Using deep learning to detect exoplanets and categorize false-positives in Kepler threshold crossing events

Rebecca C. Roskill[1]

[1]Modeling the Universe, taught by Mordecai-Mark Mac Low and Lucy Lu, Department of Astronomy, Columbia University, New York, NY

1 Background

Kepler and the Transit Method NASA's Kepler Space Telescope was deployed from 2009-2018 to survey a portion of Earth's region of the Milky Way, in search of Earth-size exoplanets in or near habitable zones. To accomplish this, Kepler was equipped with a photometer that continually monitored the brightness of approximately 150,000 target stars. Using Kepler data, scientists can use the "transit method" to identify planets by looking for tiny dips in the brightness of a star when a planet crosses in front of it. Most commonly, the transit method involves three steps. First, images undergo a complex search process to identify transit-like signals called threshold-crossing events (TCEs), and a limb-darkened transit model is fit for each signal. Next, the TCEs are filtered by either programmatically defined rules or classifiers that make use of Machine Learning (ML) to identify those most likely to be exoplanet; for example, the DR 24 and DR 25 KOI TCEs were vetted using versions of Robovetter, by Coughlin et al. [1] and Thompson et al. [2]. Finally, these TCEs undergo manual before being released as Objects of Interest for future analysis [3]. Through this method, Kepler is estimated to have observed 3,246 planets to date, a number that continues to grow significantly.

Identifying Exoplanets with Deep Learning Contributing to the sustained discovery of planets, even years after Kepler's retirement, is the use of novel Deep Learning ML models, such as ExoMiner, which was credited for the discovery of 301 stars in 2021 [3]. ExoMiner is an advanced hybridization of, on one hand, non-Deep-Learning automatic methods, such as RoboVetter [1], which rely on computationally heavy engineered features (e.g. odd and even, weak secondary flux, centroid motion, difference image tests) and, on the other, earlier Deep-Learning methods, which rely only on phase-folded flux time series. Despite its impressive performance (AUC=1.000±0.000), reproducing ExoMiner was too daunting a project for my timeline, so I instead studied the earlier AstroNet [4].

AstroNet AstroNet achieved impressive performance for its time (AUC=0.993±0.003, on par with RoboVetter's AUC=0.994±0.003 [3]) and facilitated the discovery of two planets: one is a member of a five-planet resonant chain around Kepler-80; the other orbits Kepler-90, a star that was previously known to host seven transiting planets [4]. AstroNet takes two phase-folded flux time series as its only inputs: the first is a global view, depicting a fixed fraction of the TCE's period; the second is a local view, depicting a fixed fraction of the TCE's duration. The model behind AstroNet is an artificial neural network consisting of two convolutional branches -

one for each input array - whose outputs are concatenated and passed through a series of fully-connected layers. Whereas RoboVetter outputs 3 classes – planet candidate (PC), astrophysical false positive (AFP), and non-transiting phenomenon (NTP) – AstroNet combines the non-PC classes and performs binary classification. In my analysis, I attempt to extend AstroNet to perform multi-class classification that also distinguishes between astrophysical false positives and non-transiting phenomenon, or false positives resulting from instrumental error.

2 Methods

Data pre-processing AstroNet's published libraries were used to reproduce the preprocessing pipeline in [4]. From my end, this entailed downloading 15,737 TCEs from Kepler's Q1-17 DR24 [5] then running the input-generation script (astronet/data/generate_input_records.py) to produce 10 Tensorflow-compatible data shards: 8 were meant to be used for training shards, 1 for validation, and 1 for testing. In practice, I ended up reorganizing the training/validation/testing proportions in my training notebooks (FinalProject/TrainBinaryModel.ipynb and FinalProject/TrainMultiClassModel.ipynb) so that a random 75% of examples were set aside for training, 12% for validation, and 13% for testing. The pre-processing code is complex, so I will give a brief overview and point to their paper, [4], for more details. In short, the time series are first detrended for each quarter by iteratively removing 3σ outliers, fitting a cubic spline, and repeating until convergence (at which no data used to fit the spline were 3σ outliers). This process is shown in Figure 1. Then, all quarters were concatenated and the entire time series was phase folded. The data were shifted to a zero median and normalized such that the minimum value (the lowest point of the transit) equals -1. Finally, the time series were binned into arrays of lengths 2001 (global) and 201 (local), as the architecture requires uniform-length inputs. An example of the final result is visualized in Figure 2.

