# Search for Exoplanets

Petra Brčić
Applied Mathematics
petrabrcic94@gmail.com

Sandro Lovnički
Computer Science
lovnicki.sandro@gmail.com

June 30, 2018

**Abstract:** With the launch of Kepler space observatory in March 2009, the search for exoplanets has extensively began. There are currently $3,758$ confirmed exoplanets and contrary to the early approaches of validating exoplanet candidates by hand, machine learning has begun taking its toll on the subject. We would like to investigate neural network model for detecting transiting exoplanets from Kepler observatory light-curve data collected over a period of 4 years, with over $200,000$ documented stars in our galaxy, the Milky Way. On April 18, 2018, a new satellite (TESS) has been launched, specificaly designed to search for exoplanets using the transit method. After it's 2-year planned mission, complete data should be available to public and more than $20,000$ new exoplanets are expected to be found.

**Keywords:** Kepler, light-curve, exoplanet, machine learning, neural network, TESS.

## Contents

## I. Introduction

In March 2009, NASA launched Kepler space observatory to discover Earth-size planets orbiting other stars in our galaxy. With most of the discoveries made after Kepler data was anounced, there are currently 3,758 confirmed exoplanets in 2,808 systems, with 627 systems having more than one planet. Contrary to the early approaches of validating exoplanet candidates by hand or with humanly constructed models, recent years have been fruitfull for usage of machine learning for classifing Kepler's data.

## II. Methods

There are many methods for detecting an exoplanet existance, as it can be seen in [3].
For example, *gravitational lensing* method exploits the stars' gravitational fields' effect on magnification of distant background stars' light. If the star in question has a planet, its gravitational field makes a detectable contribution to the lensing effect. Over the past 10 years, just a 1000 such events have been observed because specific alignment of bodies is required. Nonetheless, 19 exoplanets have been discovered using this method, whose full list is given in [4].
It is notable to mention *radial velocity* method which uses variations in the speed with which specific star moves away or towards the Earth. Those variations are influenced by orbiting planet's gravitational pull and therefore we can also deduce the planet's mass. Up to 2012, this method was most effective for exoplanet detection whose full list is given in [5].

The most effective method today is the *transit photometry*. When a planet crosses (transits) in front of its star, the star's brightness diminishes depending on planet relative size to it. To detect those brightness variations, observer's relative position must be properly aligned with star-planet axis which is not something one can control. The probability of a random alignment producing a transit in a system with Sun-sized star and a planet at 1AU [1] from it is 0.47%. Ilustration of how those brightness variations are manifested in Kepler's data can be seen in Figure 1.

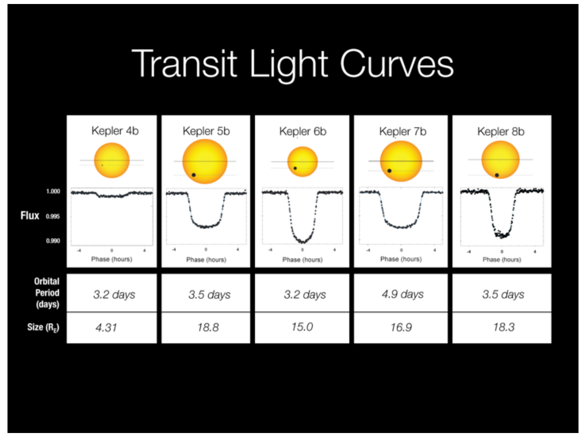


**Figure 1:** *Illustration from Bill Borucki's Jan 2010 AAS Presentation*

Another disadvantage of this method is a high rate of false positives due to measurement precission and external noise that can be interpreted as a transit. For this reason, transit photometry is often combined with radial velocity to produce better results and eliminate false positives. The main advantage of transit photometry is planet's size deduction from light curve drops, which combined with transit photometry's mass deduction yields planet's density - a property much appriciated in search of habitable worlds. Also, by observing spectrum of light passed through the planet's atmosphere, one can detect which chemical elements are present. By April 12, 2018, Kepler detected 2,343 exoplanets using transit photometry as a base method.

## III. Data

Data used for this project is obtained by combining NASA Exoplanet Archive data for Threshold-Crossing Events which (currently) consists of 34032 documented TCEs, each having $-CHECKNUMBER-$ column features

[1] 1 Astronomical unit ≈ Earth's ditstance to the Sun ≈ 150 million kilometres

and raw Kepler light-curves (star's flux (brightness) over time) at Mikulski Archive for Space Telescopes consisting of over $2000000 - CHECKNUMBER-$ star lightcurves monitored by Kepler.

