun pairing dense layers with something known as a dropout layer, a type of regularization method which erases a randomly selected subset of the nodes in a certain layer as well as all of their connections. This recent innovation has the effect of adding noise to the connections, causing individual nodes to carry more weight, and is particularly effective when applied in conjunction with dense layers, since they contain the maximum possible number of connections. The

primary purpose of introducing noise is to combat the key issue of overfitting, a concept which will be discussed in further detail (Shallue & Vanderburg 2018).

Finally, the output layer of the deep convolutional neural network is mediated with a sigmoid function, an activation function particularly useful in predicting the probabilities of binary outputs rather than just giving the binary prediction. The S-shaped non linear curve it produces is probabilistic which allows the neural network to output a rational confidence level for each input rather than a binary "yes" or "no". This allows us to better understand which types of light curves the neural network had trouble on and which it was more confident in. This feature was useful in leading us to identify that transits with lower Signal to Noise Ratio (SNR) or low levels of noise relative to the transit signal, were predicted as positive with significantly less confidence than those with higher SNR.

Modifications

Our chosen neural network functions through two fundamental algorithms called feed-forward calculations and backward propagation. Feed-forward calculations describe the transformation of the input vector through a non-linear activation into the output vector. The backward propagation constantly modifies the values of the weights in each filter in a way that minimizes the cross-entropy cost function. The weights in each filter are updated back to front from the last convolutional layer to the first using the Adam optimization function. Adam has an adaptive learning rate that constantly updates itself, and it has been one of the most accurate optimization algorithms since it was developed. (Kingma et al. 2014) The weights are constantly updated until either the cost function is optimized or the error stops converging.

The DCNN architecture relies on input light curves that can be spatially analyzed and described by complex features. All hidden layers use the ReLU activation function, and the output layer uses the sigmoid activation function. The output of the model is essentially the predicted confidence score that the input is a transiting USP planet. The closer the value is to 1, the higher the model confidence is that the input is a real USP planet while values closer to 0 indicate high confidence that the input contains no transit. We also used the Adam optimization algorithm (Kingma & Ba 2015) to minimize the cross-entropy error function over the training set. We applied dropout layers to the fully connected architecture, which helped prevent overfitting by randomly "dropping" some of the output neurons from each layer during training to prevent the model from becoming overly reliant on any of the data features.

Fully-connected layers are responsible for the classification process as they can learn the non-linear combinations of the high-level features represented by the convolutional layers.
The neural network utilized a special activation function known as the Rectified Linear Unit (ReLU) (Nair & Hinton 2010). ReLU is the most commonly used activation function in deep learning models and serves special utility in DCNNs because it reduces nonessential data. The function is also computationally inexpensive, simply returning 0 for all negative inputs and the original input for all others. (Frustagli et al. 2001).

Each individual convolutional layer slides a small filter over each input, sums the result, and adds it to the feature map, building a more complex object. The DCNN uses many consecutive convolutional layers and in the deeper layers, simpler features learned in previous layers are combined. During training, the parameters of the convolutional filters are adjusted to minimize a cost function, a measure of how far the model's predictions are from the true labels in its training set. The convolution operation also has the secondary utility of downscaling the data. (Petrovich et al. 2018). Since the primary method of USP planet detection involves transit photometry, we implemented max-pooling layers to highlight transit-like spatial signatures for better detection. We tested SGD (Stochastic Gradient Descent), RMSProp (Root Mean Square Propagation) and Adam (Adaptive Moment estimation) optimizers with various hyperparameter configurations. After thorough testing we chose Adam as the optimizer in our model. For the learning rate we tested various values between 1*10^−3 and 1*10^−7. In the model itself we chose a very low rate of 1*10^−6.

One way in which we combated overfitting was by adjusting the Adam optimization function. Because all neural networks must have a mechanism in place to improve themselves, we implemented the Adam optimization function to update the learning parameters (weights and biases) backwards from the last layer to the first. Adam optimization is widely recognized as one of the most reliable algorithms for neural network optimization, due in large part to its special adaptivity. Whereas some other optimization functions have fixed learning rates, or "updating rates," Adam optimization is capable of adapting its own learning rate at the same time as it is adjusting the actual network's learning parameters in order to proportionally improve the network as it approaches higher accuracy values. During the first couple of iterations of the neural network, the training curve indicated that the neural network's accuracy on the training dataset converged much faster than its validation, a problem indicating overfitting. However, we were able to remedy this problem by adjusting the Adam learning rate, after which the training curves converged successfully (Fig. 6).

