CARLETON UNIVERSITY

COMP4107 FALL 2017

FINAL PROJECT

Exoplanet Hunting in Deep Space

1 Introduction

Exoplanet Hunting in Deep Space is a dataset currently featured on *kaggle.com.* This set is comprised of labelled time series data collected by NASA's Kepler telescope, to determine if stars have exoplanets orbiting them. "An exoplanet, or *extrasolar planet* is a planet outside of our solar system that orbits a star."[3] We examined the work of others with the dataset, and implemented our own solutions based on our findings.

2 The Dataset

The *Exoplanet Hunting in Deep Space* dataset includes thousands of observations of changes in light intensity, for thousands of stars. In the dataset, exoplanets are labelled with a *2*, whereas all other stars are identified by a *1*. The dataset is described as follows:

- Training Set: 5087 rows or observations; 3198 columns or features.

  Column 1 is the label vector. Columns 2 - 3198 are the flux values over time.

  37 confirmed exoplanet-stars and 5050 non-exoplanet-stars.

- Testset: 570 rows or observations; 3198 columns or features.

  Column 1 is the label vector. Columns 2 - 3198 are the flux values over time.

  5 confirmed exoplanet-stars and 565 non-exoplanet-stars. [1]

3 Machine Learning Libraries

Three machine learning libraries were used to develop our scripts, and create our neural network model: TensorFlow, scikit-learn, and imbalanced-learn. TensorFlow and sci-kitlearn were both approved in the project description, however imbalanced-learn was not. "Imbalanced-learn is a python package offering a number of re-sampling techniques commonly used in datasets showing strong between-class imbalance."[6] Although *imblearn* is not explicitly imported from *sklearn*, it is contained in the *scikit-learn-contrib* libraries. We are using the imbalanced-learn API to deal with the imbalanced dataset.

4 Methodology

Our initial tests were completed with the raw data so we could understand the effects of an imbalanced dataset on machine learning tasks. After those tests were complete, we researched different methods for handling data imbalances, and came up with three potential solutions: Naive Random Over-sampling (ROS), Synthetic Minority Over-sampling Technique (SMOTE), and NearMiss. As the names imply, ROS and SMOTE are used to over-sample a dataset, whereas NearMiss is used for under-sampling.

4.1 Under-sampling vs. Over-sampling

Under-sampling balances a dataset by randomly eliminating elements from the majority class. It is effective in reducing runtimes with very large datasets, however a potential side-effect is the loss of meaningful data. Conversely, over-sampling creates additional elements in the minority class by replicating data. Over-sampling ensures there is no data loss, however overfitting the dataset can be a concern. In general, over-sampling is more effective than under-sampling; we tested with both methods.

The imbalance-learn library provided all of the functionality for balancing our datasets as outlined below:

- imblearn.under_sampling.NearMiss: performed under-sampling on the dataset using the 'majority' setting for the *ratio* parameter.
- imblearn.over_sampling.SMOTE: performed over-sampling on the dataset using the 'minority' setting for the *ratio* parameter. Additionally, we set *k-neighbors* to 4, and *kind* was used with both the 'regular', and 'svm' settings.
- imblearn.over_sampling.RandomOverSampler: we used the RandomOverSampler (ROS) with the simple additon of the *ratio* parameter set to 'minority'.

4.2 Data Augmentation

Another approach we tried in balancing the dataset was to augment the data by increasing the number of data points corresponding to stars with confirmed exoplanets. Following Peter Grenholm, we translated the light measurements in time by a random amount, and added these points to the dataset.[4] The light intensity magnitudes were not altered. This method appears to expose the network to a greater number of data points corresponding to stars with confirmed exoplanets. However, it could be argued that training with this augmented data is of limited use, since translating the data in time does not replicate unique measurements obtained from an actual star. In addition, convolutional networks are designed to detect features that are translation invariant, and in this dataset, time is our feature space.

In order to obtain new data points without being able to obtain real data, the SMOTE oversampling method mentioned above provides a method to do this. It infers new data points from existing minority data points using k-means clustering. If we were to continue to experiment with this dataset, it would be valuable to explore the various parameters that we could give to SMOTE in order to generate new data. We could then measure the efficacy of these parameters against network performance on a real world test set.