Most important TCE features for our projects are `kepid`, `tce_period`, `tce_time0bk`, `tce_duration` and `av_training_set` which (respectively) correspond to the Kepler identifier of host star, period of the event, time of the first appearance (more precisely; time of the begining of drop + `tce_duration`/2), duration of the event (drop in star's brightness) and label from set $\{PC, NTP, AFP, UNK\}$ meaning planet candidate, non-transit phenomena, astrophisical false positive and unknown. Those features will enable as to download just the Kepler light-curves of interest and later on to process each light-curve to extract and amplify features of TCE.

## i. Creating Dataset

At the time of collecting our data, an older version of a TCE table was available with around $20000 - -GETCORRECT-$ labeled TCEs. Distribution of `av_training_set` labels is as follows:

- -NUMBER- TCEs with label PC (planet candidate)
- -NUMBER- TCEs with label NTP (non-transit phenomena)
- -NUMBER- TCEs with label AFP (astrophisical false positive)
- -NUMBER- TCEs with label UNK (unknown)

We used that table, specifically `kepid` colum to download just the light-curves for those stars from Kepler data. These light-curves are taking about 90GB of space and are just a fraction of all the Kepler data.

Still, our machines are not fully prepared to tackle the preprocessing and learning from such a vast amount of not at all simple data, so we decided to choose randomly just some of the TCEs from table. We ended up selecting 1058 TCEs labeled as planet candidate, 471 TCEs labeled as astrophisical false positive and 734 TCEs labeled as non transit phenomena.

As an example, here is a small table of selected features of one TCE documented for a star with `kepid` 9517393:

| | |
|---|---|
| kepid | 9517393 |
| tce_period | 219.322 |
| tce_time0bk | 320.728 |
| tce_duration | 0.5125000000000001 |
| av_training_set | PC |
| ... | ... |
| tce_plnt_num | 1 |
| tce_eqt | 313.0 |
| ... | ... |

## ii. Preprocessing Lightcurves

In order to feed our model, we first need to process the raw light-curve into more useful and compact form. Raw light-curve is loaded as an array of smaller arrays called quarters which represent distinct periods in Kelper telescope's configuration, position, etc. We thus might expect quarters to be on a slightly different scale. In figure 2 we show lightcurve for `kepid` 9517393 right after loading `time` and `flux` arrays.

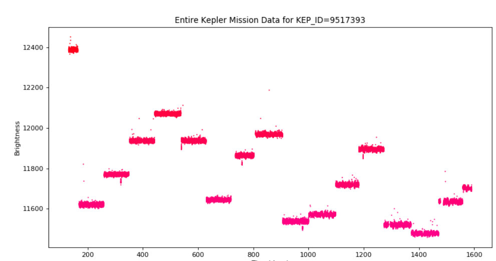


**Figure 2:** *Raw light-curve data*

It is easy to see that almost every star has an unpredictable variation in its flux, so the data is pretty distorted and irregular which is something that is to be avoided because it would interfere with our model's conclusions and feature extraction. Thus, we calculate a B-spline over the entire light-curve data, ignoring points that cannot be fitted or are of type `None`. This spline can be seen in figure!3 colored black.

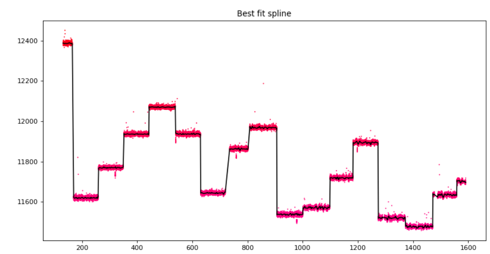


**Figure 3:** *Raw light-curve data with fitted B-spline*

Next, we divide light-curve by the spline to obtain 4. Notice how specific TCEs can now be seen without even enlarging.

