IDENTIFY EXOPLANET CANDIDATES WITH DEEP LEARNING MODELS

A THESIS

SUBMITTED TO THE DEPARTMENT OF PHYSICS

OF THE UNIVERSITY OF HONG KONG

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

By

Wenchao Wang

August 2021

Abstract of thesis entitled

IDENTIFY EXOPLANET CANDIDATES WITH DEEP LEARNING MODELS

Submitted by

Wenchao Wang

for the degree of Doctor of Philosophy

at the University of Hong Kong

in August 2021

The work is about identifying planet candidates using deep learning models. Finding these objects manually is a very labor intensive task. For example, *The Large Synoptic Survey Telescope (LSST)* is expected to generate about $200,000$ images per year, which is equivalent of more than 10^6 GB of data. Therefore using reliable algorithms to manage the data is necessary. Deep learning can be helpful because it suits well for very large input data. In general, having more data only makes deep learning models perform better.

We use *Kepler Space Telescope* and *Transiting Exoplanet Survey Satellite* (*TESS*) to detect planet candidates by using a convolutional neural network model. We apply the Q1-Q17 (DR24) table as our training and test sets. The model takes two phase-folded light curves and some parameters of each transit-like signal and then outputs whether the signal represents a planet candidate (PC), a non-transiting phenomena (NTP) or a false positive (FP). In the current model, we feed 17 features into a dense

neural network model, such as transit durations and depth of signals. At this stage, the models achieve AUROC and accuracy of about 97.7%, 95.9% respectively for the test set. The accuracy for the training set can be over 99%, which means that the model can easily overfit the data. The most straightforward way to the problem is to use more data to train the model. Therefore, we plan to train it with more simulated data later in order to increase the AUROC and accuracy of predictions.

Identify Exoplanet Candidates with Deep Learning Models

Department of Physics, The University of Hong Kong, Pokfulam Road, Hong Kong

Wang Wenchao

3030053350

Declaration

I hereby declare that this whole dissertation report is my own work, except the parts with due acknowledgment, and that it has not been previously included in a thesis, dissertation or report submitted to this University or to any other institution for a degree, diploma or other qualifications.

Signature: _____

Name: Wenchao Wang

Date: August 2018

Acknowledgments

Lots of people give me much precious help. First of all, I really appreciate my supervisor Dr. Stephen Chi Yung Ng for his great support, patience and kindness. Prof. Takata also provides me much help in term of pulsars' emission mechanisms and numerical simulations and is really charming. In addition, I thank Ms. Ruby Cho Wing Ng for her invaluable guidance and advice. They give me not only knowledge and techniques but also encouragements. Therefore, I would like to express my most sincere thanks to them.

Abstract

The work is about identifying planet candidates using deep learning models. Finding these objects manually is a very labor intensive task. For example, *The Large Synoptic Survey Telescope (LSST)* is expected to generate about $200,000$ images per year, which is equivalent of more than 10^6 GB of data. Therefore using reliable algorithms to manage the data is necessary. Deep learning can be helpful because it suits well for very large input data. In general, having more data only makes deep learning models perform better.

We use *Kepler Space Telescope* and *Transiting Exoplanet Survey Satellite* (*TESS*) to detect planet candidates by using a convolutional neural network model. We apply the Q1-Q17 (DR24) table as our training and test sets. The model takes two phase-folded light curves and some parameters of each transit-like signal and then outputs whether the signal represents a planet candidate (PC), a non-transiting phenomena (NTP) or a false positive (FP). In the current model, we feed 17 features into a dense neural network model, such as transit durations and depth of signals. At this stage, the models achieve AUROC and accuracy of about 97.7%, 95.9% respectively for the test set. The accuracy for the training set can be over 99%, which means that the model can easily overfit the data. The most straightforward way to the problem is to use more data to train the model. Therefore, we plan to train it with more simulated data later in order to increase the AUROC and accuracy of predictions.

