Anomaly Detection from Kepler Satellite Time-Series Data

Nathaniel Grabaskas, Dong Si

Computing and Software Systems Department, University of Washington Bothell
ngrab@uw.edu, dongsi@uw.edu

Abstract. Kepler satellite data is analyzed to detect anomalies within the short cadence light curve. Traditional algorithms and neural networks are implemented to detect these anomalies. Modified Z-score, general extreme studentized deviate, and percentile rank algorithms were applied to intially detect anomalies. A refined Windowed Modified Z-score (WMZ) algorithm was used to determine "true anomalies" that were then used to train both a Pattern Neural Network and Recurrent Neural Network to detect anomalies. Windowed Modified Z-score and Windowed Generalized Extreme Studentized Deviate (WGESD) show better performance on curved data and a trained neural network had better performance on horizontal data.

Keywords. Recurrent Neural Network, Pattern Neural Network, Modified Z-Score, General Extreme Studentized Deviate, Percentile Rank, Time-Series Data, Anomaly Detection

1 Introduction

The Kepler satellite has been in operation since 2009 continuously looking at a star field in the Cygnus-Lyra region [1]. In 2013, two of its reaction wheels failed preventing it from looking at one continuous section of space. It was re-tasked in 2014 to look at a pre-defined section for 80 day periods and labeled as the K2 mission. During this time, it has been measuring the brightness of stars using Simple Aperture Photometry (SAP).

The main mission of this project is to detect exoplanets that are in orbit around distant stars. The transit method is used to detect these planets. This method can be simply stated: by watching a star continuously a dimming in the brightness of the star as the exoplanet passes in front of it can be observed [2]. When a possible transit is detected, it is labeled as a Kepler Object of Interest (KOI). Typically, multiple instances of the transit need to be observed before it will be labeled as a confirmed exoplanet.

This paper highlights two methods operating on the data from the Kepler satellite and uses both supervised and unsupervised machine learning to detect these transits. A transit is essentially an anomaly or an outlier in the dataset. Therefore, method one used modified Z-score, General Extreme Studentized Deviate (GESD), and percentile rank anomaly detection as a starting point and are modified to fit time-series data.

Method two, used the identification from the traditional algorithms and trained neural networks as a different method of detecting potential transits.

In the next section of this paper we will discuss the methods of pre-processing, traditional algorithms, and neural networks used. In section 3, we will show the results obtained from the various methods. Section 4 offers a discussion on these results and finally what future work is still needed.

2 Methods

2.1 Pre-Processing

Data Acquisition. NASA releases Kepler light curve data in the form of Flexible Image Transport System (.fits) files every 90 days. These files are hosted on the Mikulski Archive for Space Telescopes (MAST). All data used in this project is from KOI_Q16_short (contains KOI files) [3].

'Long' vs 'Short' cadence refers to the frequency of readings. In long cadence light curves the reading is taken every 15 minutes, and in short cadence light curves the reading is every one minute [4]. Long cadence files contain a limited number of samplings over 90 days; short cadence files contain many more samplings but vary in the period they cover. Short cadence files were used in these experiments due to the greater number of samplings and the higher likelihood of distinct anomalies being present.

Data forms. Short cadence light curve data takes two general forms: horizontal and curved. Horizontal data is easier to detect anomalies on as they will be above or below the mean. Any data that is not horizontal is considered curved data and is more challenging because anomalies can be between the trough and crest of a curve which would put them around the mean. See Figure 1 and 2.