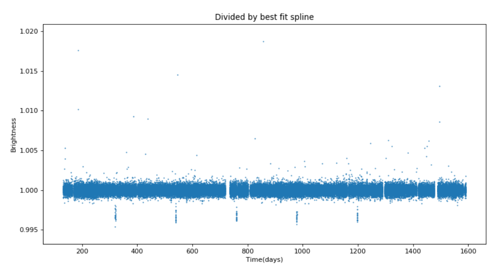


**Figure 4:** *Raw light-curve data divided by spline*

Now we need to exactly find the position of desired TCE which we know from TCE table, column `tce_time0bk`. Focusing just on that segment of this plot, we get figure 5

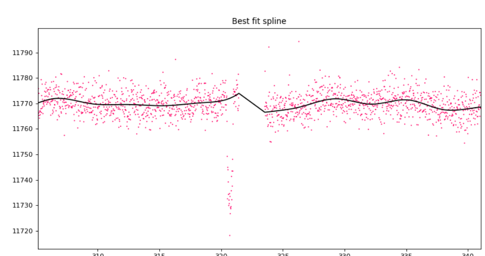


**Figure 5:** *Enlarged area of the TCE*

We see that the amount of useful data of this "drop" is very poor, but we must remember that those events are periodic by definition so we hope that there is more instances of that same "planet" crossing during the lifetime of Kepler. We find the `tce_period` from TCE table and fold every drop separated by `tce_period` to `time0bk`. Now we divide TCE's duration into 201 discrete points and calculate median value for flux in each of 200 intervals in between. This results in a much richer, and uniform for all events, representation of a TCE which can be seen in!6. We call that view a "local view" and it is going to be the input of our model.

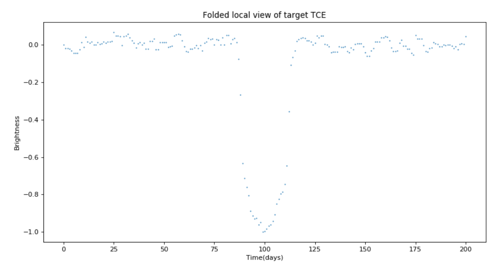


**Figure 6:** *Folded local view of the TCE*

## IV. Convolutional Neural Network

With the recent rise of neural networks, many variations are being explored for specific problems. In classification, convolutional neural networks have been proven to yield best results and we will be using one in this project. Let's first briefly introduce the reader with the concept of convolutional neural networks and how exactly they differ from regular neural networks.

A convolutional neural network (CNN) can be viewed as a deep neural network with addition of feature extraction mechanism in the from, that is — before the classification with fully connected layers begin. Building block of convolutional neural network are convolutional layers with pooling and fully connected layers for classification. General scheme of one CNN with two layers of 2D convolutions can be seen in figure 7.

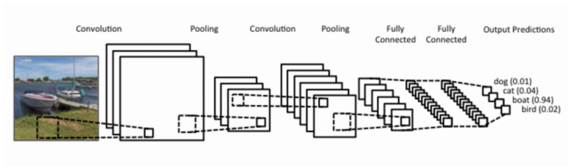


**Figure 7:** *Scheme of a 2D CNN*

In previous figure, we have a CNN that is used for classifying 2D data (generally, images are third order tensors, but we can assume image to be grayscale). Our data is not two-dimensional so we will be using 1D convolution with ReLu function as activation and maximum pooling, each of which will be explained in the following subsections, specifically as they were in out project.

## i. Convolutional Layer

A convolutional layer consists of neurons that are not fully connected, that is — each neurons in this layer is connect to just the small portion of the input. The convolution layerâĂŹs parameters consist of a set of learnable filters. During the forward pass, we slide (more precisely, convolve) each filter across the input and compute dot products between the entries of the filter and the input at any position. As we slide the filter over the the input, we will produce a 1-dimensional activation map that gives the responses of that filter at every position. Furthermore, we can stack multiple convolutional layers before pooling their output.

We have a set of 16 filters in first convolution block and each of them will produce a separate 1-dimensional activation map. We stack these activation maps along the depth dimension and produce the output. So, for out input representation — arrays of length 201 (shape $(?, 201)$, where question mark is batch size during training and 1 during prediction), the output of convolution layer will be of shape $(?, 201, 16)$. We then feed this data into another convolutional layer with a set of 16 filters yielding as outputs tensors of shape $(?, 201, 16)$.

Also, there are 32 filter in the second convolutional block taking as input the output of pooling the outputs of the first convolutional block. The output of this second convolutional layer is the set of features our convolutional blocks have learned and it is the reduced with maximum pooling, flattened to shape $(?, 1472)$and passed to fully connected layers for classification.

## ii. Rectified Linear Unit

For our activation function, we used rectified linear unit (ReLu). It is applied to the dot product of filter and current area the neuron is "looking" at, to produce an output value. One other valid choice for this task would be the sigmoid, but it would not stress enough the importance of larger values.

To be specific, when our network is trained for 1000 steps with sigmoid activation function, it only manages to get accuracy of 0.5265 while with ReLu it goes to 0.9336 in the same number of steps. Furthermore, it has 0 true positives in the confusion matrix and we present why exactly that is so in the next figure.