Contents

Declaration i

Acknowledgments ii

Abstract iii

List of Figures vi

List of Tables 1

1 Introduction 2

1.1 Transit Photometry 2

1.2 *Kepler Space Telescope* 3

1.3 Machine Learning 4

2 Data Preparation 5

2.1 Download Light Curves 6

2.2 Light Curves Preprocessing 8

2.2.1 Remove Multiple Signals in the Same Stellar System 9

2.2.2 Remove Outliers and Long-term Trend 10

2.2.3 Fold and Bin Light Curves 10

3 Deep Learning Models **15**

List of Figures

1.1 The distribution of TCE depths of *Kepler* DR24 data. 3

2.1 The screen shot of webpage containing light curves of kepid 000757137 7

2.2 Plot of normalized flux to time of kepid 11442793. 8

2.3 Plot of normalized flux to time of kepid 1164109. 9

2.4 Comparison between original light curve and flattened light curve. . . 11

2.5 Screen shot of code snippets of binning method. 13

2.6 Light curve of kepid 1164109 after binning. 13

2.7 Binned light curves of some PCs and Non-PCs. 14

List of Tables

2.1 Descriptions of column names. 6

2.2 Parameters of three TCE for kepid 1162345. 9

2.3 Parameter values chosen in the LightKurve flatten method. 10

Chapter 1

Introduction

1.1 Transit Photometry

A planet is too faint to be found directly by telescopes, so people study it by observing its host star. Since planets orbit around their host stars, they have some periodical effects on the stars such as gravitational pull and light blocking. When a exoplanet pass in front of its host star, it blocks a very small fraction of the star. By measuring the drops of the starlight, basic stats of the planet can be obtained such as period and size.

Because planets are in general very tiny compared to host stars, the drops of the starlight are very small. Therefore, some statistic methods are used to identify transits. The confirmed transits that passed statistic tests are called TCE (Threshold-Crossing Events). In *Kepler* DR24 data products (the data used in this thesis), the mean and median of TCE are about 12516 ppm and 125 ppm respectively (ppm stands for Parts Per Million). The distribution of TCE depths is shown in the following figures.

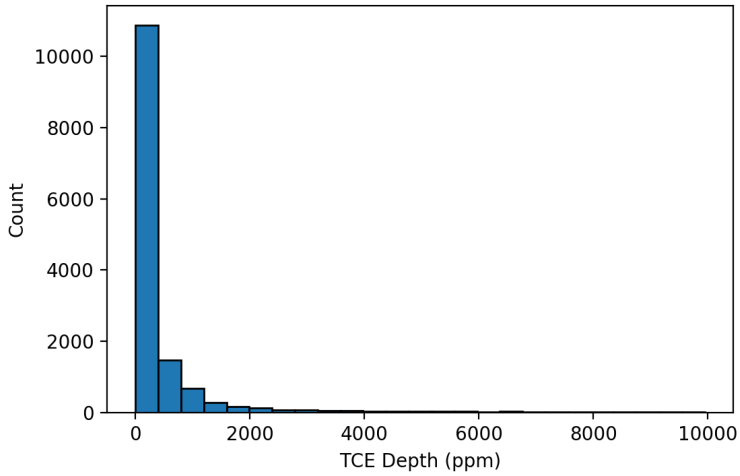

Figure 1.1: The distribution of TCE depths of *Kepler* DR24 data. There are 20367 TCE in the data and only 1831 are larger than 10000 ppm (less than 10%). The histogram shows that most of TCE depths are smaller than 500 ppm.

1.2 *Kepler Space Telescope*

Kepler Space Telescope is a space telescope for finding Earth-like terrestrial exoplanets. It was launched on March 7th, 2009 and was retired in late 2018. During its service, *Kepler Space Telescope* observes more than 500 thousand stars and finds about 2600 exoplanets.

The *Kepler* pipeline is a set of programming tools which can generate calibrated light curves which can then be fed into the algorithms to discriminate between planet candidates (PC) and other types of light curves. Threshold Crossing Events (TCE) are generated after Transiting Planet Search (TPS) which is part of the *Kepler* pipeline. And then we can identify planet candidates by analyzing the generated TCE.

