CS230

Deep Learning Approach to Finding Extra-solar Planets from *TESS* Light Curves

Ingrid M. Beerer
dirgni@stanford.edu

Category: *Other (Astronomy, time series, binary classification)*

Abstract

The *Transiting Exoplanet Survey Satellite (TESS)* has measured the brightness of hundreds of thousands of stars since 2018, in the hope of detecting exoplanets. A periodic dip in star brightness may indicate the existence of a planet transiting the star, but could also be explained by phenomena other than a planet. Currently, teams of experts visually inspect and classify observations as evidence of a planet or not. A handful of attempts have been made to automate the classification process using 1D CNNs with limited success. In this paper, I present a new model, an RNN with LSTM layers, for classifying planet and non-planet signatures in stellar brightness time series. The RNN model achieves an accuracy of 96% and an F1 score of 0.43 on the testing dataset, which is quite similar to results ahieved by the current state of the art CNN model for exoplanet classification.

1 Introduction

The *Transiting Exoplanet Survey Satellite (TESS)*, a NASA-sponsored, space-born telescope, just completed its 2-year science mission in mid-2020. The primary goal of *TESS* is to discover planets smaller than Neptune that orbit stars beyond our solar system. *TESS* monitored hundreds of thousands of stars in order to produce a time series of brightness, also known as a "light curve", for each star. Scientists study these light curves looking for periodic drops in brightness caused by planetary transits, events when a planet passes in front of its star.

TESS has already produced thousands of light curves and discovered many stars indeed have periodic dips in their brightness. A transiting planet, however, is not the only possible cause of periodic drops in stellar brightness. There are a few other astrophysical phenomena, such as eclipsing binary stars, that could result in a "false positive" transit signal. Typically, teams of experts manually inspect and classify light curves as planet candidates or not. The goal of this project is to produce a binary classification model that can differentiate true planetary transit signals from false positives. The input to the model is time series data of star brightness (light curves). The output is a classification of "planet" or "not planet".

2 Related work

There have been a handful of attempts to apply classical machine learning and deep learning algorithms to the field of extra-solar planets. *Kepler*, a NASA-launched space telescope, collected photometry data on thousands of stars between 2009 and 2018. Similar to *TESS*, its goal was to find transiting extrasolar planets. Armstrong et al. 2016 (3) developed a model using Self Organizing

,

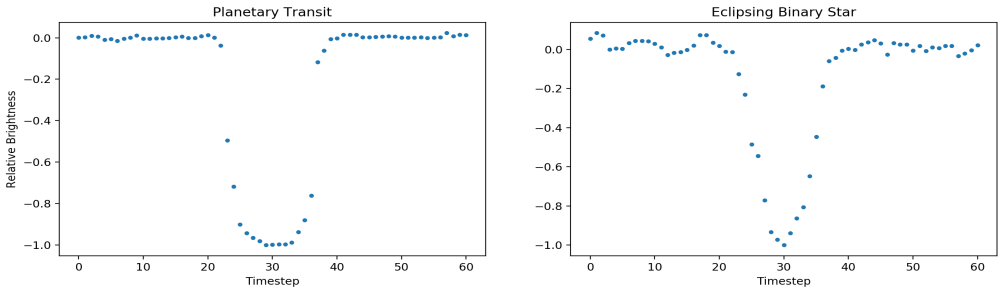

Figure 1: Example planet (left) and non-planet (right) lightcurves from the dataset.

Maps to distinguish planetary transits from false positives among *Kepler* light curves. Pearson et al. 2017 (5) and Shallue et al. 2018 (6) developed convolutional neural network (CNN) models to classify *Kepler* planets. The Shallue et al. model was named *AstroNet*.