4.3 Neural Network Model

For a star in the dataset with a confirmed exoplanet, the main features to detect are periodic drops in light intensity measurements. Some de-noising was applied to the data by NASA after being beamed down to earth from the Kepler Space Telescope. For examples of timeseries data from a star with a confirmed exoplanet, and a star without

a planet, see *Fig. 2* and *Fig. 3*. In an attempt to filter these features, we first used a network with three 1-dimensional convolutions.

In addition, we followed the structure of a high-performing model developed by Peter Grenholm on Kaggle[4]. The kernel size for each convolution was 11. The first layer scans for 8 features, the second for 16 features, and the third layer scans for 32 features. We performed max-pooling after each convolution, followed by two fully-connected layers. To prevent over-fitting, Dropout is used after every layer with a probability of 0.8 for the convolutions, and 0.5 for the fully-connected layers. Further details of the network can be seen in *Fig. 1*.

We also developed a smaller model with two convolutions, one max-pooling layer, two fully-connected layers and no dropout. The kernel in both convolutions is of size 6, smaller than the kernel used in the larger network mentioned above. In addition, we augmented the data manually without *sklearn*, by translating some of the positive (stars with confirmed exoplanets) results in time. We then created training batches with equal numbers of positive and negative results. Each training batch contained new negative results, along with augmented positive results in the manner developed by Peter Grenholm[4], which was discussed above.

The test batches were also balanced and augmented, then randomly shuffled. Since there are only 5 positive results in the entire test set, each test batch was of size 10. This model, along with data augmentation, outperformed our other models, obtaining precision and recall scores above 70%. However, we are highly skeptical of the results. The model performed poorly on an imbalanced test set. We will discuss the results of this model below.

5 Testing

We ran an array of tests in an attempt to fine-tune the outcome. All tests are included in the *test_results* directory of our project, in the file *results.txt*. Our scripts compile all data in a list structure, and append it to our *results.txt* file at the end of each test iteration. In other words, this is a large file that contains the outcome of many test runs; however each iteration is denoted by a header. For example:

```
Starting Batch Testing - Day/Time:  2017-12-18 13:03
```

Our Random Forest test results - see Alternate Approaches - are included in the *results.txt* file. The format of the Random Forest classifications are defined by the *scikit-learn* libraries we used. Numerous graphs, and scatter plots have also been created to visualize the dataset, and the affect of the *imbalanced-learn* functions. These are presented in the *Data Figures and Visualizations* section, with descriptions of each. Finally, since TensorFlow was used, the computational graph was generated, along with other data visualizations. Again, these are available at the end of the document.

The model with two convolutions and no dropout, mentioned at the end of *Section 5.3*, consistently attained precision and recall scores above 70%, when tested on a test set that was balanced and augmented using time translation. It is important to emphasize that this performance value was obtained from a balanced test set with augmented data, rather than a real, imbalanced test set with no time translation of positive results. A screenshot from a five epoch test run can be seen in *Fig. 10.*.

The model performed poorly when tested on an imbalanced test set with no time translation of positive results. It tended to produce more false negatives, rather than false positives, and did occasionally(<50%) correctly identify

a star with an orbiting planet, but not all 5 stars in the test set.

Our goal in creating this model was to create a classifier that could perform above 50% with the given training and test data augmentation and balancing. Due to difficulty in acheving any prediction metric results above 50% with our other convolutional networks and data balancing methods, we decided to simplify the network, shrink the kernel in the convolutions, and perform simpler, less sophisticated data augmentation. The code for the model is contained in the file *exoplanet_CNN_2convoV3.py*. This could be a starting point from which to build a model with more predictive power and generalization capacity.

6 Alternate Approaches

Working with an imbalanced dataset poses many challenges, and neural networks aren't always the best solution. We did research on alternate techniques, and found ensemble learning methods such as Random Forest, and various Boosting algorithms are often more useful. Ensemble learning typically relies on Classification and Regression Trees (CART) for classifying data.[7] We also explored the use of Recurrent Neural Networks.