Kepler team developed a tool called *Kepler Robovetter* [1] to identify PCs and false positives and it achieves a high accuracy (over 97%). It is a traditional algorithm

[1] https://github.com/nasa/kepler-robovetter

which is purely written in C++ and is very fast. However, it is difficult to understand and one can view its source code on the github. Therefore, try different method like machine learning may be a feasible choice.

1.3 Machine Learning

Machine learning is a subset of artificial intelligence and has been successfully used in many areas for a variety of tasks such as self-driving cars and optical character recognition (ORC) which can extract words and their meanings from images. Machine learning methods can even generate art works such as pictures which are nearly indistinguishable from art works created by artists. Therefore, it is natural to think that machine learning methods can also be applied to astronomy.

Machine learning can be classified differently based on different criteria. For example, if we predict a product's price which is a concrete value, we are doing a regression task. Otherwise, if we need to classify pictures as dogs or cats, then it is a classification task. Meanwhile, an algorithm is called supervised learning if we train the machine learning model by giving corresponding labels. Otherwise, it is a non-supervised learning algorithm. In this thesis, we input the TCE with corresponding category labels generated by *Kepler* pipeline and output a number 0 or 1 to indicate PC or non-PC. Thus, we use a supervised learning method to do a classification task.

There are a large quantity of machine learning algorithms for classification tasks such as logistic regression, decision trees and support vector machines. However, in this thesis, besides trying these machine learning methods, we focus more on deep learning methods, especially convolution neural networks (CNN).

Chapter 2

Data Preparation

As previously mentioned, we use *Kepler* DR24 data [1] with labels to train out model. It is a CSV file while other types are also supported. Note that the data contains training labels generated by autovetter which is basically a random forest technique. Random forest is also a supervised machine learning algorithm, thus it also needs labels to tell the algorithm which one is PC. The labels fed into random forest algorithm are classified by humans. The reason why use deep learning method to redo the classification task is because deep learning performs very well when trained with large amount of data. Therefore, we try to use the human-made and normal machine learning method generated labels to train a deep learning model in order to get better accuracy.

The labels contain four unique values: PC (Planet Candidates) , AFP (Astronomical False Positive), NTP (Non-transiting Phenomenon) and UNK (Unknown). There are 27 columns in the data including the label column named "av_training_set". Each other column represents a property of one TCE and some of the columns are more important for our classification task, such as "tce_period", "tce_duration", "tce_prad", "tce_depth", etc. Meanings of the some column names are listed in the

[1] https://exoplanetarchive.ipac.caltech.edu/docs/Kepler_TCE_docs.html

following table. [2]

Column Name	Definition
tce_period	time interval between two consecutive transits in days
tce_depth	starlight drops in ppm
tce_prad	the planet radius in Earth Radii

Table 2.1: Descriptions of column names. The listed properties are more important for identifying PC.

The CSV table needs to be preprocessed. First of all, we need to drop all data labeled with 'UNK'. Then since deep learning algorithms only deal with numerical labels, we need to label PC and non-PC as 1 and 0 respectively. Non-PC includes both AFP and NTP because we do binary classification to find out planet candidates. After processing the basic properties for each TCE, we also need to download their light curves.

2.1 Download Light Curves

Light curves are downloaded based on the column name "kepid" in the *Kepler* DR24 data table previously discussed. There are about 16 fits files which is composed of four year observation corresponding to a kepid. All the light curves can be found on the website *https://archive.stsci.edu/pub/kepler/lightcurves/0020/* . A tricky thing is that there are 20367 TCE in the table containing more than 330,000 light curve files. Downloading these files can be very time consuming (about 2 weeks by estimation). Therefore, I scrape the webpage in parallel to filter the light curves data and download them.

Take kepid 000757137 as an example. The URL of the light curves for this TCE

[2] https://exoplanetarchive.ipac.caltech.edu/docs/API_tce_columns.html

is *http://archive.stsci.edu/pub/kepler/lightcurves/0007/000757137* . There are 17 light curve files as the following screen shot shown.