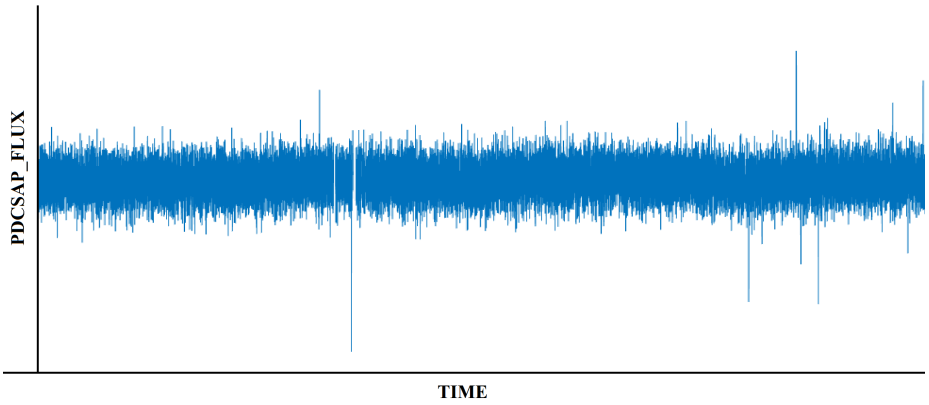

Fig. 1. Horizontal Raw PDCSAP_FLUX data from a short cadence light curve containing a KOI.

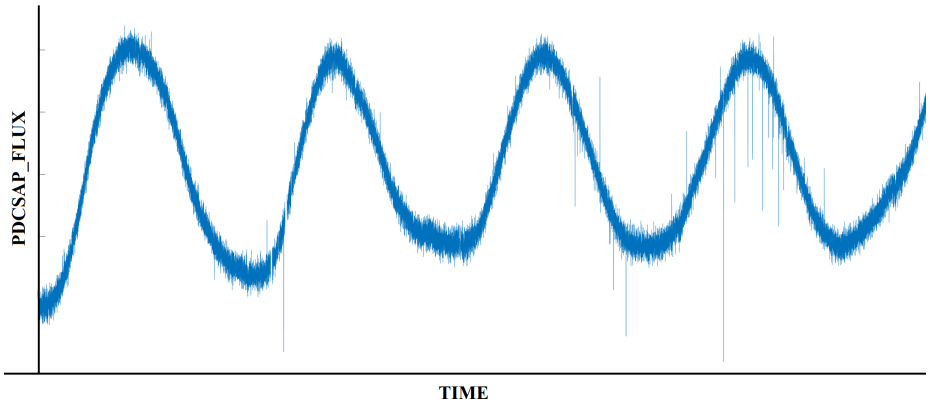

Fig. 2. Curved Raw PDCSAP_FLUX data from a short cadence light curve containing a KOI.

Data Reduction. Raw light curve data contains 20 unique attributes and each file consists of 5,000-50,000 entries. When attempting to detect transits the SAP_FLUX (Simple Aperture Photometry Fluctuation) and PDCSAP_FLUX (Pre-search Data Conditioning SAP Fluctuation) attributes are the main indicators. SAP_FLUX is the flux in electrons-per-second contained in the optimal aperture pixels collected by the Kepler Satellite [4].

All attribute fields except for SAP_FLUX and PDCSAP_FLUX are removed. Additionally, any instances of zero for either of these fields are removed as this will corrupt the detection methods.

The data must be normalized as the brightness of each star varies and this causes variance in the SAP_FLUX and PDCSAP_FLUX fields. A mean division normalization method is used where t is time and μ is the mean.

$$PDC_{normalized} = \frac{PDCSAP(t)}{\mu} \tag{1}$$

Finally, a time indicator field is needed. The time attribute that is contained within the .fits file may contain gaps. This again causes issues when analyzing data, to mitigate this a new time field is added for the entire dataset that increments by 1 for each instance.

2.2 Traditional Methods

Windowed Modified Z-Score (WMZ). The Z-score is a way of determining how many standard deviations a given value is from the mean of the dataset. Equation 2 is used where *PDCSAP(t)* is the given data instance, μ is the mean, and σ is the standard deviation.

$$Z_{score} = \frac{(PDCSAP(t) - \mu)}{\sigma} \tag{2}$$

Using the concept of the Z-score, the modified Z-score algorithm [6] reduces the effect of anomalies on μ and σ by replacing μ with a sample median and σ with the Median of Absolute Deviations (MAD).

$$Z_{modified\_score} = 0.6745 \times \frac{(PDCSAP(t) - \mu_{sample})}{MAD} \tag{3}$$

This method was modified using a windowed technique were the function is run only on a set range of the data. Once the entire set of data has been run through the function, anomalies from each window were compiled into a complete list. The two variables used to alter this algorithm were the window size and threshold (Z-score) used to determine if an instance was an anomaly.