6.1 Random Forest

Random Forest gets its name from the large number of CART structures used. In short, *k-datapoints* are selected from the dataset. They are used as input to a CART, and the data is processed to determine the decision nodes in a tree. Each point is bagged as a prediction, and the mode of all predictions is the result for a single CART. This process occurs for *n-trees*, and the mode of the results for all trees, is the winning point for the forest. The power of Random Forest, and other ensemble learning algorithms is the large number of trees. Generally speaking, accuracy increases with the size of the forest.[8]

We explored Random Forest to see if we could generate any meaningful results to use as a benchmark, or for comparison to our CNN implementations. Our first observation was based on the efficiency of the algorithm. When our *rf_train* function was called, the results were returned almost instantly. For example, in one test run random forest completed all tests in 7.04 seconds. This included the raw, over, and under-sampled datasets. Furthermore, precision was significantly higher with Random Forest: 0.99122 for the SMOTE and Raw data tests. Interestingly, the NearMiss classification had very poor precision, at less than 0.01% for most tests. We didn't spend a great deal of time examining the results, but did take away a weak verification that ensemble learning is more efficient and accurate. All of the cited test results, and others, are available in the *results.txt* file, in the *test_results* directory.

6.2 Recurrent Neural Networks (RNN)

Since RNNs are typically used for time-series data, it seemed like a good fit for the problem. However, since most of the data is consistent over time, it's difficult to visualize in a graph. We've included the model *exoplanetRNN.py*, but please note that it used some *keras* library functions. We created a couple of simple charts in Microsoft Excel to

gain an understanding of what we should be looking for, and generated one graph with our script for comparison. These are all available in the last section of the report. As expected, the graph itself gave little information, and we were unable to generate reasonable accuracy data. Given more time, this is an avenue we feel would be worthwhile exploring.

7 Conclusion

The *Exoplanet Hunting in Deep Space* was an interesting dataset that posed some challenges. Imbalanced datasets require a great deal of pre-processing to extract meaningful results. Although we were able to create a CNN that showed signs of learning, we feel we need more experience, and time to investigate the best approaches. One concern we had with our results was the precision hovering around 50%. We feel this indicates the model is predicting the same label for each iteration, however we were unable to derive a solution, and improve results.

Random Forest was introduced to examine other documented approaches for imbalanced data. As expected, random forest was far more accurate, and completed its classification of the data in just a few seconds. However, on a relatively small dataset such as the one we used, undersampling was highly ineffective. Again, this confirmed some of our research that was noted in an earlier section.

As mentioned, when we tested our second CNN model on a balanced test set with time translation of positive results, we were able to obtain precision and recall scores above 70%, and often as high as 90%. The model performed poorly when tested on an imbalanced test set with no time translation. The model identifies time-translated datapoints of stars with confirmed planets, better than the same datapoints with no translation. It was trained using the same data augmentation scheme, so perhaps this result is to be expected.

In our attempt to make up for a lack of positive data results, we created a model that was dependent on augmented data. This does not address the challenge of operating in real-world conditions where we cannot artificially create more data when we don't even know which star actually has an orbiting planet. It is also possible that this model is overfitted, since we did not use dropout. While the balanced test set with data augmentation was new to the network, and randomly shuffled each time, we cannot conclude that the model would perform similarly with a larger, more diverse, real-world imbalanced dataset.

It is possible that our data augmentation scheme created a distinguishable set of features in the positive results for the network to identify, and distinguish from the negative. This is a result of time-translating the positive features to generate new samples. Time-translating the negative samples would have created an even and fair test set; however, the time translation was used to create more positive samples when there was a shortage. Instead, we chose to put new negative samples in each training, and test batch. Nevertheless, it appears our use of time translation with the second model created an artificial test set that the network was able to easily classify.

This leads us to conclude that other data balancing techniques such as those provided in sklearn are more appropriate, and in order to obtain a higher performing model, we need to do more data pre-processing, and model experimentation.