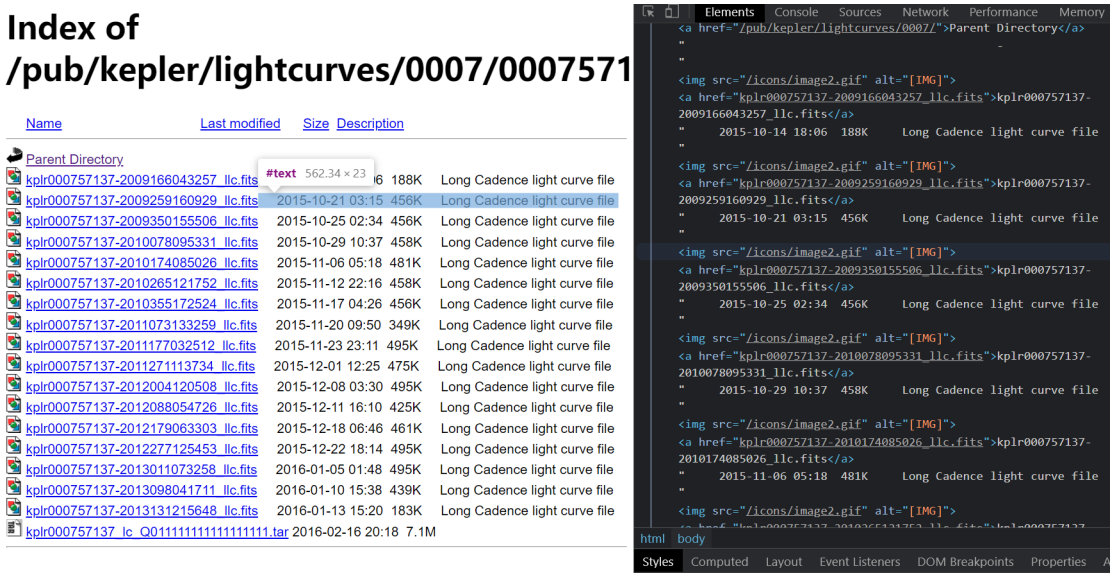

Figure 2.1: The screen shot of webpage containing light curves of kepid 000757137

We can find the hyper link from the anchor tag and navigate to the URL of the desired data. The code is written in Golang, which is very good at concurrency. All code can be found on *github* . In short, the basic procedure is

1. Parse all the hyperlinks from the root URL: *http://archive.stsci.edu/pub/kepler/lightcurves* . Join the current URL with the parsed hyperlinks to generate new URLs.

2. Launch new goroutine to search from the newly generated URLs.

3. Get the hyper links from the child URLs. If the hyper links are fits files, then download them to a local directory in parallel. Otherwise, as the procedure 1, get new URLs and launch new goroutine to do the search recursively.

Another problem is if I search from too many URL simultaneously, my IP will be blocked temporarily. Therefore, I need to control the total amount of processes.

After some attempts, I find that 200 processes are good enough and all light curves can be downloaded within 3 hours. The point is data are usually very large and complex in the field of astronomy, thus dealing with data efficiently is important.

2.2 Light Curves Preprocessing

As previously discussed, the depths of TCE are in general very small and the median value is around 125 ppm. The following figure is the four-year observation data. Thus we need to fold the light curves and bin them in order to magnify the effect. Moreover, we can see some data points in the top right side of the figure. We also need to remove the outliers to improve out model's prediction accuracy.

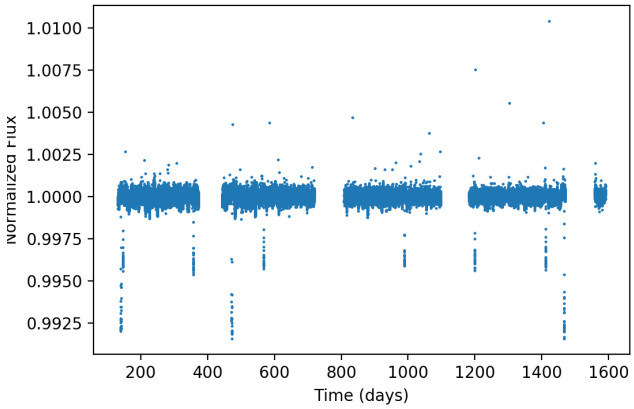

Figure 2.2: Plot of normalized flux to time of kepid 11442793. The flux is normalized by dividing the original flux by their median value.