Windowed General Extreme Studentized Deviate (WGESD). GESD is a variation on the Grubb's test that allows for more than one anomaly to be detected. There can be up to *N* number of anomalies in the data. Equation 4 is performed *N* times and the dataset shrinks by a size of one on each iteration as the value that maximized the function is removed. Once *N* number of values are found the critical value is used to determine which of the *X* values are anomalies [7].

$$X_{max} = \frac{MAX_i \left| (X_i - \mu_{sample}) \right|}{\sigma_{sample}} \tag{4}$$

In Equation 4, μ is the mean, and σ is the standard deviation. This method was also modified using a windowed technique were the function is run only on a smaller set of the data until the entire set has been run through the function. Once complete, anomalies from each window are compiled into a complete list. The two variables used to alter this algorithm were the number of anomalies being checked for (*N*) and the window size.

Percentile Rank (PERC). PERC is determined by finding the frequency distribution of the dataset and selecting the specified furthest extremes or least frequent values [8]. The percentile rank is found using Equation 5.

$$PERC_{rank} = \frac{PDC_{pos}}{N} \times 100\% \tag{5}$$

PDC_pos is the position of the score within the distribution and *N* is the total number of instances in the dataset. Two factors determine percentile rank anomalies: the number of instances in the dataset and the extreme values. The two variables used to alter this algorithm were the low and high extreme boundaries.

2.3 Neural Networks

Artificial Neural Network (ANN). ANNs, as the name suggests, are inspired by the network of neurons in the human brain. This paradigm seeks to mimic the way the

neurons in the human brain not only process information but also learn as new information is given. This is accomplished with the creation of a heavily interconnected network of neurons that together can solve problems through learning. The training of an ANN is perhaps the most important aspect. Information with known answers (ground truth) is fed through the ANN and the input weight and activation of each neuron is adjusted to move the network state towards being able to correctly identify all the training data answers [9].

Figure 3 (a) shows a diagram of a more standard neural network. In this example the value from each of the three inputs is fed to each neuron of the hidden layer. Through training the weights of each input and the threshold for the activation function will be adjusted. It will continue to learn and adjust if better results can be achieved. When no variation can achieve better results the network is considered trained.

Many aspects of a network can be modified such as activation function, training function, number of neurons in a hidden layer, how many hidden layers are used, how each layer connects, etc. This creates endless variations of ANNs, the key is to determine which configuration will have the best results for any given problem.

In this project two types of networks were used: Pattern Neural Networks (PNN) and Recurrent Neural Networks (RNN). PNNs used scaled conjugate gradient backpropagation and RNNs used Levenberg_Marguardt as the training function.

Recurrent Neural Networks. RNNs differ from standard neural networks by allowing the output of the hidden layer neurons to feedback and serve as the input to themselves or other neurons. This allows the network to use history as a way of understanding the sequential nature of the data [10]. We used a RNN where the output from the activation function was an additional input to the hidden layer, Figure 3 (b).