8 Data Figures and Visualizations

8.1 The CNN Architecture

Figure 1: Network Model

8.2 Visualizations from our CNN

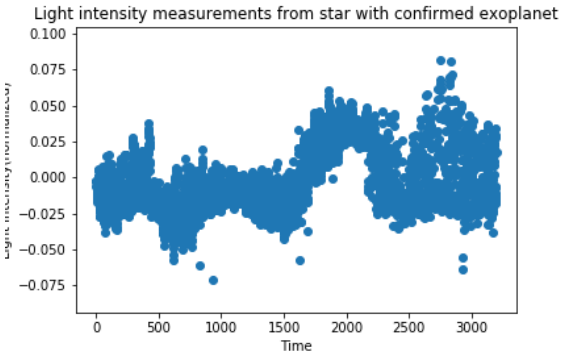

Figure 2: Normalized Light Intensity measurements from star with confirmed exoplanet

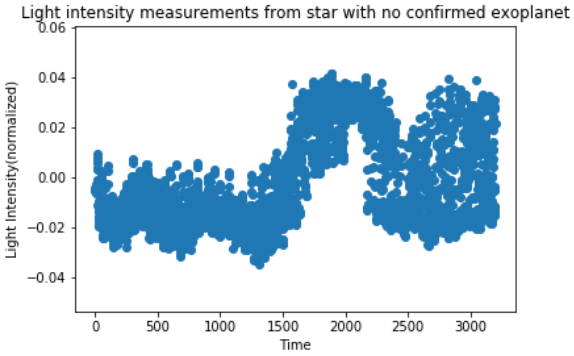

Figure 3: Normalized Light Intensity measurements from star with no confirmed exoplanet

```
[1 0 1 1 1 1 0 0 1 0]
accuracy in epoch 0 : 0.9
precision in epoch 0 : 0.833333333333
recall in epoch 0 : 1.0
[0 1 1 0 0 1 0 0 1 0]
accuracy in epoch 1 : 0.9
precision in epoch 1 : 1.0
recall in epoch 1 : 0.8
[1 1 0 0 0 1 1 0 0 1]
accuracy in epoch 2 : 1.0
precision in epoch 2 : 1.0
recall in epoch 2 : 1.0
[1 0 1 1 1 1 0 0 1 0]
accuracy in epoch 3 : 0.9
precision in epoch 3 : 0.833333333333
recall in epoch 3 : 1.0
[0 1 0 0 0 0 1 1 1 1]
accuracy in epoch 4 : 1.0
precision in epoch 4 : 1.0
recall in epoch 4 : 1.0
```

Figure 4: Test results over 5 epochs for network with two convolutions, two fc-layers, no dropout. Augmented training and test data, test batches of size 10.

8.3 Over and Under-sampled Datasets

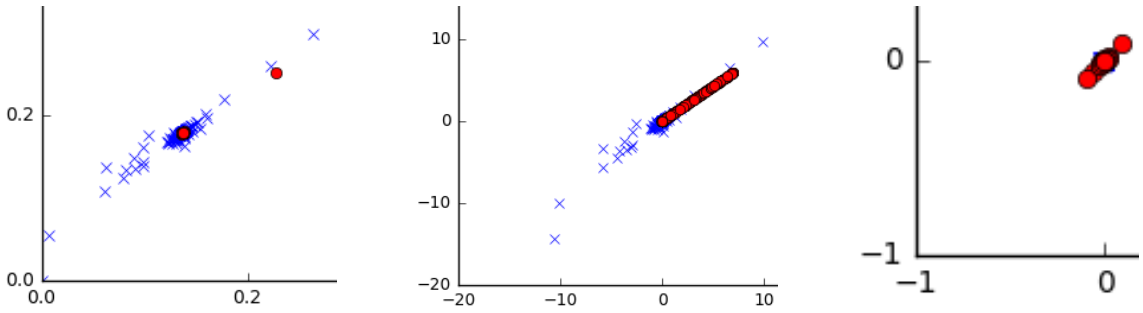

Figure 5: Close-up dataset images showing the distribution of data points. Left, the raw exoplanet dataset; centre, the over-sampled dataset resulting from SMOTE; right, the under-sampled dataset resulting from the NearMiss technique.