In addition, there is long-term trend which is clearly not a transit signal in the light curves for some kepids. For example, we can see a clear periodical fluctuation in the light curve for kepid 1164109 as the following figure shown. The fluctuation has bad effects for the deep learning model, thus even though we don't have to deal with the long-term trend, removing the trend may increase the model's performance.

Finally, since there may be more than one planet associated with a kepid, we re-

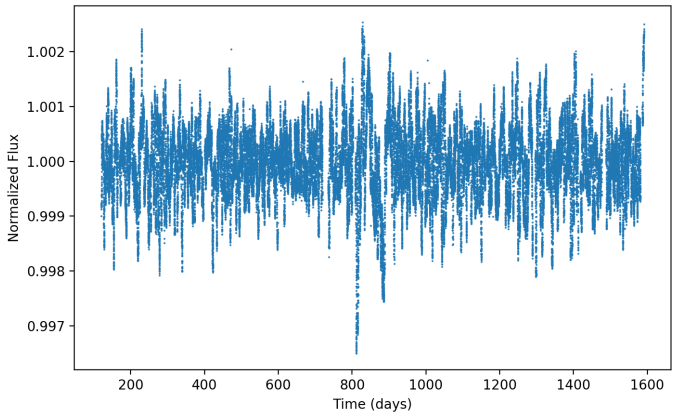

Figure 2.3: Plot of normalized flux to time of kepid 1164109.

move signals of other planets in the same system to avoid interference. The complete progress of light curves preprocessing is described as follows.

2.2.1 Remove Multiple Signals in the Same Stellar System

The following table shows some useful parameters of TCE for kepid 1162345. When preprocessing kepid 1162345, we fetch all the TCE — there are 3 TCE in the stellar system. Then we process the TCE one by one, with other two TCE removed. The remove logic is simple: mask out the data points within the range [center-duration, center+duration] and return the new flux and time pairs.

tce_plnt_number	**tce_period (day)**	**tce_duration (hr)**	**tce_depth (ppm)**
2	0.831850	2.392	2.636
3	0.831833	2.181	27.100
1	0.831777	2.349	24.270

Table 2.2: Parameters of three TCE for kepid 1162345. When processing the TCE for planet 2, we remove the signals for planet 1 and 3.

2.2.2 Remove Outliers and Long-term Trend

Before doing further processing to the light curves, we need to remove the outliers first to increase our model's performance. We just use a simple sigma clip to do the job. The method firstly derive the median and standard deviation value of the flux (m, σ). Then any points not within the range of $m \pm n\sigma$ are removed, where n is a parameter with default value of 3.

Then we should remove the long-term trend in the light curves. In this thesis, we use a python package called lightkurve [3] which internally uses Savizky-Golay filter to remove the low-frequency trend. The lightkurve package contains an python object called LightKurve which has a "flatten" method. After some experiments, we change some default parameter values as the following table shown:

	window_length	**polyorder**	**break_tolerance**	**sigma**
values	201	2	40	3

Table 2.3: Parameter values chosen in the LightKurve flatten method. The detailed descriptions of these parameters can be found on the lightkurve official website.

Take kepid 1164109 for example, the following figures are original normalized flux and flattened flux. After removing the long-term trend from the original normalized light curve, the transit signal becomes more clear.

2.2.3 Fold and Bin Light Curves

First of all, we fold the flattened light curves to increase the data points within the range of the transit signals. Then we move the transit signal in the center of the folded light curve. This may increase the performance of the deep learning model