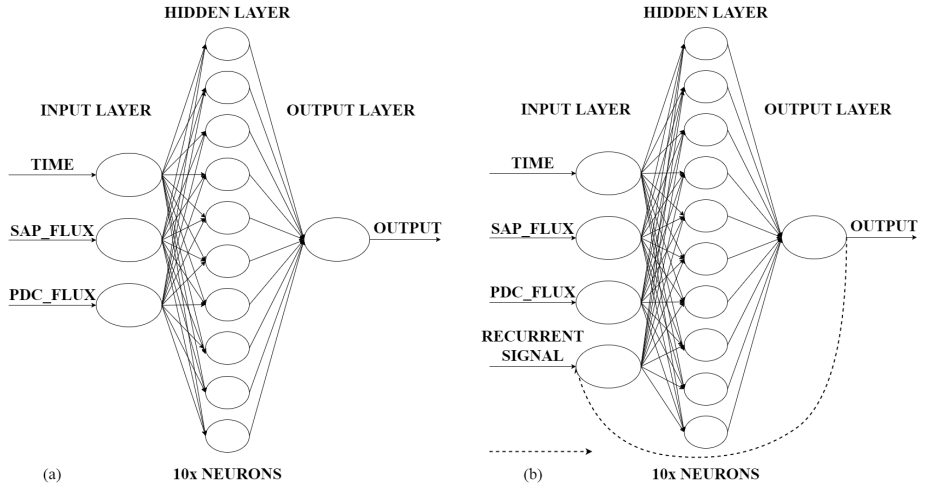

Fig. 3. (a) Shows a 10-neuron pattern neural network. (b) Shows a 10-neuron recurrent neural network where the output from the activation function is fed back in as a fourth input.

Data Replication. Unbalanced classes heavily influence the training of neural networks. In the short cadence files, there was an average of 30 anomalies per 50,000 instances. Resampling is one method to counter unbalanced classes, but due to the limited number of anomalies in the dataset we chose to use replication as a method to counter this data imbalance problem. Anomalies were first detected using a traditional algorithm and then replicated.

On horizontal data, the anomalies were multiplied a given number of times, randomly redistributed throughout the dataset, and then run through the detection algorithm a second time. Results are shown in Figure 4.

On curved data, random redistribution would not work. A method of detecting the anomalies using a traditional algorithm, moving both forward and backward within the time sequence a set number of steps, and then replacing the existing data with the anomaly data a given number of times was used.

Additional Factors. Having no ground truth labels for the anomalies in our data makes determining which network is most accurate more difficult. Two additional factors in our detection are which algorithm and parameters are used for the initial training data and the test data.

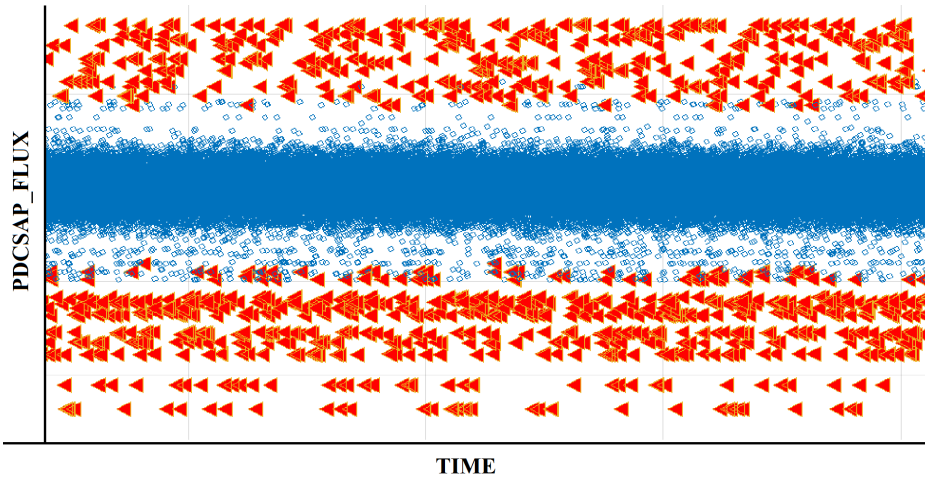

Fig. 4. Approximately 1,500 anomalies were created in the horizontal dataset to train neural networks. Anomalies are displayed as arrows.

Measurement Terminology. Precision - Positive Predictive Value (PPV) measures how many of the instances labeled as an anomaly are true anomalies when compared with traditional algorithm detection.

$$Precision = PPV = \frac{TP}{TP + FP} \tag{6}$$

For Equation 6, TP (true positive) is the number of anomalies correctly detected, FP (false positive) is the number incorrectly labeled as anomalies.

Sensitivity – True Positive Rate (TPR) measures how many of the anomalies from the traditional algorithms were correctly labeled.

$$Sensitivity = TPR = \frac{TP}{TP + FN} \tag{7}$$

For Equation 7, *TP* (true positive) is the number of anomalies correctly detected, *FN* (false negative) is the number of true anomalies incorrectly labeled.

3 Results

3.1 Traditional Algorithms

Traditional algorithms were run on short cadence light curves to detect anomalies within the dataset. Figure 5, Table 1, and Table 2 show results when traditional algorithms are run on horizontal and curved datasets. The horizontal dataset contained 106,839 instances and the curved dataset contained 60,376 instances.

