Proposal

January 29, 2018

1 Machine Learning Engineer Nanodegree

1.1 Capstone Proposal

Patrick Daskas January 26, 2018

1.2 Proposal

1.2.1 Domain Background

I will be using data from the Kepler telescope (2009-2013). The Kepler mission was designed to observe thousands of stars to look for transiting planets. The mission looks for reductions in a star's brightness, and if the reduction occurrs on a set period, then the star could be considered as having planet candidates. From the data that it is collected, NASA has been able to detect thousands of new planets orbiting other stars. The largest solar system detected by the Kepler mission has been 7 planets. Just last year, Google created a convolutional neural network to detect exoplanets. In the system that had 7 planets, Google discovered an 8th. This finding was recently confirmed by NASA.

This will not be using the K2 data. After some hardware failures in 2013, the Kepler mission's progress came to a halt. With the failing hardware, NASA repurposed the Kepler telescope to acquire data that spanned much smaller timescales.

More information describing the Kepler mission and the K2 mission can be found here: https://www.nasa.gov/mission_pages/kepler/overview/index.html

More information about Google and NASA discovering an 8th planet by using artificial intelligence can be found here: https://www.nasa.gov/press-release/artificial-intelligence-nasa-data-used-to-discover-eighth-planet-circling-distant-star

Additionally, the technical paper describing the use of a CNN can be found here: https://arxiv.org/pdf/1712.05044.pdf

1.2.2 Problem Statement

The amount of data acquired by the Kepler telescope is enormous. How are exoplanets detected in all the data? Almost every star observed has over 4000 data points per quarter. There are approximately 70000 data points that describe the change in brightness and the time in which it occurred for each star. Add to that over 100000 stars with this type of data, and one can quickly realize that manual detection of exoplanets seems absurd. There are websites today that pull this data and have a userbase analyze each of the image sets. They are attempting to manually categorize stars as having or not having exoplanets.

I plan to build a convolutional neural network to automatically classify targets in the dataset. I will use images as inputs to the network and the model will learn to classify images that have planet candidates from those that do not. These images will graph the change in brightness for each target across time.

1.2.3 Datasets and Inputs

I am downloading "fits" files for roughly 5000 stars, and this data set is about 40 gigabytes. These fits files have been provided by NASA. With the intent to classify, I will download data that has already been categorized as planet candidate targets and false positive targets.

For the planet candidate targets, I am taking the data from http://archive.stsci.edu/missions/kepler/lightcurves/tarfiles/Exoplanet_KOI/. I will download each long cadence tgz file (_long.tgz) for all 17 quarters. Each contains a collection of fits files for various stars that were scanned in that quarter.

I also need a set containing false positives. For this, I will execute wget scripts by quarter from http://archive.stsci.edu/missions/kepler/lightcurves/tarfiles/wget_scripts/ for the false positives. These scripts will download long and short cadence files. To be consistent, I will remove all short cadence files (these files ended in sc.fits) and kept the long cadence files (ending in lc.fits).

I will use roughly 2800 planet candidate targets, and 2200 false positive targets. 80% of the data will be for training. The remaining 20% will be used for training. 33% of the training data will be used for validation. I may reduce the planet candidate targets to be equal to the false positive targets to use accuracy as a metric for evaluating the model, however if I do not, I will likely use the F1 score.

Using PyKE (https://github.com/KeplerGO/pyke), I intend to stitch together 17 quarters of data for each star. The data will then be normalized using PyKE. Normalization is necessary, as after each quarter, the telescope rotates (to maximize solar gains on the solar panels). Normalization is an expensive operation and will be done by splitting the work across multiple cores on the CPU. The normalized fits file contains datapoints that represent brightness and time. Using this data with matplotlib, I will create a picture of the change in brightness for a given star across 17 quarters. The file will be saved and used as inputs to the convolutional neural network. The dimensions of the image will be 3600 x 432. This is needed to capture small variations in the data (where the brightness dips minimally for smaller planets). This large image size will create memory issues, but with the use of a generator I will be able to load them in batches as training occurs (https://keras.io/preprocessing/image/, https://keras.io/models/model/ (keras.fit_generator)).