[3] https://docs.lightkurve.org/

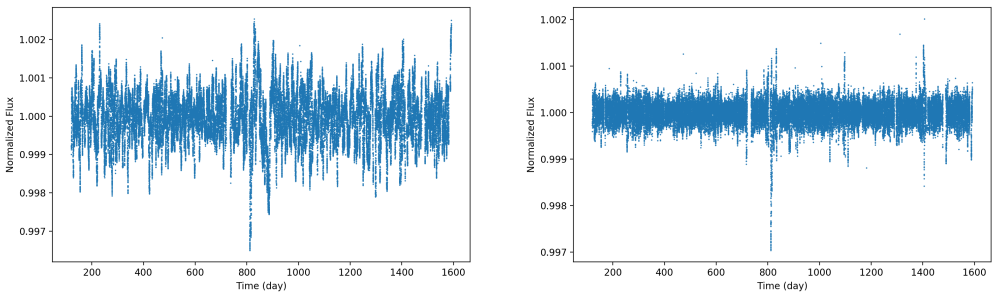

Figure 2.4: Comparison between original light curve and flattened light curve. The right panel is the normalized light curve of kepid 1164109 after flattening.

since the data is more "clean". Then we need to bin the folded light curves for the following benefits.

- There are too many data points in the folded light curves (around 65000 for each folded light curve). As we know, the number of "useful" data points which are within the range of a TCE's duration is very small. Therefore, most of the 65000 data points are nearly "useless" and will waste a lot of computing power.
- Binning can make the light curves a bit more smoother and reduce variants. Since a complex machine learning model tend to fit the variants and causes over-fitting, which is very bad for the model generalization. Thus, the performance of the over-fitted model tend to perform bad on the unseen data.

There are some python packages such as lightkurve to do the binning operation, however, we can only choose the bin width (w) or number of bins. And the distance between each bin (d) is equal to the bin width ($d = w$). But these two values can be different: if $d < w$, the binned light curves can be more smooth and may can improve the model's performance. Therefore, we need to write our own bin method. Even though the logic of binning method is simple, it is a little bit tricky to implement.

For example, if the bin width is so small that there is no data points in this range, what value should we choose to represent this bin? In this thesis, we choose the median value if the bin has no data points. Another question is should we choose the same bin width and distance for all the light curves? After some experiments, we don't find a good criteria to tune the binning parameters for each light curve. Therefore, we use the same parameter configuration for all the light curves. Finally, the performance for this method is very important because we need to do many training experiments, thus we have to generate preprocessed light curves for all the TCE multiple times.

As mentioned previously, the TCE duration is very small compare to the period. Take the TCE with kepid 1164109 for example, the period is around 622 days while the period is only 12 hours. Thus we would like to focus more on the data points within the TCE durations and have more bins in this range. In short, the binning logic is the following. Firstly, we pass the duration as a parameter to the bin function. Because we have already arranged the transit signals in the center of light curves, we choose the data points within the range of [-duration, +duration] around the center. Therefore we divide the light curve into three parts: before transit, during and after transit. And for each part, we apply the median bin method discussed above. Then we count the number of the data points (n), and the number of bins in the transit range is under most situations ($n/5$). If we denote the total desired number of bins (the input shape of the model) as n_total. For the other two parts, the number of bins are the same, which equals to $(n\_total - n/5)/2$. The Fig. 2.6 is the screen shot of parts of the binning code.

The binning operation is the last step of the preprocessing. The Fig. 2.6 is an binned "light curve" of kepid 1164109.

After normalizing, removing transit signals in same stellar systems, outliers, low-frequency signals, fold and bin the original light curves, we obtain a final "light

```python
def another_bin(x, y, num_bins, duration=None, normalize=True):
    """x, y represent folded light curves"""
    assert len(x) == len(y)

    tce_width = 0.5*duration
    x_min, x_max = x[0], x[-1]
    x_mid = (x_min + x_max) / 2
    tce_min, tce_max = x_mid - tce_width, x_mid + tce_width

    i1 = np.argwhere(x<tce_min).ravel()
    i2 = np.argwhere((x>=tce_min) & (x<=tce_max)).ravel()
    i3 = np.argwhere(x>tce_max).ravel()

    n2 = int(min(len(i2)/5, 0.4*num_bins))
    if n2 == 0:
        return median_bin(x, y, num_nins, normalize=normalize)

    n1 = n3 = (num_bins - n2) // 2 + 1

    a = median_bin(x[i1], y[i1], n1, bin_width_factor=1.0, normalize=False)
    b = median_bin(x[i2], y[i2], n2, bin_width_factor=4, normalize=False)
    c = median_bin(x[i3], y[i3], n3, bin_width_factor=1.0, normalize=False)
```

Figure 2.5: Screen shot of code snippets of binning method. In the figure, n2 is the number of bins we want in the range of [-duration, +duration], which equals to $(n\_total - n/5)/2$, where $n = len(x)$ is the total number of data points. Sometimes a there are too many data points within the range and exceeds the total bin number, thus we need to use **min** function to limit the bins in the transit.