We also took steps to rearrange the neural network structure, decreasing the number of layers from 8 convolutional layers to only 6, increasing dropout from 0.2 to 0.3, and adding more max pooling layers. Having fewer layers, and thus fewer neurons, and dropping out more neurons decreased the potential for overfitting, and increasing pooling allowed us to capture more prominent features faster, both of which contributed to an overall increase in validation score. In addition to restructuring the neural network, we adjusted the SNR for the artificially generated transits so that the neural network could more clearly identify the transit features. Ultimately, we ended up with the most success in both training and testing for transits by increasing the minimum SNR so that the transits had a signal of 5σ or higher, the threshold at which the transit signals became strong enough to be clearly distinguished from the noise.

After all the described adjustments had been made, each convolution layer has 16 filters and a kernel size of 32, and is followed by a batch normalization layer and a ReLU activation function. Every other convolutional layer is followed by a max pooling function with a pool size of 8. The data is then flattened and finally fed through two sets of dense layers and corresponding dropout layers before arriving at the sigmoid output function, where it is converted into the confidence output. This final model was able to achieve a validation accuracy of 99.5%, which is sufficient to begin the search for USP planets.

Results

The DCNN is implemented in Python 3.6.6 using the open-source libraries TensorFlow and Keras. In this work, the DCNN was trained exclusively on the artificial light curves because the sample size of the real light curves were too small. The Adam optimizer was implemented in this model for computational efficiency and to simplify the number of parameters needed to be trained. (Kingma et al. 2014) The training curves in Figure 7 show that the training accuracies are always higher than the testing accuracies. Compared to the time that it took for the training accuracy to converge, the testing accuracy took longer to converge. This is likely to be caused by the overfitting of the DCNN.

The training phase involved an Adam optimization algorithm with a learning rate of 0.000001, ensured to reduce volatility in training and increase the stability of the model, which was trained for 100 epochs, with a batch size of 32, on a dataset composed of 2,000,000 synthetic samples: 1,000,000 positives (has transit signal) and 1,000,000 negatives (has no transit signal). 1000 synthetic lightcurves were also set aside for testing. The results of this network were excellent as the training and validation accuracy both began to approach 100 percent, with a final and maximum training accuracy of 99.7% and a maximum achieved validation accuracy of 99.5%. This is the model when validation loss is lowest and the optimal weights are saved. However, there is a high level of fluctuation in the validation accuracy: up to 10 percent at times. The most likely explanation for this result is that the learning rate was too high, causing the algorithm to prematurely "jump out" of some local

minima and resulting in missed solutions. To an extent, fluctuations in the validation accuracy could also originate from oscillations that occur when the convolutional layers are incapable of recognizing all characteristics of the transit signals and consequently will consistently and cyclically change a certain parameter in failed attempts to optimize the loss function. Some methods to mitigate this could include adding more convolutional layers, or increasing the size of each convolutional layer. Still, the neural network achieved sufficiently high training and validation accuracy scores, and as shown in the following section, was able to identify positive signals with a high rate of success.

After training the neural network model on synthetic data, the model was tested with 10,000 artificial samples to evaluate its performance. The overall results were promising: the model was able to correctly identify the samples as either having a transit signal or having no signal 97.18% of the time, at a high precision of 99.10% as well.