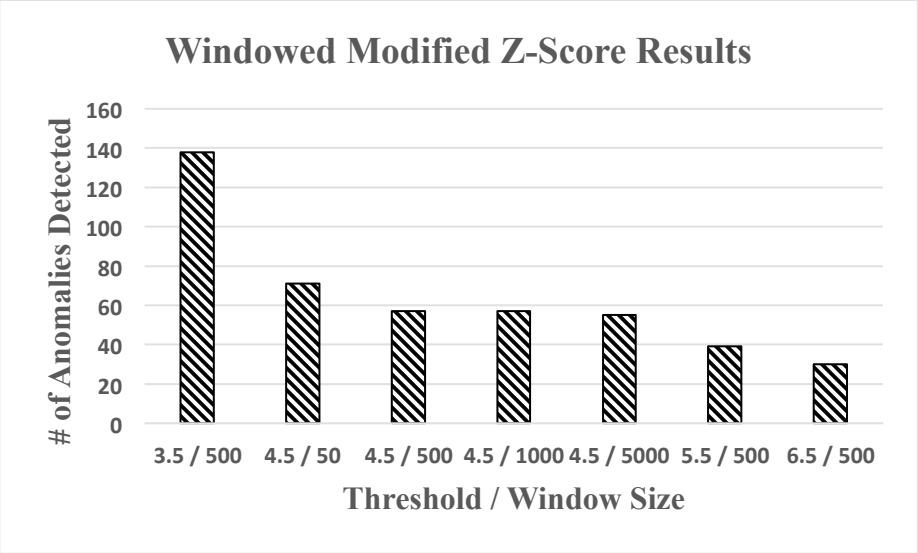

Fig. 5. Number of anomalies detected when both the threshold and window size are alterred using WMZ.

Performance. The window function allows the modified Z-score algorithm to execute on only a smaller portion of the data at a time and thus it is unaffected by the horizontal or curved shape of the datasets, Table 1 and 2. Window size has a small

effect on the number of anomalies that are detected, Figure 5. However, threshold has a more profound effect; the lower the value the more sensitive the algorithm.

Because of the same window function as seen with WMZ, the WGESD algorithm can operate on both horizontal and curved datasets. When using WGESD the number of anomalies being checked for has a small effect. Inversely from WMZ, the window size has a larger effect on the number of anomalies detected.

With the PERC method, the greater the number of instances in the dataset the more anomalies that will be detected. And the closer the extreme values are to the center the more anomalies that will be detected. PERC method requires tuning and knowledge of where anomalies are in the dataset to ensure their detection. This algorithm also performs poorly on curved data. It is only able to detect anomalies that are outside the mean range of the entire dataset, but anomalies that are within the mean range are unable to be detected.

Table 1. WGESD and PERC results are compared with WMZ (4.5, 500) detected anomalies on horizontal data. The format for parameters used is: WMZ – threshold / window size, WGESD - # of anomalies to check for / window size, and PERC – lower percentage boundary / higher percentage boundary.

Method	Time	# Detected	Precision	Sensitivity
WMZ - 4.5 / 500	1.55s	57	-	-
WGESD - 2 / 50	3.38s	162	34.57%	98.25%
WGESD - 2 / 500	0.34s	64	78.13%	87.72%
WGESD - 5 / 500	0.50s	72	79.17%	100.00%
WGESD – 10 / 500	0.73s	73	78.08%	100.00%
WGESD - 2 / 5000	0.14s	26	100.00%	45.61%
PERC - 0.03 / 99.97	0.008s	64	79.69%	89.47%

Table 2. WGESD and PERC results are compared with WMZ (4.0, 750) detected anomalies on curved data. The format for parameters used is: WMZ – threshold / window size, WGESD - # of anomalies to check for / window size, and PERC – lower percentage boundary / higher percentage boundary.

