Helping the search for far-away exoplanets using deep neural networks and Big-Data

Arad Zekler – 305600579, Naor Dahan - ,308399393
Dolev Hindy - 312126643

Software Requirements Specification

Document

Version: (1.0) **Date: (20/12/2020)**

Table of Contents

1. Introduction

In recent years different scientific fields are experiencing major changes due to developments in processing power and in data-driven methods to process and analyze data. Different fields such as Biology, Chemistry and Astronomy are redefined by powerful models and tools that helped scientists find answers to questions that were just until now out of reach.

Since his launch in 2009, NASA's Keppler Space Telescope has discovered more then 6000 (1) exoplanets orbiting stars in the Milky Way Galaxy. That and other telescopes currently in use are one of the main drivers to the growth of Astronomical datasets in recent years, both in size and depth as more multi-wavelength and sequential observations are taken and processed together with Big Data technologies.

Another field that has been on the rise in recent years is Machine Learning – specifically Deep Learning – as the goal of it is to combine input features into a useful output, for example, we wish to build an image classification model that learns to classify if an image of a pet is a dog or a cat, by using already-classified data we train a machine to make that prediction for us.

These two fields now intersect in the classification and detection of far-away exoplanets using features such as transit-signals, temporal illumination and massive quantities of data from observations. Using supervised learning networks – as in learning algorithms that are used to predict a target result from a set of features, we could build a predictive model which will provide reliable results as to whatever a planet is an exoplanet or a false-positive (such as another body passing by, background noise...)

1.1 Purpose

NASA's Kepler Space Telescope was designated to estimate the frequency of which Earth-sized exoplanets appear in our galaxy, but these planets (in terms of size and illumination) are of the edge of the telescope detection spectrum. At first the samples were (and still) analyzed by astronomers who classify each sample, but in light of the recent developments in data science, astronomy and deep learning (2) we put our focus in helping scientists and enthusiasts with the search for new earths.

By looking at the data from thousands of observations from the NASA Kepler satellite-telescope we strive to develop the best possible model for detecting exoplanets orbiting far-away stars in our galaxy, together with a proper development pipeline and API to create a better user experience for inputting and outputting data.

1.2 Scope

The product itself is a deep learning model, comprised of a deep neural network which when entered data through a designated API, will output a predictive result about the particular data entered.

As stated, the model will be packaged with a user-friendly API and documentation that will make the process of making a prediction easier for the user.

1.3 References

(1) Identifying Exoplanets with Deep Learning: A Five Planet Resonant Chain Around Kepler-80 And an Eight Planet Around Kepler-90, December 12th, 2011
(2) Dalya Baron, Machine Learning in Astronomy: A practical overview, April 17th, 2019
(3) https://exoplanetarchive.ipac.caltech.edu/docs/Kepler_KOI_docs.html
(4) https://exoplanetarchive.ipac.caltech.edu/docs/API_kepcandidate_columns.html, DOI 10.26133/NEA4
(5) https://choosealicense.com/licenses/gpl-3.0/

2. The Overall Description

The product is a DNN (Deep Neural Network) model packed inside an easy-to-use API for making predictive results about exoplanets from previous observations (contained in a dataset).

2.1 Product Perspective

This product is independent and self-contained project. The learning process will be made through a Dataset (3) provided by Caltech, and as the end result – A Machine Learning pipeline for finding exoplanets will be provided.

The general factors that affect the product and its requirements are mainly hardware-related, as deep learning continues to tackle problems using deeper still neural networks, processing power becomes a large issue. Normalizing large amounts of data and feeding it to a neural network will require large amounts of both memory and processing power and will be a limiting factor during the project.

2.1.1 System Interfaces

The model will require a well-documented API that will allow the user to input data and receive an output that will include a prediction but also visualization and other details about the prediction itself. The model itself will be able to interface with Windows 10/Linux systems as long as the required libraries are installed.

2.1.2 Interfaces

The model itself will be wrapped in an API that will include an easy-to-use CLI (Command Line Interface) for inputting and outputting data and tuning the model. The CLI will be the main interaction point but the code itself will also be documented and open-sourced for easy modification.

2.1.3 Hardware Interfaces

The model will come pretrained so there aren't GPU (Graphical Processing Unit) requirements in the user system unless he will want to modify the existing model and retrain. In that case a modern CUDA enabled GPU will be required.

In term of memory constraints, the user's system will need to have at least 2GB of RAM. There is a high chance the software total weight will be around 100's of MB to 1's GB of memory.

2.1.4 Software Interfaces

All future specification will come in the requierments.txt packaged in the product. The software itself will be written in Python 3.0. Further requirements may be needed as project development advances.
Also, an updated version of Microsoft Windows 10/Linux (preferably Ubuntu) will be needed to minimize bugs.