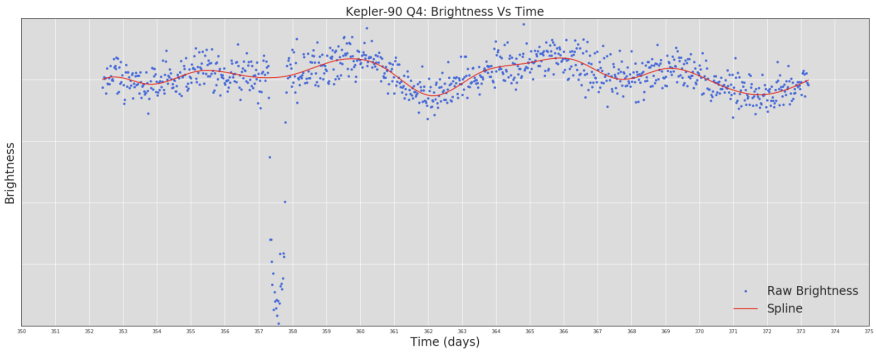


Figure 1: Example of a final spline fit to Kepler-90's 4th-quarter light curve, using iterative 3σ outlier removal to find the low-frequency trend. Figure courtesy of [4].

Binary classification of planet candidates The neural net's architecture (Figure 3) was closely based on the optimized AstroNet CNN model [4]. However, mini-batch (batch) normalization, a method used to standardize layer inputs for more stable training, was introduced to the convolutional branches and dropout, a method used to avoid over-fitting, was introduced to the dense (fully-connected) branch; the inclusion of these were tested as two hyperparameters. The final model was determined using random search of the hyperparameter space, including: batch normalization, included or excluded; dropout rate, selected from 3% intervals from 0-20%;

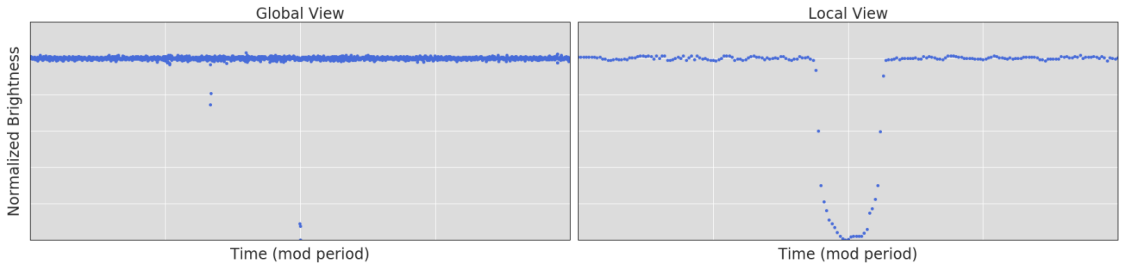


Figure 2: Example of fully processed global (left) and local (right) view inputs for Kepler-90. Figure courtesy of [4].

and the Adam optimizer's learning rate (α), selected from 2e-5, 8e-5, and 2e-4, and ϵ value, selected from 1e-8, 5e-8, and 1e-7. The addition of batch normalization and dropout (3%) proved marginally beneficial, and (2e-4, 5e-08) were selected as (α, ϵ).