Yu et al. 2019 (7) and Osborn et al. 2020 (4) adapted the AstroNet 1-D CNN model originally developed for *Kepler* data to *TESS*. Osborn et al. 2020 trained their model on simulated *TESS data*, however, and found that the model did not extend well to real *TESS* data. The recall accuracy of their model was only 61% when tested on real *TESS* data. Yu et al. 2019 (7) achieved of precision of 69%.

The AstroNet models employed by Yu et al. 2019 (7) and Osborn et al. 2020 (4) are 1-D CNN models. One of the drawbacks of CNNs is that the model only considers correlations between nearby points in the time series. My hypothesis is that we may be able to achieve better results with a recurrent neural network (RNN).

3 Dataset and Features

The input data are 1-D timeseries of star brightness. The *TESS* data processing pipeline provides light curves that have been corrected for instrument and other non-astrophysical effects. These data are publicly available on the *Mikulski Archive for Space Telescopes (MAST)* (2).

I adopted the dataset from Yu et al. 2019 (7). Yu et al. manually inspected and labeled about 16,000 light curves. They made their full dataset and classifications available on Github (1). The raw TESS light curves were binned and centered on potential planetary transit events. Example light curves for a known planet and a known eclipsing binary star (non-planet) are shown in Figure 1.

Since the time Yu et al. constructed their dataset, some of their examples have been classified using follow-up astronomical observations to confirm the existance of an exo-planet. I replaced the Yu et al. 2019 label with the confirmed classification where available.

I used 80%, 10%, and 10% of examples for the training, dev, and test sets, respectively. The breakdown of labeled examples is shown in Table 1.

Dataset	Planets	EBs	Other Non-planets	Total
Training	350 (2.7%)	1799 (14%)	11,049 (84%)	13,201
Dev	42 (2.5%)	239 (14%)	1,377 (83%)	1,658
Test	44 (2.7%)	183 (11%)	1,430 (86%)	1,657

Table 1: Composition of training, dev, and test sets.

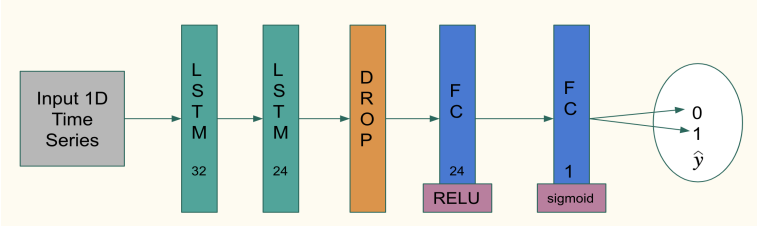

Figure 2: Model Architecture.

Epochs	Train Set F1	Dev Set F1
15	0.55	0.42
30	0.65	0.4
50	0.67	0.39
100	0.81	0.43
200	0.9	0.4

Table 2: Train and Dev set F1 score after number of epochs.

4 Methods

4.1 Baseline AstroNet CNN

As a baseline for comparison, I used the "AstroNet" CNN model developed by Yu et al. 2019 (7), which is available on GitHub (1). I trained their CNN model on the training set and was able to closely reproduce their reported precision on the dev set, as reported in Table 4.

4.2 RNN Model Architecture

My hypothesis is that an RNN model with LSTM layers can do a better job on binary classification of 1-D time series. The architecture of my RNN model is shown in Figure 2. I have a bi-directional LSTM layer with 32 nodes, a second bi-directional LSTM with 24 nodes, followed by a Dropout layer, a Dense fully-connected layer with 32 nodes and *Relu* activation, and a final Dense layer with one output and *sigmoid* activation for binary classification.

I chose bi-directional LSTM layers because it is valid to assume we have the full time series at the time of classification. I experimented with having one to three LSTM layers and with one to three fully connected Dense layers. My final architecture is the one that achieved the best result with the least complexity.

I ran a brute-force search to choose the number of nodes in the LSTM and first Dense layers and the dropout rate. A subset of the hyperparameters used and their results are shown in the Appendix.