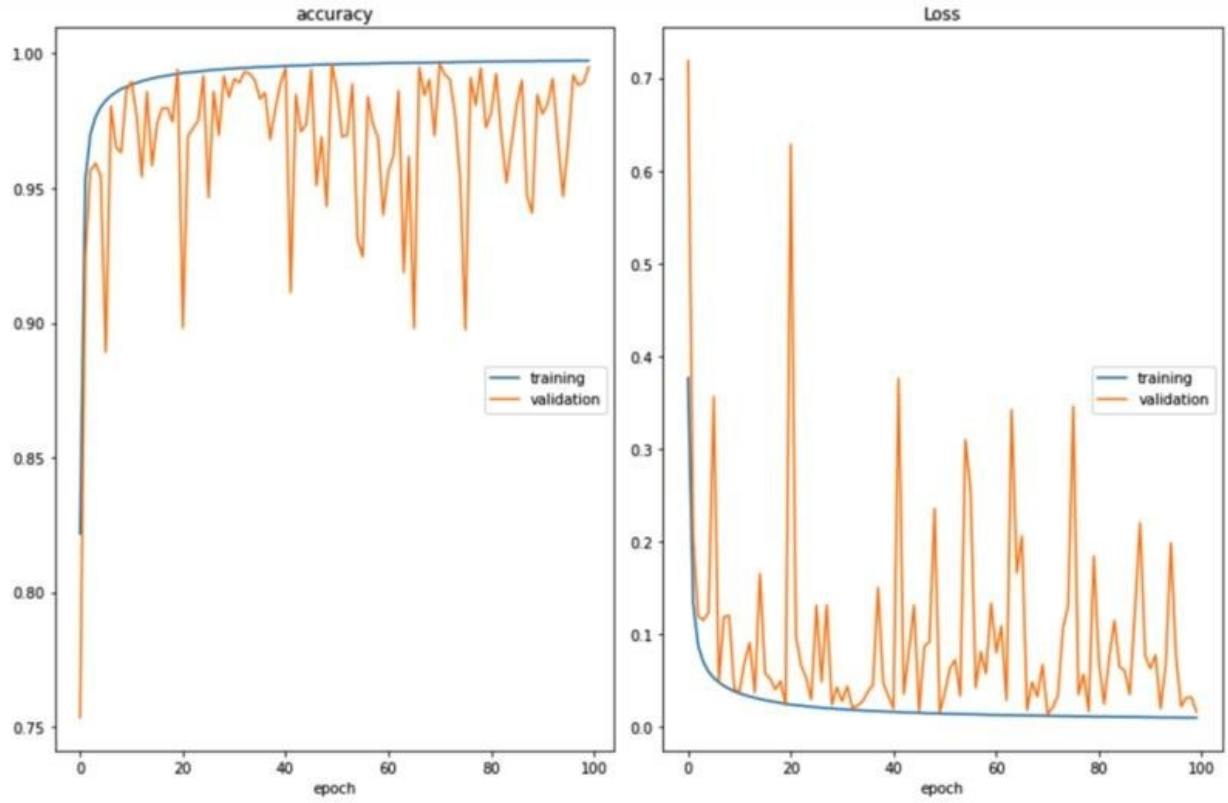

Figure 5. Final model of the neural network: accuracy and loss for both the training and validation sets, plotted against time. The training curves converge very quickly, with all fluctuations being relatively insignificant. The training set reached the final values of 99.7% accuracy and 0.9% loss, and the validation set reached 99.5% accuracy and 1.6% loss, both of which indicate that the neural network is performing exceedingly well.

Using the final model of the DCNN, we tested the preprocessed USP data and managed to rediscover all 106 existing confirmed USP planets. After the success of this benchmark test, we turned towards the Kepler database and searched through the complete archive of manually preprocessed Kepler light curves. We searched through all existing planetary systems or KIDs in the Kepler database, totaling over 2500 light curves.

We were able to comprehensively search through the roughly 2500 preprocessed lightcurves in the Kepler database and validate thousands of existing planets. Our model found several promising USP planet transits with true scores of over 0.8 and strong transit visibility but through our thorough vetting tests, we were able to falsify all of them. This vetting process is composed of various external false positive tests looking for

red flags in components such as stellar rotational activity and transit depth. Using these further tests, we searched for sine wave patterns indicative of stellar rotational activity, a phenomenon which can cause false positive transit-like signals. After subsequent tests, we were able to narrow down the list of possible candidates to the strongest ones which showed strong transit features and harmonic metrics. However, in order to further certify these discoveries, we used the BLS method, which fit a boxed-shaped transit according to a set range of periods, which encompassed the found period and a buffer on either side. The method outputted almost the exact same period as that which was GPU folded, further reinforcing the planet candidate. Further certification of the candidates was pursued through transit fitting with a Markov Chain Monte Carlo (MCMC) simulation (Goodman \& Weare, 2010). With the folded transit and additional stellar parameters as the input, the algorithm solves for additional transit parameters. We inputted the parameters of orbital period, t0, stellar mass, and stellar radius for the transit fit. Here the candidates failed, as their transit fits were not sufficiently strong and their SNR proved to be below the required value for most exoplanets. Further investigation testing for harmonic signals, revealed that these transits were harmonics of already existing planets.

For KID 7294743, we found a planetary radius of 1.885 and a radius ratio of 0.906. For KID 5446285 we found a planetary radius of 0.002 and a radius ratio of 0.002. Our neural network was able to predict with 99.998% and 86.42% certainty, respectively, that the two candidates contained the periods 0.99252 d and 0.57757 d, and further tests made by masking out the already known planets in the same star system confirmed that the transit signals made by planet-like bodies.