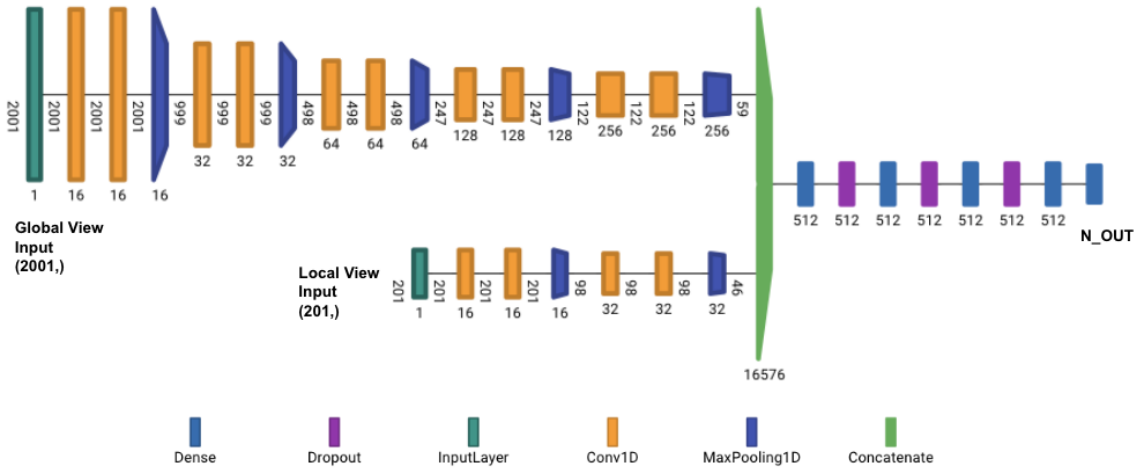
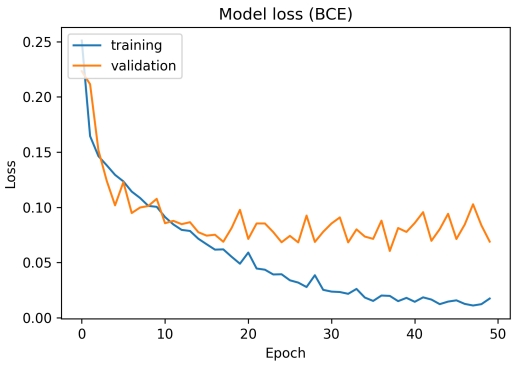


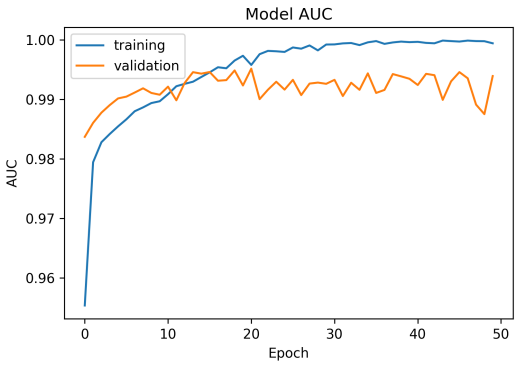
Figure 3: Overview of model architecture used for both binary and multi-class models. Numbers should be interpreted as follows: those between layers indicate input sizes; those beneath convolutional layers indicate the number of filters; those beneath max-pooling layers indicate the pool size; those beneath dense (fully-connected) layers indicate the number of units.

Before training, the input labels were converted to a single binary input (1 for PC, 0 for AFP or NTP). Because the dataset was heavily imbalanced (only $\approx 23\%$ PC), random oversampling was used to achieve a 50%/50% balance in the training set. The model was trained using a batch size of 64, which was found to be optimal for AstroNet [4], for 50 epochs, however the final weights were selected from epoch 18, where performance converged for the validation set (Figure 4).

Categorical classification of planet candidates and false positives The model architecture and training process for the multi-class model were almost entirely the same as those for the binary model. However, here, the input labels were converted to three one-hot-encoded features, which corresponded to AFP, NTP, and PC. The optimal hyperparameters found in this case were quite similar: batch normalization and 3% dropout were selected, along with (2e-5, 1e-08) for (α, ϵ). The model was trained for 100 epochs, and the final weights were chosen from the checkpoint at which performance on the validation set converged, near epoch 40 (Figure 5).

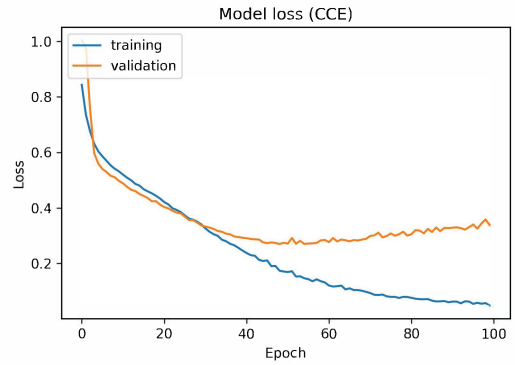


(A) Binary cross entropy (BCE) losses.