4.2.1 Unbalanced Data

Planet light curves comprise only 2.6% of the dataset. Oversampling positive examples in the training set can help avoid the model learning too much detail about the negative examples. I wrote a method that constructs a training dataset where the percentage of positive examples is an input to the method. I used *numpy* to randomly sample from the positive and negative training examples. The portion of positive examples in the training set became a hyperparameter of my model. In addition to the original training set, which contains 2.6% positives, I trained the model on datasets consisting of 10, 15, 20, 25, and 50% positive examples. I ultimately found the best results using a 10% positive example training set, and chose that hyperparameter value for my final model architecture.

I also tried using class weights in my loss function, where the loss from positive examples is weighted heavier than negative examples in inverse proportion to the size of the training set it comprises. I found that using class weights in training improved the training performance on my model, but caused the dev set performance to decline significantly. I therefore abandoned class weights going forward.

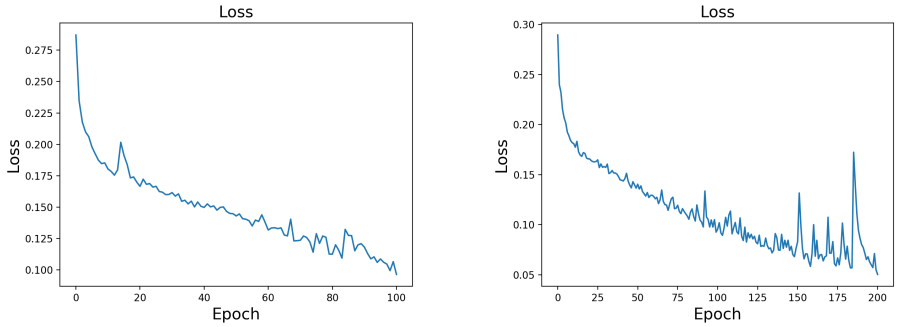

Figure 3: Loss after training for 100 epochs (*left*) and 200 epochs (*right*).

Dataset	Precision	Recall	F1	Accuracy
Train	0.77	0.85	0.81	0.96
Dev	0.37	0.51	0.43	0.96
Test	0.4	0.45	0.43	0.96

Table 3: RNN model performance on the train, dev, and test sets.

5 Experiments and Results

5.1 Training and Evaluation

I am solving a binary classification problem with a significant class imbalance. While accuracy is still important, I chose to optimize the F1 score of my model. The F1 score better captures how well the model performs on an unbalanced dataset.

I used the Binary Cross-entropy loss function and the ADAM optimizer for training. I used a batch size of 64. I experimented with batch sizes of 24, 32, 64, and 128. I saw little variance with these different batch sizes. I experimented with learning rates of 0.01, 0.001, and 0.0001 and found the model converged fasted with the default learning rate 0.001.

I trained the model using many combinations of hyperparameters over 30 epochs. I report a subset of the models trained in the Appendix. I found the models tended to result in an F1 score between 0.2 and 0.4 on the dev set. I chose the model with one of the highest dev F1 scores for further training. The final model architecture is shown in Figure 2.

I trained the model on a training set consisting of 10% positive examples with a learning rate of 0.001 for 15, 30, 50, 100, and 200 epochs. I plot the loss function from training over 100 epochs and 200 epochs in Figure 3. In Table 2, I report the train and dev set F1 scores after training for the numbers of epochs. I found that while the training set F1 score continues to improve with the number of epochs, that dev set F1 score stays relatively constant after just 15 epochs. These results may suggest that the model is overfit to the train set. However, we should also note that the composition of the dev and train sets differ. The dev and train sets contains 2.5% and 10% positive examples, respectively, which may impact the F1 score.

The shape of the loss curve is another matter for concern. The loss curve is relatively smooth up to 100 epochs. It becomes quite noisy after 150 epochs. I therefore chose to employ early stopping; I take the model trained for only 100 epochs as my final model. I briefly experimented with learning rate decay to solve the noisy loss late in training, but was unsuccessful at the reducing noise.