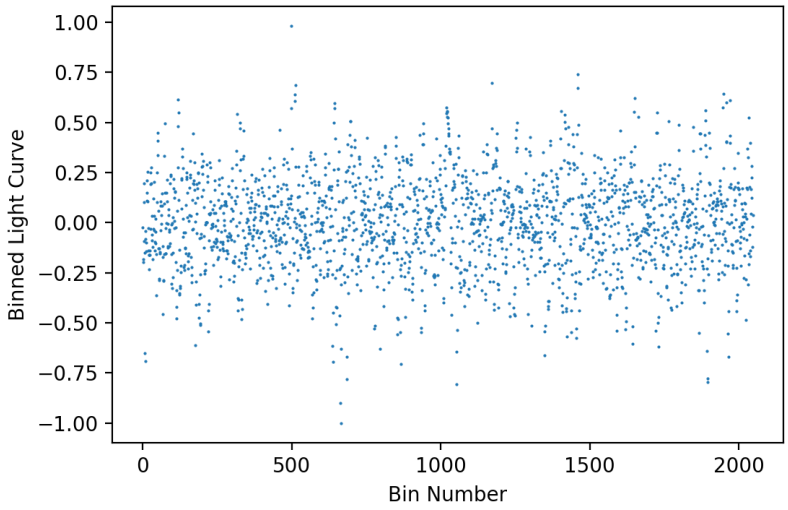

Figure 2.6: Light curve of kepid 1164109 after binning. The binned light curve contains much less data points compare to Fig. 2.4, thus can save lots of computational time. In addition, kepid 1164109 is not a planet candidate.

curve" with shape of (2048, 1). Since all light curves have the same shape after preprocessing, we can then feed them into machine learning models finally. The Fig. 2.7 compares the binned light curves of planet candidates with non planet candidates.

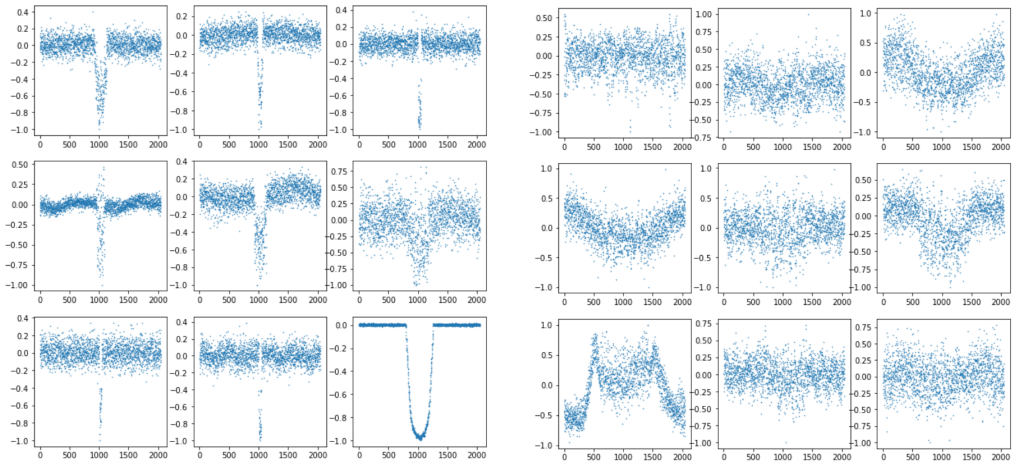

Figure 2.7: Binned light curves of some PCs and Non-PCs. The left and right panels are PCs' and Non-PCs' binned light curves respectively. We can clearly see the transit signals in the subplot of PCs and the light curves are a bit more messy for Non-PCs.

Chapter 3

Deep Learning Models

Bibliography