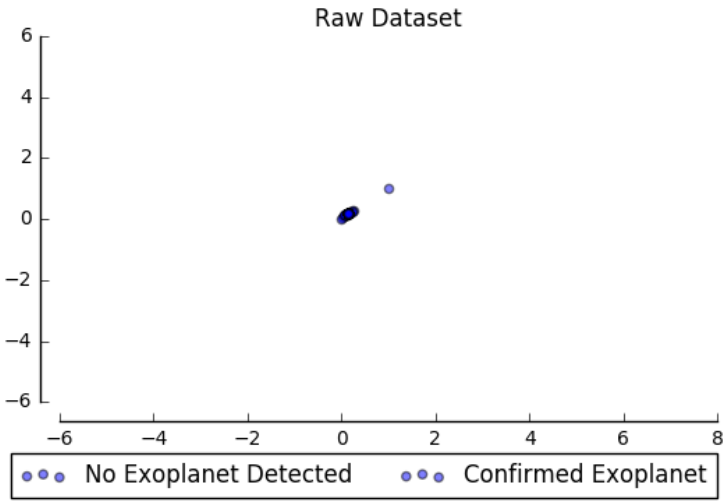

Figure 6: The raw exoplanet dataset showing the distributions of stars and exoplanets.

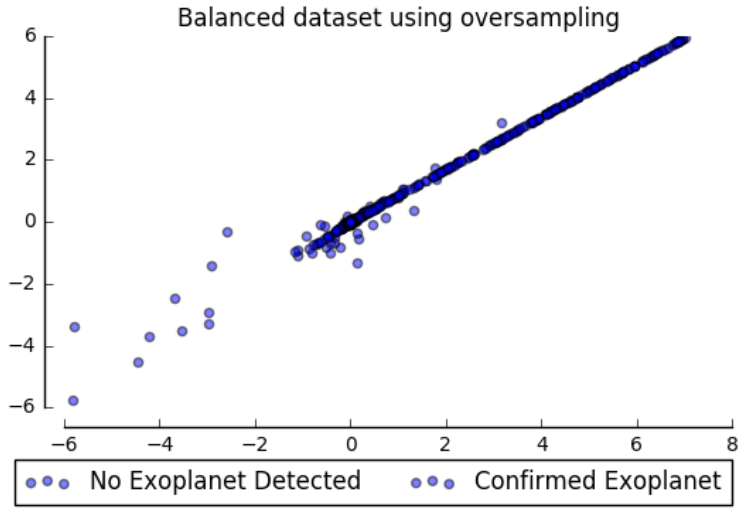

Figure 7: Another visualization of the over-sampled dataset showing a balanced distribution of stars and exoplanets.

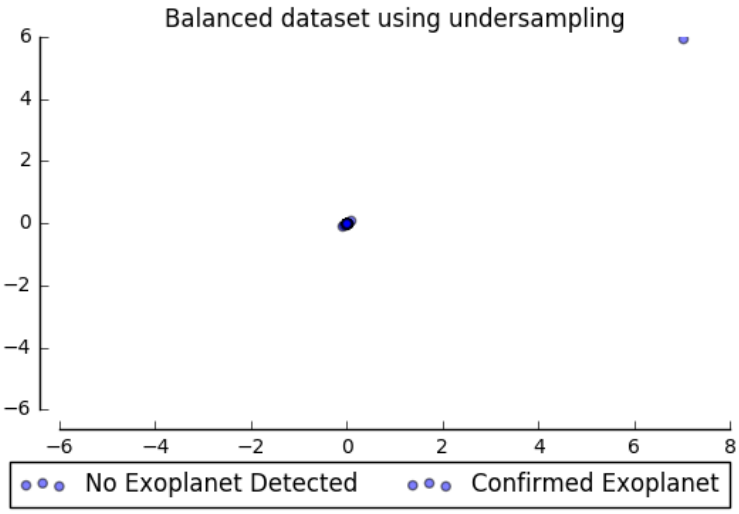

Figure 8: The under-sampled dataset shown as a distribution of stars and exoplanets. Given the lack of datapoints, it's easy to see why this dataset performed so poorly.