5.2 Discussion

Table 3 reports the precision, recall, F1 and accuracy of the final RNN model on the train, dev, and test datasets. Fortunately, the model performs just as well on the test set as the dev set, indicating the model is not overfit to the dev set. The accuracy is 96% on all datasets. The precision, recall, and F1 are better for the train set than the dev and test sets by a nearly a factor of 2. I partially attribute

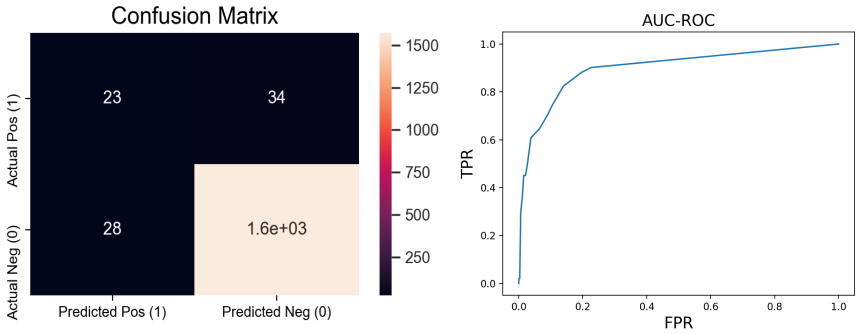

Figure 4: Confusion matrix (*left*) and AUC-ROC curve (*right*) constructed from the final RNN model's predictions on the test set.

Model	Precision	Recall	F1
AstroNet CNN	0.72	0.31	0.44
RNN	0.37	0.51	0.43

Table 4: Precison, recall, and F1 score for the baseline CNN and my best RNN model as measured on the dev dataset.

this to the difference is composition between the train and dev/test sets. I also speculate the model is somewhat overfit to the training set.

Figure 4 presents the confusion matrix and AUC-ROC curve derived from the model's predictions on the test set. The confusion matrix predictions are obtained using a classification threshold of 0.5. I constructed the AUC-ROC curve by calculating the True Positive Rate (TPR) and False Positive Rate (FPR) on the test set while varying the classification threshold from 0.0 to 1.0. The area under the ROC-curve is well above 0.5, indicating the model is able to effectively discriminate between the two classes of data.

In Table 4, I compare the dev set precision, recall, and F1 score for my best trained RNN model and the AstroNet CNN model which I trained on the same dataset. My RNN model achieves nearly identical F1 score compared to AstroNet.

5.3 Error Analysis

In order to better understand where the model fails, I inspected the most confident misclassifications. The most confident false positives are mostly eclipsing binary stars, which makes sense as that astrophysical phenomena results in light curves that most closely resemble a planetary transit. I also notice that most misclassified light curves are quite noisy.

6 Future Work

Given that the model has difficulty classifying very noisy light curves, filtering out noisy light curves from the train, dev, and test sets may improve the accuracy and F1 score of the model. *TESS* produces hundreds of thousands of light curves so it is reasonable to use an automated classification model on just those with high signal-to-noise ratio. The model also struggles most to differentiate between planets and eclipsing binary stars. Eclipsing binaries make up 12% of the train set. We could try oversampling them in the training set to help the model learn more detail about those examples.

If more time allowed, I would have explored adding attention to the LSTM units. Attention vectors allow the model to place greater weight on certain points in the timeseries. Given that the shape of the dip is the key distinguishing factor between planet and non-planet signals, attention may yield promising results.

7 Contributions

All work presented here, other than that explicitly referenced to published works, was done by me, Ingrid Beerer.

References

[1] AstroNet Vetting GitHub Repo. `https://github.com/yuliang419/AstroNet-Vetting`, note = Accessed: 2021-02-28.

