MACHINE LEARNING ENGINEER NANODEGREE

Capstone Proposal

OTHER EARTHS

Exoplanet-star Classifier

Darsh Kaushik
October 20th, 2020

Domain Background

The question of the century perhaps, is there another blue dot of life out there in this humungous universe? Are we alone? Is there any other ball of mass which could sustain life like our Earth?

As the space technology advances day by day, the researchers hunting the potential exoplanets are bombarded with data from the satellites observing the distant solar systems. And I fortunately stumbled upon one such dataset on Kaggle from the **Kepler's second mission: K2**.

Using the **transit method** to detect brightness changes, the K2 mission entails a series of sequential observing "Campaigns" of fields distributed around the ecliptic plane and offers a photometric precision approaching that of the original Kepler mission. After looking onto a particular solar system for longer periods, whenever a revolving exoplanet happens to be in the path of the satellite and the sun, the intensity of light (flux) received by the satellite from the star dims, indicating the presence of a potential exoplanet.

In this project my main focus would be to use the time-series data of flux received by the satellite for pattern recognition and identify the stars with potential exoplanets revolving around them.

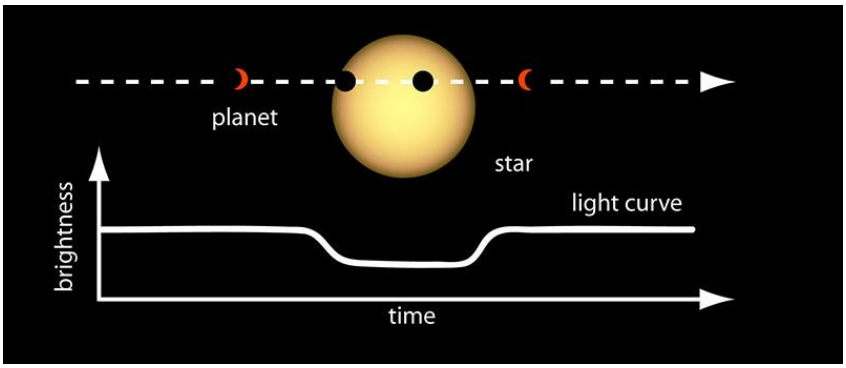


Figure: The Transit Method
Courtesy: NASA

Kepler Mission Overview

Transit method for detecting exoplanets

Problem Statement

Identifying the solar systems with potential exoplanets using the sequential data of light intensity captured in K2 mission.

The associated tasks include:

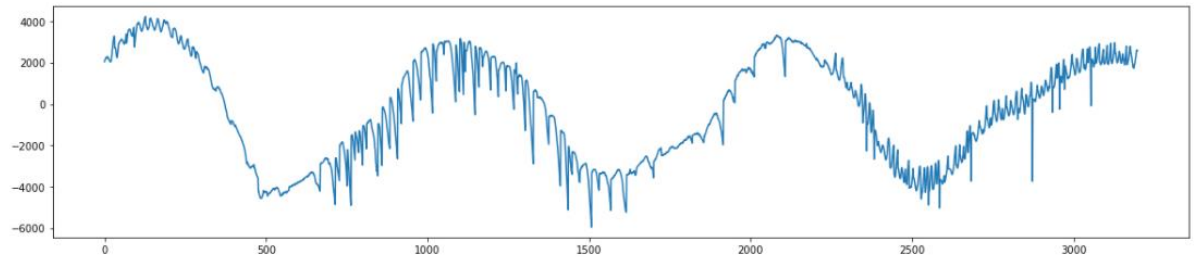
- Realizing periodic patterns in the flux data to mark the presence of revolving bodies
- Choosing an algorithm robust enough to capture the sequential, local and spatial cues present in the data.
- Smoothing out the data to remove noise and enhance the dimming in flux.
- Mitigating the effects of imbalanced classes in the dataset

Datasets and Inputs

I found a Kepler labelled time series dataset named as "Exoplanet Hunting in Deep Space" on Kaggle. The datasets were prepared late-summer 2016 and over 99% of this dataset originates from Campaign 3 of the K2 mission.