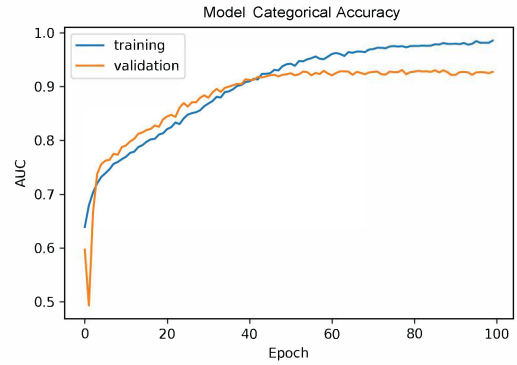


(B) Area under the ROC (AUC) scores.

Figure 4: Binary classifier performance on training and validation sets during training process.



(A) Categorical cross entropy (CCE) losses.



(B) Categorical accuracy scores.

Figure 5: Multi-class classifier performance on training and validation sets during training process.

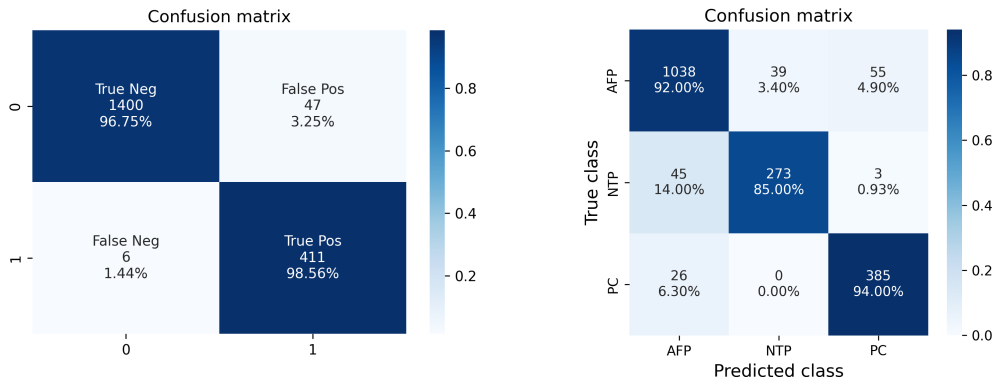
3 Results

Binary classification of planet candidates The binary classifier achieved impressive performance on reserved testing data, on par with the results of [4]. Table 1 shows several important metrics for binary classifier performance: binary accuracy, the percentage of correctly labeled samples; precision, correctly identified positive samples as a fraction of all inputs classified as positive; recall, correctly identified positive samples as a fraction of all actually positive samples; and the area under the ROC curve (AUC), which can be interpreted as the probability that the model will correctly distinguish between a positive and a negative sample. We see impressively high accuracy (97.2%), recall (98.6%), and AUC (99.5%). While precision is low, this is perhaps the most favorable outcome, since the pipeline expects further vetting to be required after TCE classification.

	Loss	**Accuracy**	**Precision**	**Recall**	**AUC**
Binary classifier	0.081	0.972	0.897	0.986	0.995

Table 1: Performance metrics for best binary classifier trained on reserved testing data: binary cross entropy loss (Loss); binary accuracy (Accuracy); precision; recall; and area under the ROC curve (AUC).

Categorical classification of planet candidates and false positives When the classifier was extended to distinguish between types of false positives as well as PCs, the same



(A) Binary classifier

(B) Multi-class classifier

Figure 6: Confusion matrices demonstrating performance of best classifiers trained on reserved testing data.

architecture maintained impressive performance. Table 2 shows fairly high accuracy (91.0%) and much higher accuracy if we consider whether the label was in the top-2-likelihood predictions (99.4%). This could reflect that the model can successfully distinguish between false positives and PCs (as in the binary case), but less accurately distinguishes between NTPs and AFPs. The confusion matrix shown in 6 confirms that this is indeed the case: the model achieves 94% accuracy on the PC class, 92% accuracy on the AFP class, and only 85% accuracy on the NTP class, with a significant portion of NTPs (14%) being misclassified as AFPs. Another possible explanation for this dip in performance on the NTP class could be the relative lack of NTP examples (2541, as opposed to the 9596 AFPs and 3600 PCs). The results most likely reflect a combination of the comparative similarity between false positive classes and data imbalance. To address this, one could try over- and/or under-sampling training data to better balance the classes, as I did for the binary classifier.