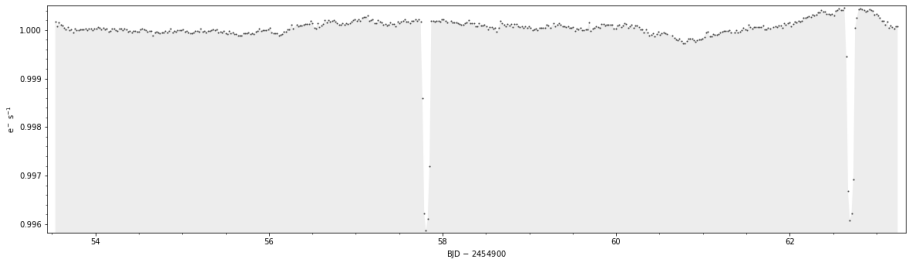
For the project, I will download the raw data from NASA and stitch together the full fits files that contain the brightness and time. For the reviewer, I will upload the images that I use for training, testing, and validation into the project's Github repository. Additionally, I will provide all scripts that I use for preprocessing (taking the quarterly fits files and stitching them, as well as normalizing and generating the image files in a png format).

1.2.4 Solution Statement

I will build a convolutional neural network to determine if a target star has planets. For the scope of this project, I will not attempt to create a model which can determine the number of planets for a given star. I will simply try to identify planet candidate stars from false positives. To do this, I will need to preprocess a significant amount of data. My dataset will be classified as planet candidates and false positives. The inputs will be images, and I will split them entire set into a

training, testing, and validation set. The training set will be 80%, and 20% will be used for the testing set. 33% of the training set will be used as a validation set. I may explore k-fold cross validation if time permits.

The graph below illustrates what I will attempt to detect:



Around 58 days, the star's brightness dropped and again at day 61. This is an indicator that this star likely has a planet. I would classifiy this image as a planet candidate.

1.2.5 Benchmark Model

For the benchmark model, I will construct a simple convolutional neural network. I will train this model and evaluate the performance. Afterwards I Will build a deeper CNN that leverages pooling and dropout. This new CNN should outperform the basic network.

1.2.6 Evaluation Metrics

I will try to balance the data set so that there exists equal planet candidates and equal false positives. I would rather not throw away data, so if I have unbalanced amounts of data between the two classes then I will likely use the F1 score to evaluate the performance of the model.

If not, I will be looking at loss and accuracy to evaluate the model. I will graph the history of the accuracy, loss, validation accuracy, and validation loss across several epochs. Overall, I will look to reduce the validation loss and increase the validation accuracy. Using Keras, I will also use the evaluate function against my test data to determine its respective loss and accuracy.

1.2.7 Project Design

Environment Python 3.0+ Keras Tensorflow Scikit-learn Numpy Matplotlib PyKE

1. Acquire dataset for 5000+ targets (across two categories: False Positives, Planet Candidates) for 17 quarters (2009-2013)
2. Build new dataset for the 5000+ targets where the data is merged and normalized
3. Convert the dataset containing the brightness and time to a graph using Matplotlib
4. Save all graphs as images

5. Split the dataset into training/testing/validation (80/10/10) sets.
6. Build convolutional neural network starting with 5 convolutional layers with dropout, max-pooling and batch normalization between each. The final fully connected layer will have 2 units which represents the two categories and will have a softmax activation. *Each convolutional layer will have a relu activation
7. Assess the accuracy or F1 score on the training, and validation set.

8. Use the model to make predictions against the test set and assess the accuracy.
9. Work to improve the accuracy on the test set by altering the hyperparameters.

1.2.8 Notes / Questions for the reviewer:

I intend to build my CNN using Conv2D layers. Is it possible though to convert my images so that I can use a Conv1D layer? I dont understand the difference between the two. Is there a benefit to doing this? The paper by Google and NASA describes using Conv1D layers (https://arxiv.org/pdf/1712.05044.pdf page 5-6).