It contains the flux data of 5,087 stars in training set and 570 in the test set, where each star is labelled as 2 if it is an exoplanet star and 1 if it is a non-exoplanet-star. For each star there are 3,197 columns with flux values over time. When plotted, we can get the curve of flux vs. time as shown below.

Flux vs. Time graph for an exoplanet star:



The data is highly imbalanced, out of 5,087 stars only 37 stars labelled as exoplanet stars in training set and only 5 out of 570 stars in the test set.

Kaggle Dataset

Solution Statement

If a star is watched for long periods over several months or years, we can get a long time-series data of the light intensity or flux observed by the satellite. If a star tends to have planets revolving around it, regular dimming or lowering of flux may be observable over such long periods of time and this may serve as an evidence of presence of an orbiting body around the star making it a 'candidate' for further investigation.

In order to classify such stars, it is important to detect the regular spatial patterns over time (which may involve lowering of the flux) to estimate whether there is any body revolving around the star. From the perspective of a deep learning model it is possible to use consecutive fully connected layers but as the feature-space is large the training will take more time as we would need larger number of layers.

A better and robust approach would be to use **convolutions** on these time-series followed by fully connected layers in order to better detect the local patterns emerging in the data (such as lowering of the light intensity). Also, the training might be faster as the number of trainable parameters would decrease sharply. Also, it would provide the flexibility to apply various convolution kernels on the data to detect different kinds of characteristics in data to help better classify the stars.

Also, some data augmentation (mentioned in the Project Design Section) steps will be needed in order to mitigate the imbalance in classes. Evaluation metrics will have to be chosen accordingly.

Benchmark Model

Although there are not many notebooks available on Kaggle for this dataset, I was able to find a particular notebook (linked below) which used some of the techniques for the deep learning model mentioned above and was able to attain an accuracy of more than 90% on the test set. Therefore, I would consider the model in this notebook as my benchmark model.

Benchmark model

Evaluation Metrics

Although I have planned to take steps for data augmentation, along with **accuracy** I would also take into consideration the **precision** and **recall** as there are very few positive examples. I expect all the three metrics to be at least higher than 80%.

Project Design

Data preprocessing which will be very crucial for this dataset because of the class imbalance and noise in the readings. The data cannot be used directly for making batches for training purposes. It is important to have balanced batches in order to attain proper training results. Therefore, the first step would be to smoothen the flux time-series and create balanced batches for training.

1-dimensional uniform filters in scipy can be used to smoothen out the time series and remove noise.

To counter the imbalance in classes, I will have to **repeat the positive examples** multiple times such that each batch for training contains equal number of examples from both classes. There is also a possibility of **data augmentation** by rotating the time-series of positive examples through time. Using **weighted binary cross entropy loss** might also be helpful.

Some customary data exploration like plotting the flux vs. Time plots before and after the smoothening may be needed to check whether the data is not completely flattened out.

Regarding the model architecture, in order to identify exoplanet stars, the **local recurring patterns** such as lowering of flux must be detected from the available flux time-series. From all the possible approaches of Deep Learning, I have planned to use a **combination of 1-dimensional convolutions and pooling** followed by **fully connected layers**. 1-d convolutions act like local filters for features and helps to extract the spatial aspects of the data, pooling will help in further amplifying the detected features and reduce the dimensionality to be further apply the fully connected layers on.

I also presume batch normalization will be helpful in training the model efficiently.

After training the model it will be important to note the recall and precision on the test data to get the accurate picture.

Tools, resources, libraries and frameworks I am planning to use: Python 3, Jupyter Notebook, Pytorch, Numpy, Pandas, Matplotlib, Sci-kit learn, Scipy.