8.4 TensorBoard Visualizations

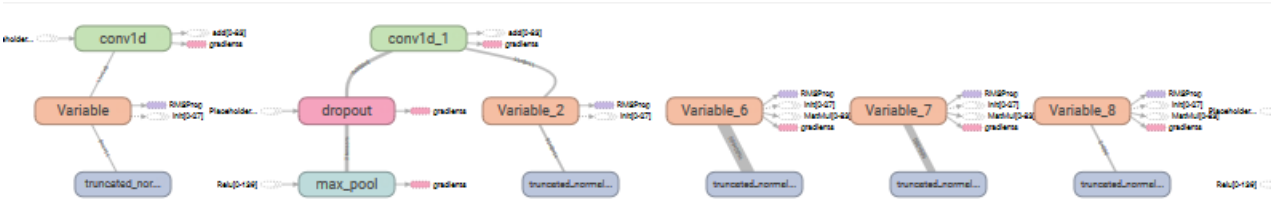

Figure 9: A segment of the computational graph for the CNN.

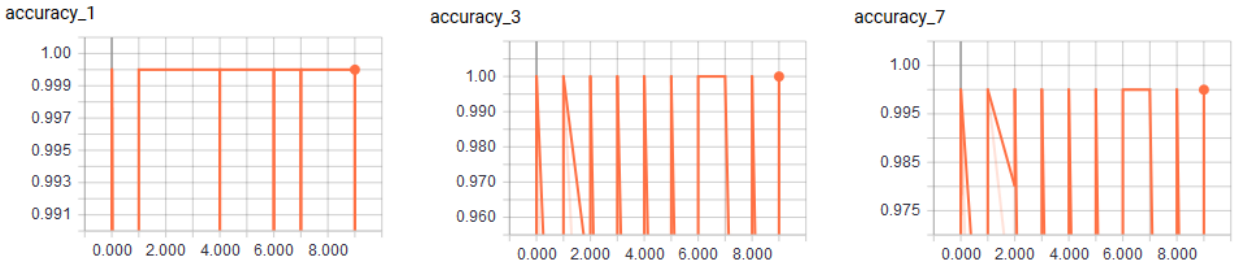

Figure 10: A sample of accuracy scalar visualizations extracted from TensorBoard.

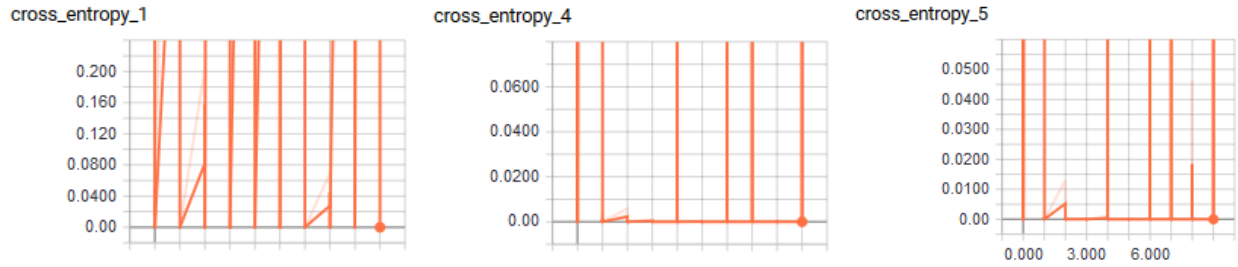

Figure 11: A sample of accuracy scalar visualizations extracted from TensorBoard.

Figure 12: A full context view of accuracy scalar visualizations in TensorBoard.