2.1.5 Communications Interfaces

Different web services such as Google Collab, AWS, Spark etc. will be included in this section when used (further down development). The final product will not require network connectivity at all (as long as the data is provided locally by the user).

2.1.6 Site Adaptation Requirements

No site adaptation is needed to run the mode. Any regular PC with the hardware and software requirements listed above will suffice to run the product.

2.2 User Characteristics

This model is intended for astronomers and enthusiasts alike, Python knowledge is not required but will make interaction with the model and functions easier.
No need of prior knowledge in Machine/Deep Learning but this too could help make things a bit clearer for the user.

2.3 Constraints

This section will provide details about our constrains during development and production.

From software perspective we will need a unified working environment which can be achieved by using Anaconda. This will help contain all the required libraries and their versions updated and in one place.
From hardware perspective, for the learning process we will need at least 16GB of RAM and a CUDA abled GPU. The Dataset itself weights in about 32GB so more processing power will shorten learning times. If a strong local GPU is not an option it is worth checking other IaaS options (such as AWS).

Also, there is no private data involved and all the code is open-sourced.

2.4 Apportioning of Requirements.

If trying to identify the requirements that may lead to a delayed deployment of other versions the main concern is processing power vs model depth. A model contains a huge neural network could be trained in an unrealistic time and so delay the whole deployment of the product and we should take note of the training time and hyperparameter-optimization into account.
During training we might compromise to get results in a specific time to achieve a model that could theoretically be trained on a PC.

3. Specific Requirements

This section contains information about the specifics of the system itself and its requirements.

In section 2 we talked about the requirements in orientation to the user. In here we will discuss the system in orientation to the developer and here will be Defined any other requirements not covered elsewhere in the SRS.

The system should take an inquiry (dataset which a prediction about is wanted), normalize the data and feed it into a pretrained deep neural net model which will output an accurate prediction. We define here that an accurate prediction is any prediction above 85% accuracy (this may be changes in revisited versions).

Accuracy is defined as ***Number of correct predictions*** *divided by* ***Total number of predictions.*** *This value is bound between 0-1 and will be used as our main evaluation metric to assess the performance of the final model (or a model created by the user. This evaluation metric will be computed during test and validation phases of the training (where a model is fed with an 'unknown' subset of the dataset to predict the same result we strive to predict during the train phase to determine how successful the model is).*

The system should have an API fully written in Python that packages functions for the user to make actions and communicate with the system itself, making the experience of making meaningful planet predictions easier. All the commands will be pipelined through a CLI.

The user should be able to make changes to the model itself, and train the model by himself if needed as in changing or adding layers, or tuning hyperparameters.

The system should contain a documentation detailing all of its functions and ways of use.

The system should give a visual representation about the learning process of a model.

The system may allow to input subsets of datasets from various sources and output a correct prediction.

3.1 External Interfaces

This contains a detailed description of all inputs into and outputs from the software system. It complements the interface descriptions in section 2 but does not repeat information there.

It contains both content and format as follows:

- *Name of item*
- *Description of purpose*
- *Source of input or destination of output*
- *Valid range, accuracy and/or tolerance*
- *Units of measure*
- *Timing*
- *Relationships to other inputs/outputs*
- *Screen formats/organization*
- *Window formats/organization*
- *Data formats*
- *Command formats*
- *End messages*

This section will be reviewed later as development continues.

3.2 Functions

This section details the basic functions that should be included in the final version of the system (other than the system's main purpose).

The system shall check the validity of the inputted dataset, and alter it to match the neural network input parameters if not.

The system shall output an error massage if faults were detected along the pipeline.

The system shall output the result to a CSV/JSON file format for easy handling.

The predictive output made by the system shall be detailed with the prediction metrics along with the prediction itself.

Further functions will be added during development.

3.3 Performance Requirements

The pre-trained model itself will not be performance intensive but specific changes will have to be made during training in order to usefully run on some personal computers.

3.4 Logical Database Requirements

The dataset used (4) from Caltech is available through an API or direct download to a local machine, and contains data collected by the Kepler Space Telescope during multiple observations and years, as stated: "Kepler Objects-of-Interest (KOI) activity table reports dispositions based on the final processing (DR25) of the Kepler data and a combination of automated and human-based vetting to produce a "best-knowledge" catalog of planetary CANDIDATEs and FALSE POSITIVEs for use by the Astronomical community in selecting KOIs for follow-up observations and further study. "

3.5 Design Constraints

This section will include design constrains of the model and final product. More will be added as development continues.

3.5.1 Standards Compliance

There are no regulatory or financial standards the product needs to be complied to. The product itself is under GNU General Public License v3.0 (GNU GPLv3) (5).

3.6 Additional Comments

Will be noted as the project advances.

4. Change Management Process

*Any changes to this document will be made by members of the project, will be logged in this section with the proper version number **before** the change was made.*

5. Appendixes

5.1 Issues

Issues during development phase and after the final assessment will be presented here.