Method	Time	# Detected	Precision	Sensitivity
WMZ - 3.5 / 750	1.01s	55	-	-
WMZ - 4.0 / 750	0.94s	29	-	-
WMZ - 5.5 / 750	0.93s	17	-	-
WGESD - 2 / 500	0.20s	35	65.71%	79.31%
WGESD - 2 / 750	0.13s	27	85.19%	79.31%
WGESD - 2 / 1000	0.10s	17	94.12%	55.17%
PERC - 0.03 / 99.97	0.006s	36	11.11%	13.79%

Ground Truth Label. Both WMZ and WGESD have seemingly accurate detection of anomalies within the data. WMZ was chosen as the algorithm to use for ground truth labels as it was slightly less sensitive to data points that were on the edge of being normal points. From Table 1, this oversensitivity was why WGESD consistent-

ly had a higher number of anomalies detected. And in the curved data from Table 2 both sensitivity and precision were affected.

3.2 Neural Network Anomaly Detection

Table 3. Neural network result on horizontal dataset.

#	Type	# Neurons	Time	Comparison	Training Data	Precision	Sensitivity
1	**PNN**	10	0.042s	5000, 6.0	400, 4.5	38.75%	100.00%
2	**PNN**	10	0.030s	1000, 4.0	400, 4.5	21.94%	100.00%
3	**PNN**	10	0.031s	400, 3.5	400, 4.5	89.86%	43.97%
4	**PNN**	32	0.043s	750, 4.1	750, 4.1	91.94%	100.00%
5	**RNN 1:1**	10	0.062s	5000, 6.0	400, 4.5	45.59%	100.00%
6	**RNN 1:1**	10	0.063s	1000, 4.0	400, 4.5	25.85%	100.00%
7	**RNN 1:1**	10	0.054s	400, 3.5	400, 4.5	97.06%	46.81%
8	**RNN 1:3**	10	0.081s	750, 4.1	750, 4.1	91.80%	98.25%
9	**RNN 1:3**	32	0.267s	750, 4.1	750, 4.1	91.67%	96.49%

Parameters. In Table 3 and 4 these parameter formats are used. RNN X:Y where X indicates the lowest time delay and Y indicates the highest time delay. For example, 1:5 means that the input is delayed at 1, 2, 3, 4, and 5 time steps backwards.

All comparison and training data was processed using the WMZ algorithm. The first number indicates the window size and the second number indicates the threshold.

In horizontal replication used for Table 3, the anomalies were replicated 50 times, redistributed randomly in the dataset, and then labeled again using WMZ with a window size of 400 and threshold of 6.0. In Curved replication, the first number is how many times the data was replicated and the second number is how many time steps outward each iteration was replicated.

Number of neurons [X, Y] indicates that multiple hidden layers were used and how many neurons were in each layer.

Table 4. Neural network result on curved dataset.

#	Type	# Neurons	Comparison	Training Data	Replication	Precision	Sensitivity
1	**PNN**	10	500, 4.5	500, 4.5	30, 2	100.00%	19.35%
2	**PNN**	1000	500, 4.5	500, 4.5	30, 10	100.00%	19.35%
3	**PNN**	[10, 10]	500, 4.5	500, 4.5	30, 10	100.00%	19.35%
4	**RNN 1:2**	10	500, 4.5	500, 4.5	20, 2	100.00%	9.68%
5	**RNN 1:2**	10	500, 4.5	500, 4.5	40, 10	10.39%	25.81%
6	**RNN 1:2**	[10, 10]	500, 4.5	500, 4.5	30, 10	100.00%	25.81%
7	**RNN 1:3**	[10, 10]	500, 3.5	500, 3.5	10, 10	31.37%	51.61%
8	**RNN 1:3**	10	500, 4.0	500, 4.0	30, 10	100.00%	32.26%
9	**RNN 1:5**	10	500, 4.5	500, 4.5	30, 10	100.00%	6.45%

Performance. With the properly adjusted parameters, both the PNN and RNN achieved high precision and sensitivity when detecting anomalies in horizontal data. The PNN achieved both higher sensitivity and precision, but both operated well. Table 3 method 4 and method 8 demonstrate the highest performances achieved. Figure 6 shows a comparison between WMZ and RNN anomalies detected and shows that both detections are nearly identical.