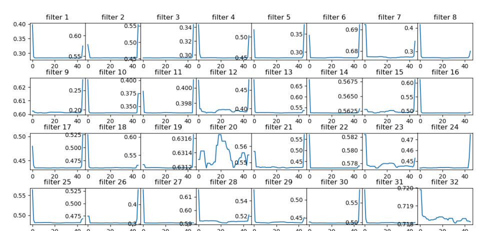

**Figure 8:** *Output of 32 filters after last pooling before FC (using sigmoid activations)*

As we can see in figure 8, almost no features are detected and therefore almost every example will be classified as non-transiting phenomena. Notice how vertical scale is pretty much the same and this is because of pretty much the same outputs from sigmoid. We also present the same picture for ReLu.

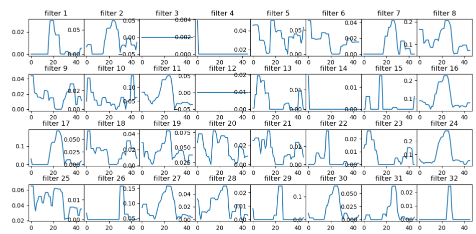

**Figure 9:** *Output of 32 filters after last pooling before FC (using ReLu activations)*

### iii. Max Pooling

Pooling is a method of downsizing the input data, but keeping the structure. It is used to abstract features obtained by convolutional layer into more complex features that then go further into neural network. We used max pooling with size 7 and stride 2 which lead to reducing the output size of the first convolutional block from 201 to 98 and output size from the second convolutional layer from 98 to 46. Max pooling is done almost like convolution, but we do not calculate the dot-product of "pool filter" of length 7 with the part of the input of length 7. Instead, we just take maximum value of 7 input values we are looking at, slide 2 steps, collect a maximum value again and so on.

### iv. Fully Connected

After our second convolutional block, we ended up with 32 values of length 46 which are now flattened to form a 1472 inputs for network of 2 fully connected layers with 1024 neurons each. We also use ReLu as activation for those neurons.

### v. Sigmoid Output

Lastly, the computations of fully connected layers are sumarized in a single output value representing the probability that a TCE is a planet candidate. Therefore, we use a sigmoid function as activation for this last neuron.

## V. Training

We train a convolutional neural network model on a randomly constructed set of data consisting of 1058 planet candidates, 471 astrophysical false positives and 734 non-transiting phenomena.

— TENSORFLOW IMAGES —

— CONFUSION MATRICES —

### i. Parameters

### ii. Progress

## VI. Results

In this last section, we present some of the most interesting results we encountered during our journey of training and testing our exoplanet classifier/finder.

We are proud to say that our model outperformed the initial creator's of astronet model at classifying the planet they discovered. While their model was "just" 0.9480018 confident that it is a planet, our predicts it with certainty of 0, 96??. However, we are aware that this is most probably due to our limited training set, but we cannot be sure until we, hopefully soon, get our models going on a more powerful machine.

### i. The weirdest

### ii. The cleanest

## References


[1] `https://en.wikipedia.org/wiki/Kepler_(spacecraft)`

[2] `https://en.wikipedia.org/wiki/Transiting_Exoplanet_Survey_Satellite`

[TESS] `https://tess.mit.edu/`

[3] `https://en.wikipedia.org/wiki/Methods_of_detecting_exoplanets`

[4] `https://en.wikipedia.org/wiki/List_of_exoplanets_detected_by_microlensing`

[5] https://en.wikipedia.org/wiki/List_of_exoplanets_detected_by_radial_velocity

[6] https://en.wikipedia.org/wiki/List_of_exoplanets_discovered_using_the_Kepler_spacecraft

[Shallue and Vanderburg, 2017] Christopher J. Shallue, Andrew Vanderburg (2017). Identifying Exoplanets with Deep Learning: A Five Planet Resonant Chain around Kepler-80 and an Eighth Planet around Kepler-90 arXiv:1712.05044 [astro-ph.EP]

2

[Mikulski Archive for Space Telescopes] https://archive.stsci.edu/

[NASA Exoplanet Archive] https://exoplanetarchive.ipac.caltech.edu/cgi-bin/TblView/nph-tblView?app=ExoTbls&config=q1_q17_dr24_tce

[TCEs table] https://exoplanetarchive.ipac.caltech.edu/cgi-bin/TblView/nph-tblView?app=ExoTbls&config=q1_q17_dr24_tce