	Loss	**Accuracy**	**Top-2 accuracy**
Multi-class classifier	0.293	0.910	0.994

Table 2: Performance metrics for best multi-class classifier trained on reserved testing data: categorical cross entropy loss (Loss); categorical accuracy (Accuracy); top-2 categorical accuracy (Top-2 accuracy).

Interpreting the binary classifier's performance Looking further into the classifiers' failure cases can sometimes tell us more about why the model is misclassifying some samples, which can give insight into how to improve the model or another aspect of the training pipeline. In this case, however, no obvious conclusions could be drawn from visually comparing the false negatives in Figure 7 against the true positives predicted by the binary classifier 8. On one hand, the false negatives with the lowest prediction probabilities seem to be those with the lowest noise-to-signal ratios (an unexpected result since these are probably most easily recognized as transits by human eyes). On the other hand, there are also true positives with quite low noise-to-signal predicted with very high-confidence. While false positive examples are not visualized here, both [4] and [3] perform look deeply into the source of these misclassifications. Shallue et al. find that, in general, TCEs whose signals originate on stars other than the intended target (background planets) make up a majority of the false positives predicted by their model [4].

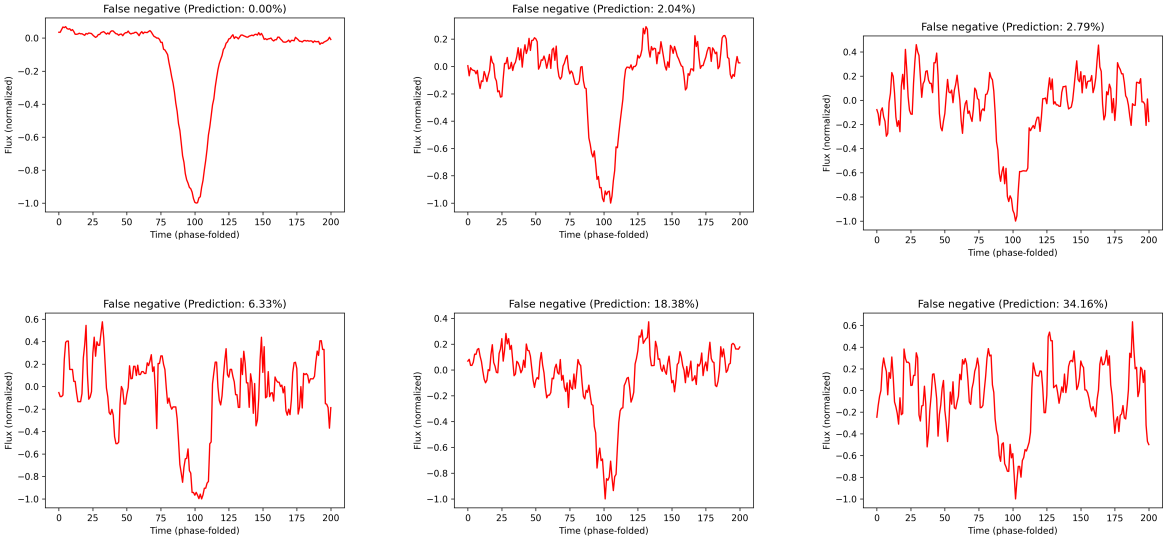


Figure 7: Example false-negative test cases for binary classifier, ordered by ascending prediction probability.