PNNs maintained high precision, but no variation of parameters achieved a higher sensitivity than 19.35%. RNNs are more effective for curved datasets. The time-delay and replication parameters had the most effect on anomaly detection, Table 4. RNN 1:3, Table 4 method 8, was the best performance achieved, yet only 32.26% of the anomalies were detected. Figure 7 illustrates this is because the network was only able to detect anomalies that are underneath the mean range of the curved dataset.

3.3 System Specifications

All pre-processing and detection methods discussed in this paper were implemented using the following system specifications:

Memory: 32 GB
Processor: Intel Core i7-6700 CPU @ 3.40GHZ x 8
Graphics: GeForce GTX 745
OS: Ubuntu 16.04 LTS
MATLAB: 9.0.0.341360 (R2016a)

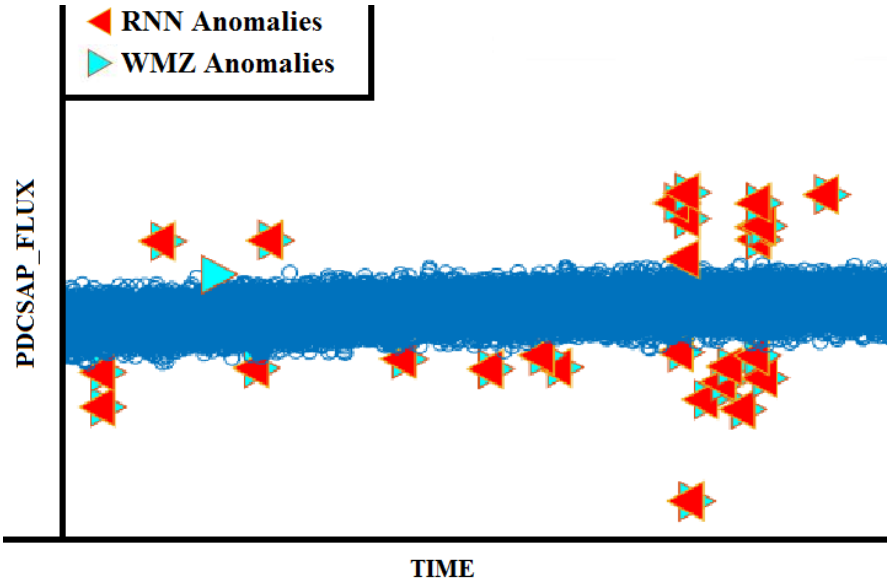

Fig. 6. Comparison of anomalies detected on horizontal data using Table 3 method #8.

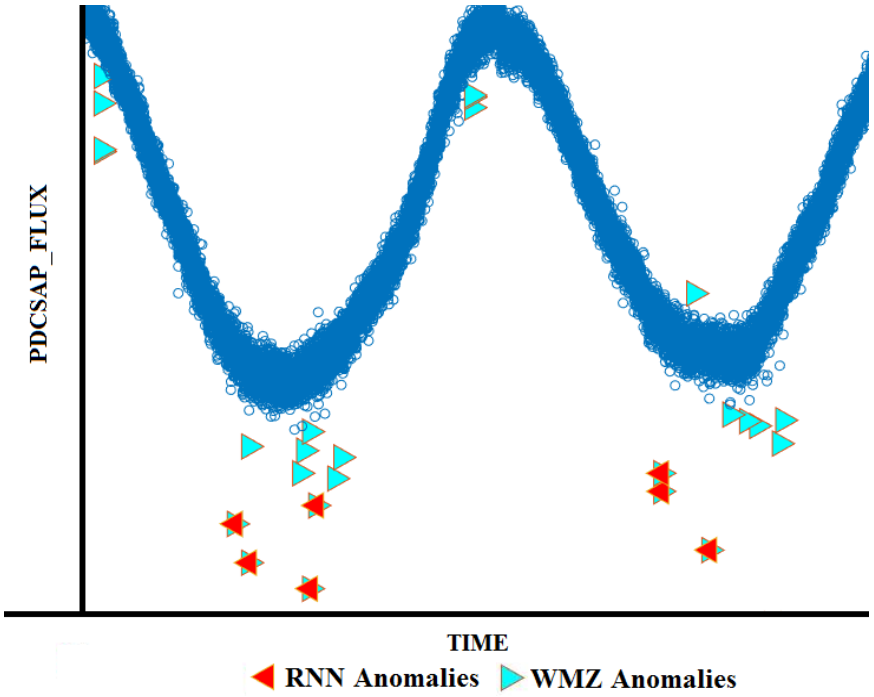

Fig. 7. Comparison of anomalies detected on curved data using Table 4 method #8.