Though these results don't produce a candidate, they are promising in their rigor, as they represent an exhaustively thorough transit detection method capable of revealing the most promising planet candidates.

Conclusion

In our paper, we present a methodology for improving the detection of transit events in Kepler data using a deep convolutional neural network and parallel processing. Our unique method of transit detection occurs in three primary phases. First, we normalize and filter our lightcurve according to a specialized spline fit as well as sigma and end clipping. In the second phase we recognize that the key principle in the transit method is to find the period, afterwhich the noise is averaged out by the square root of n folds. We draw on this fundamental idea by utilizing GPU Fast Folding to accelerate the process of folding and reduce noise as we repeatedly fold on the correct period. Light curves are folded at trial periods using specialized GPU hardware and software which can produce binned folds of 100 thousand different orbital periods in roughly 2 seconds.

The third phase consists of training the CNN. There are currently not enough USP planets to train a CNN. Therefore, we create our own vast dataset of 2,000,000 artificial light curves, which is 10-100 times larger than other CNNs in exoplanet detection. This dataset is modeled after a uniform distribution of existing USP planet parameters and trains our neural network to the point that it can recover all existing USP planets with ease, in a fraction of the time normally needed for other CNNs. In the final phase, the CNN itself allows us to engage in a more accurate and efficient USP candidate search. After the model is trained on the artificial data it is then tested on real data, assigning each TCE a confidence score on whether or not it is a true signal based on the patterns observed in our vast training set, reducing time spent on manual vetting. Manual vetting of the CNN output is still needed though.

This method was tested on USP planets specifically and builds on the existing body of research surrounding USP planets. It demonstrates the potential of neural networks as agents in photometric exoplanet detection. The overall method of transit detection provides a promising new framework, through which future exoplanet research can be conducted, and the scope of the network itself can be extended to search for other types of exoplanets such as habitable small planets, Trojan planets, or Super-Earths. From the results in Figure 7, we have confirmed that compared to BLS, we maintain high accuracy through our GPU folding method. The

training set reached the final values of 99.7% accuracy and 0.9% loss, and the validation set reached 99.5% accuracy and 1.6% loss, both of which indicate that the neural network is performing exceedingly well. Our model has an accuracy above 99% at SNRs of 7 and above. We have also shown that the 2D-DCNN-folding model can have accuracy over 90% even when the folding period differs from the transit period by up to 0.2%.

While these results are encouraging, a neural network approach is still not yet strong enough to be broadly used in the field of exoplanet detection. Even if a machine learning model works well in training and testing, it has shown to be prone to make mistakes on unseen data. As mentioned before, these methods should be used with human supervision. Nonetheless, our current model can provide a uniquely efficient approach to rule out a large number of false positives and drastically reduce the number of cases requiring comprehensive manual reviews.

In our future work, taking this reality into account, we could take steps to expand our training set. A larger training set would allow for more complex machine learning models that can predict the light curves more accurately. However, this method has the disadvantage that the planet signals will need to be removed ahead of time. Some undetected planet signals will always remain in the data, though, which means we must test if undetected signals average out across the training set. In addition, the different rotation rates, spectral types, and inclination angles may be challenging to solve and require significantly more processing time.

Additionally, our current method for folding light curves also occasionally creates false positive transits in stars with stellar variability. In the future, improving our folding process could cut down on the number of signals due to stellar variability that are classified as likely planets.

Furthermore, our research could be extended to other photometric surveys such as TESS. These surveys provide an abundance of light curve data which could potentially yield greater discoveries when analyzed by our model. All new discoveries are valuable contributions to the scarce body of only 120 confirmed USP planets and roughly 80 more possible candidates in Kepler data. We conclude that a deep-learning model which combines a 2D-DCNN and GPU folding is a promising tool for the future of transit detection.

Acknowledgments

This research has made use of the exoplanet package and its dependencies PyMC3 for the inference engine and modeling framework, AstroPy for units and constants, Kipping 2013 for the reparameterization of the limb darkening parameters for a quadratic law, and Luger et al. (2018) for the light curve calculation. This research made use of Lightkurve, a Python package for Kepler and TESS data analysis (Lightkurve Collaboration, 2018), as well as its dependencies AstroPy and AstroQuery. This research has made use of the NASA Exoplanet Archive, which is operated by the California Institute of Technology, under contract with the National Aeronautics and Space Administration under the Exoplanet Exploration Program. This research has made use of the SIMBAD database, operated at CDS, Strasbourg, France. 2000,A&AS,143,9, "The SIMBAD astronomical database", Wenger et al.