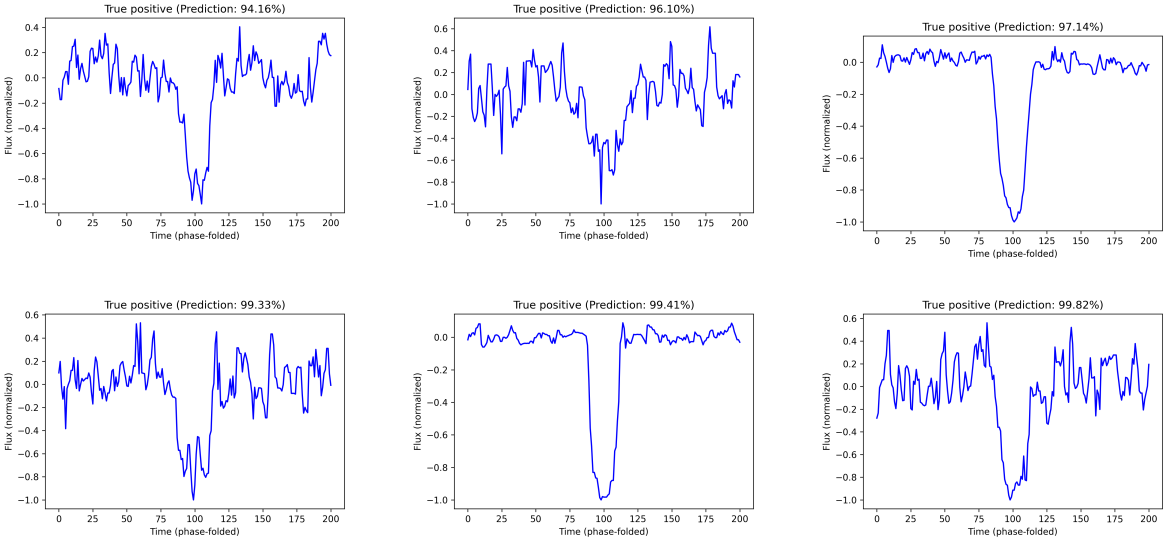


Figure 8: Example true-positive test cases for binary classifier, ordered by ascending prediction probability.

4 Lessons

This project was a great opportunity for me to gain more experience with deep learning, with which I had only a little bit of prior experience, while diving deeper into the Kepler light curve data that we started exploring in problem sets 3 and 7. I have surely gained a much better understanding of how Objects of Interest are flagged, labeled, and vetted. As someone interested in applying deep learning to fields that require domain knowledge to design good models (my other work has been in climate science and social science), it has also been very instructive to try an approach that combines some domain-based data engineering (e.g. removing quarterly low-frequency trends with a cubic spline, accounting for both global and local views separately, etc.) with a black-box neural net model. On one hand, it makes me wonder whether any of the pre-processing steps could have been learned (or even improved upon) by these models. On the other hand, though, the shortcomings of black-box models like these becomes clear when we are trying to diagnose what went wrong (e.g., why could the binary classifier recognize one PC but not another that looks extremely similar?). Ultimately, it makes me appreciate even more the diversity of numerical methods and the level of interpretability that the ExoMiner creators managed to synthesize in their analysis [3].

References

[1] J. L. Coughlin, F. Mullally, S. E. Thompson, et al. "Planetary Candidates Observed by Kepler. VII. The First Fully Uniform Catalog Based on the Entire 48-month Data Set (Q1-Q17 DR24)". In: *The Astrophysical Journal Supplement Series* 224.1 (May 2016), p. 12. DOI: 10.3847/0067-0049/224/1/12.

[2] S. E. Thompson, J. L. Coughlin, K. Hoffman, et al. "Planetary Candidates Observed by Kepler VIII. A Fully Automated Catalog with Measured Completeness and Reliability Based on Data Release 25". In: *The Astrophysical Journal Supplement Series* 235.2 (Apr. 2018), p. 38. DOI: 10.3847/1538-4365/aab4f9.

[3] H. Valizadegan, M. J. S. Martinho, L. S. Wilkens, et al. "ExoMiner: A Highly Accurate and Explainable Deep Learning Classifier That Validates 301 New Exoplanets". In: *The Astrophysical Journal* 926.2 (Feb. 2022), p. 120. DOI: 10.3847/1538-4357/ac4399.

[4] 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". In: *The Astronomical Journal* 155.2 (Jan. 2018), p. 94. DOI: 10.3847/1538-3881/aa9e09.

[5] S. Seader, J. M. Jenkins, P. Tenenbaum, et al. "Detection of Potential Transit Signals in 17 Quarters of Kepler Mission Data". In: 217.1, 18 (Mar. 2015), p. 18. DOI: 10.1088/0067-0049/217/1/18. arXiv: 1501.03586 [astro-ph.EP].