4 Discussion

Initially, all traditional algorithms were only able to operate on horizontal data. Through the introduction of the windowed method both modified Z-score and general extreme studentized deviate achieved successful results on curved data. The PERC method is weak on curved data due to anomalies often being hidden within the mean of the entire dataset. The PERC algorithm is much faster than any other method and this is due to the small number of calculations necessary to identify anomalies. This advantage is also what makes this algorithm less versatile.

As discussed earlier there are many factors in training a neural network to detect anomalies on the Kepler data. Ground truth labels would eliminate both the comparison and training data parameters and leave only the replication parameters to be adjusted.

ANNs achieved similar performance on horizontal data and once trained they can execute up to 36 times faster than WMZ and up to 11 times faster than WGESD. Unfortunately, neither of these networks were effective in overall detection on curved data. The networks could not detect the anomalies between the trough and crest of the curve, Figure 7. Given the data and networks described in this paper, these networks are not well suited for anomaly detection on curved types of time-series data.

5 Conclusion and Future Work

In this paper, we show that robust anomaly detection from Kepler satellite time-series data was accomplished using various algorithms. The type of method and parameters can be adjusted to fit any type of horizontal or curved short cadence light curve file. For curved data performance WMZ and WGESD are the best choice. However, if the data is horizontally shaped a trained PNN or RNN offers the best performance.

Given the fact that neural networks are highly non-linear end-to-end "black boxes", it is possible that other types of ANN models may be able to detect the anomalies within the curve of the data. Further research on additional methods such as Long-Short Term Memory (LSTM) integrated neural networks is needed.

The prediction of KOIs from the anomalies detected fell outside of the scope of this paper. Initial tests showed that anomalies on short cadence light curves may not be an effective indicator to whether a star is harboring an exoplanet. More research and experiments are needed to find methods of using these anomalies as a predictor to whether a file contains a Kepler Object of Interest.

References

1. NASA: Kepler: About the Mission, `https://kepler.nasa.gov/Mission`
2. Leiner, E.: Other Worlds: Analyzing the Light Curves of Transiting Extrasolar Planets. Thesis, Wesleyan University (2010)
3. Mikulski Archive for Space Telescopes: Space Telescope Science Institute, `http://archive.stsci.edu/pub/kepler/lightcurves/tarfiles/`
4. Botros, A.: Artificial Intelligence on the Final Frontier: Using Machine Learning to Find New Earths. Technical report, Stanford University (2014)
5. Guo, H., Murphey, Y., Feldkamp, L.: Neural Learning from Unbalanced Data. In Applied Intelligence 21, pp. 117-128. Kluwer Academic Publishers (2004)
6. Rosner, B.: Percentage Points for a Generalized ESD Many-Outlier Procedure. Technometrics, 25(2), 175-172 (1983)
7. Iglewicz, B., Hoaglin, D.: Volume 16: How to Detect and Handle Outliers. The ASQC Basic References in Quality Control: Statistical Techniques, (1993)
8. Lane, D., Scott, D., Hebl, M., Guerra, R., Osherson, D., Zimmer, H.: Introduction to Statistics. Rice University, University of Houston, Tufts University, 29-33 (2007)
9. Aleksander, I., Morton, H.: An introduction to Neural Computing, Second Edition. Intl Thomson Computer Pr (T) (1995)
10. Nanduri, A., Sherry, L.: Anomaly Detection in Aircraft Data Using Recurrent Neural Networks. In: Integrated Communications Navigation and Surveillance (ICNS) Conference, pp. 5C2 1-8. IEEE Press (2016)