8.5 Microsoft Excel and RNN Visualizations

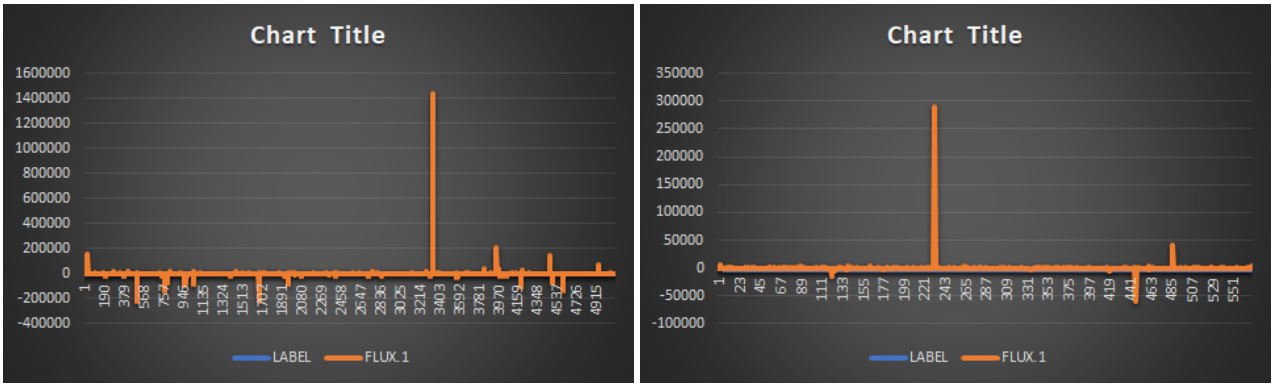

Figure 13: Microsoft Excel charts of the training, and test data. Left, the training set; right, the test set, which is what we would expect to generate with an accurate RNN model.

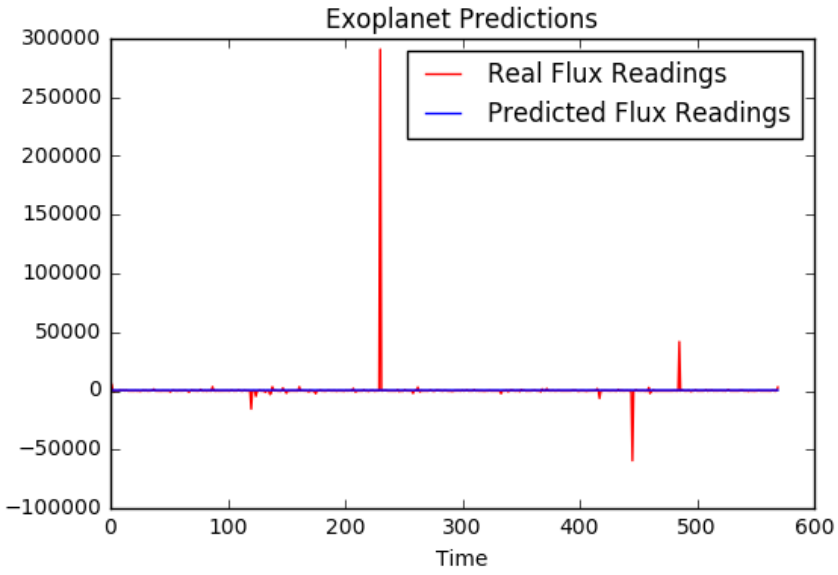

Figure 14: Predictions made by our RNN model presented as a graph. Although the predictions make sense, they don't show the extreme peaks, and valleys that represent the exoplanets.

References

[1] WΔ. *Exoplanet Hunting in Deep Space | Kaggle*, 12 Apr. 2017, www.kaggle.com/keplersmachines/kepler-labelled-time-series-data.

[2] "Mystery Planet" *Mystery Planet*, Kaggle, 17 Dec. 2017, www.kaggle.com/toregil/mystery-planet-99-8-cnn

[3] "Exoplanet." *Wikipedia*, Wikimedia Foundation, 17 Dec. 2017, en.wikipedia.org/wiki/Exoplanet.

[4] "Mystery Planet" *Mystery Planet*, Kaggle, 17 Dec. 2017, www.kaggle.com/toregil/mystery-planet-99-8-cnn

[5] Lemaitre, G, et al. "Welcome to imbalanced-Learn documentation!." *Welcome to imbalanced-Learn documentation! — imbalanced-Learn 0.3.0 documentation*, contrib.scikit-learn.org/imbalanced-learn/stable/index.html.

[6] scikit-learn-contrib. "Scikit-Learn-Contrib/Imbalanced-Learn." *GitHub*, 13 Dec. 2017, github.com/scikit-learn-contrib/imbalanced-learn.

[7] Blog, Guest, et al. "How to handle Imbalanced Classification Problems in machine learning?" *Analytics Vidhya*, 21 Mar. 2017, www.analyticsvidhya.com/blog/2017/03/imbalanced-classification-problem/.

[8] Saimadhu, Polamuri. "How the random forest algorithm works in machine learning." *Dataaspirant*, 1 Oct. 2017, dataaspirant.com/2017/05/22/random-forest-algorithm-machine-learing/.