References

Adams, E.R., Jackson, B. and Endl, M. (2016). ULTRA-SHORT-PERIOD PLANETS INK2SUPERPIG RESULTS FOR CAMPAIGNS 0–5. The Astronomical Journal, 152(2), p.47.

Adams, E.R., Jackson, B., Johnson, S., Ciardi, D.R., Cochran, W.D., Endl, M., Everett, M.E., Furlan, E., Howell, S.B., Jayanthi, P., MacQueen, P.J., Matson, R.A., Partyka-Worley, C., Schlieder, J., Scott, N.J., Stanton, S.M. and Ziegler, C. (2021). Ultra-short-period Planets in K2. III. Neighbors are Common with 13

New Multiplanet Systems and 10 Newly Validated Planets in Campaigns 0–8 and 10. The Planetary Science Journal, 2(4), p.152.

Dattilo, A., Vanderburg, A., Shallue, C.J., Mayo, A.W., Berlind, P., Bieryla, A., Calkins, M.L., Esquerdo, G.A., Everett, M.E., Howell, S.B., Latham, D.W., Scott, N.J. and Yu, L. (2019). Identifying Exoplanets with Deep Learning. II. Two New Super-Earths Uncovered by a Neural Network in K2 Data. The Astronomical Journal, 157(5), p.169.

Frustagli, G., Poretti, E., Milbourne, T., Malavolta, L., Mortier, A., Singh, V., Bonomo, A.S., Buchhave, L.A., Zeng, L., Vanderburg, A., Udry, S., Andreuzzi, G., Collier-Cameron, A., Cosentino, R., Damasso, M., Ghedina, A., Harutyunyan, A., Haywood, R.D., Latham, D.W. and López-Morales, M. (2020). An ultra-short period rocky super-Earth orbiting the G2-star HD 80653. Astronomy & Astrophysics, 633, p.A133.

Hamer, J.H. and Schlaufman, K.C. (2020). Ultra-short-period Planets Are Stable against Tidal Inspiral. The Astronomical Journal, 160(3), p.138.

Kovács, G., Zucker, S. and Mazeh, T. (2002). A box-fitting algorithm in the search for periodic transits. Astronomy & Astrophysics, 391(1), pp.369–377.

Millholland, S.C. and Spalding, C. (2020). Formation of Ultra-short-period Planets by Obliquity-driven Tidal Runaway. The Astrophysical Journal, 905(1), p.71.

Petrovich, C., Deibert, E. and Wu, Y. (2019). Ultra-short-period Planets from Secular Chaos. The Astronomical Journal, 157(5), p.180.

Sanchis-Ojeda, R., Rappaport, S., Winn, J.N., Kotson, M.C., Levine, A. and Mellah, I.E. (2014). A STUDY OF THE SHORTEST-PERIOD PLANETS FOUND WITHKEPLER. The Astrophysical Journal, 787(1), p.47.

Shallue, C.J. and Vanderburg, A. (2018). Identifying Exoplanets with Deep Learning: A Five-planet Resonant Chain around Kepler-80 and an Eighth Planet around Kepler-90. The Astronomical Journal, 155(2), p.94.

Wang, S. and Ji, J. (2015). Formation of the Planetary Candidates Observed by Kepler Mission. Proceedings of the International Astronomical Union, 11(A29A), pp.27–29.

Winn, J.N., Sanchis-Ojeda, R. and Rappaport, S. (2018). Kepler-78 and the Ultra-Short-Period planets. New Astronomy Reviews, 83, pp.37–48.

Yu, L., Vanderburg, A., Huang, C., Shallue, C.J., Crossfield, I.J.M., Gaudi, B.S., Daylan, T., Dattilo, A., Armstrong, D.J., Ricker, G.R., Vanderspek, R.K., Latham, D.W., Seager, S., Dittmann, J., Doty, J.P., Glidden, A. and Quinn, S.N. (2019). Identifying Exoplanets with Deep Learning. III. Automated Triage and Vetting of TESS Candidates. The Astronomical Journal, 158(1), p.25.

Zucker, S. and Giryes, R. (2018). Shallow Transits—Deep Learning. I. Feasibility Study of Deep Learning to Detect Periodic Transits of Exoplanets. The Astronomical Journal, 155(4), p.147.