[2] MAST TESS Archive. `https://archive.stsci.edu/tess/all_products.html`, note = Accessed: 2021-01-29.

[3] D. J. Armstrong, D. Pollacco, and A. Santerne. Transit shapes and self-organizing maps as a tool for ranking planetary candidates: application tokeplerandk2. *Monthly Notices of the Royal Astronomical Society*, 465(3):2634–2642, Nov 2016.

[4] H. P. Osborn, M. Ansdell, Y. Ioannou, M. Sasdelli, D. Angerhausen, D. Caldwell, J. M. Jenkins, C. Räissi, and J. C. Smith. Rapid classification of tess planet candidates with convolutional neural networks. *Astronomy & Astrophysics*, 633:A53, Jan 2020.

[5] K. A. Pearson, L. Palafox, and C. A. Griffith. Searching for exoplanets using artificial intelligence. *Monthly Notices of the Royal Astronomical Society*, 474(1):478–491, Oct 2017.

[6] C. J. Shallue and A. Vanderburg. Identifying exoplanets with deep learning: A five-planet resonant chain around kepler-80 and an eighth planet around kepler-90. *The Astronomical Journal*, 155(2):94, Jan 2018.

[7] L. Yu, A. Vanderburg, C. Huang, C. J. Shallue, I. J. M. Crossfield, B. S. Gaudi, T. Daylan, A. Dattilo, D. J. Armstrong, G. R. Ricker, and et al. Identifying exoplanets with deep learning. iii. automated triage and vetting of tess candidates. *The Astronomical Journal*, 158(1):25, Jun 2019.

8 Appendix

LTSM units 1	LTSM units 2	DropOut Rate	Dense Size	Train % Positive	F1 train	F1 test
24	24	0.1	24	10	0.57	0.41
24	24	0.1	32	10	0.59	0.42
24	24	0.3	24	10	0.54	0.41
24	24	0.3	32	10	0.61	0.38
24	24	0.5	24	10	0.59	0.43
24	24	0.5	32	10	0.56	0.39
32	24	0.1	24	10	0.58	0.38
32	24	0.1	32	10	0.1	0.07
32	24	0.3	24	10	0.56	0.42
32	24	0.3	32	10	0.43	0.33
32	24	0.5	24	10	0.6	0.39
32	24	0.5	32	10	0.58	0.44
32	32	0.1	24	10	0.59	0.42
32	32	0.1	32	10	0.66	0.41
32	32	0.3	24	10	0.59	0.36
32	32	0.3	32	10	0.58	0.42
32	32	0.5	24	10	0.61	0.39
32	32	0.5	32	10	0.37	0.25
24	24	0.1	24	25	0.78	0.32
24	24	0.1	32	25	0.76	0.31
24	24	0.3	24	25	0.76	0.28
24	24	0.3	32	25	0.79	0.32
24	24	0.5	24	25	0.76	0.31
24	24	0.5	32	25	0.77	0.34
32	24	0.1	24	25	0.79	0.33
32	24	0.1	32	25	0.77	0.3
32	24	0.3	24	25	0.77	0.26
32	24	0.3	32	25	0.74	0.32
32	24	0.5	24	25	0.76	0.34
32	24	0.5	32	25	0.77	0.31
32	32	0.1	24	25	0.73	0.35
24	24	0.3	24	15	0.66	0.39
24	24	0.5	24	15	0.69	0.33
32	24	0.3	24	15	0.67	0.4
32	24	0.5	24	15	0.68	0.36
24	24	0.3	24	20	0.72	0.35
24	24	0.5	24	20	0.72	0.35
32	24	0.3	24	20	0.73	0.34
32	24	0.5	24	20	0.57	0.33
24	24	0.3	24	50	0.89	0.27
24	24	0.5	24	50	0.88	0.25
32	24	0.3	24	50	0.88	0.22
32	24	0.5	24	50	0.87	0.21

Table 5: Hyperparameter search and results.