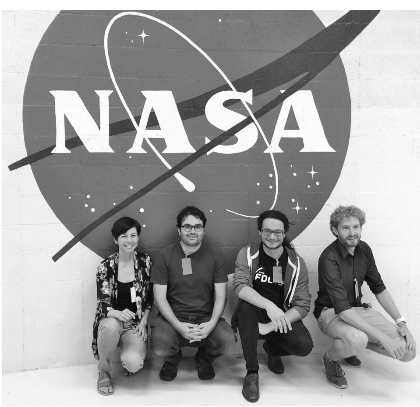
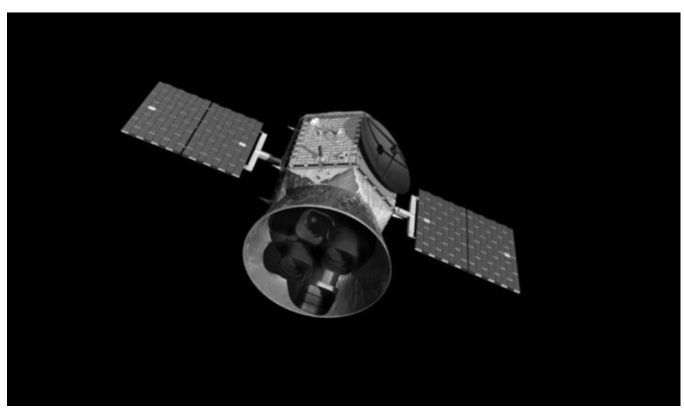
NASA Frontier Development Lab Technical Memorandum

Challenge Title: Rapid Classification of Exoplanet Transits with Deep Learning

Authors: Megan Ansdell (University of California Berkeley, United States), Yani Ioannou (University of Cambridge, United Kingdom), Hugh Osborn (Lab of Astrophysics of Marseille, France), Michele Sasdelli (University of Adelaide, Australia)

Mentors: Jeff Smith (SETI Institute, United States), Douglas Caldwell (SETI Institute, United States), Chedy Raissi (Institut National de Recherche en Informatique et en Automatique, France), Jon Jenkins (NASA Ames, United States), Massimo Mascaro (Google), Daniel Angerhausen (University of Bern), Adam Lesnikowski (Nvidia), Hamed Valizadegan (NASA Ames, United States)





August/2018

Private Partners: Google Cloud, Kx Systems

TITLE

Megan Ansdell (University of California Berkeley, United States), Hugh Osborn (Lab of Astrophysics of Marseille, France), Michele Sasdelli (University of Adelaide, Australia), Yani Ioannou (University of Cambridge, United Kingdom), Jeff Smith (SETI Institute, United States), Douglas Caldwell (SETI Institute, United States), Chedy Raissi (Institut National de Recherche en Informatique et en Automatique, France), Jon Jenkins (NASA Ames, United States), Massimo Mascaro (Google), Daniel Angerhausen (University of Bern), Adam Lesnikowski (Nvidia), Hamed Valizadegan (NASA Ames, United States)

NASA Frontier Development Lab

NASA Frontier Development Lab is a public / private research partnership between NASA, the SETI Institute and leaders in commercial AI and private space.

Hosted in Silicon Valley by the SETI Institute, the NASA FDL is an applied artificial intelligence research accelerator developed in partnership with NASA's Ames Research Center. Founded in 2016, the NASA FDL aims to apply AI technologies to challenges in space exploration by pairing machine learning expertise with space science and exploration researchers from academia and industry. These interdisciplinary teams address tightly defined problems and the format encourages rapid iteration and prototyping to create outputs with meaningful application to the space program and humanity.

SETI Institute
189 N. Bernardo Ave Suite 200
Mountain View, CA 94043

NASA Ames Space Portal
556 Edquiba Rd
Mountain View, CA 94043

Private Partners: Google Cloud, Kx Systems

INTRODUCTION

Problem: Exoplanet science is no longer data limited. Space missions such as *Kepler* (Borucki et al. 2010) produce copious amounts of data that need to be processed efficiently and systematically in order to yield reliable statistics on exoplanet occurrence rates. The recently commissioned Transiting Exoplanet Survey Satellite (*TESS*; Ricker et al. 2014), launched in April 2018, will amplify this data influx problem: every 27 days, *TESS* will observe a sector of the sky containing 16,000 stellar target stars at 2-minute cadence, producing 6 TB of data that need to be quickly and reliably searched for exoplanets candidates for ground-based follow-up observations during the short observing season following the download of each TESS sector.

Background: *Kepler* and *TESS* works by measuring the brightness of target stars as a function of time, producing a flux time series known as a light curve; exoplanets are identified when they transit in front of the star, causing a drop in the observed brightness. However, exoplanet signals are small compared to the instrumental noise/systematics as well as the inherent stellar variation also present in the data. Additionally, false-positive planet signals due to a variety of mechanisms (e.g., eclipsing binaries, background eclipsing binaries, instrumental noise) also need to be reliably culled. Traditionally this problem is approached using complicated pipelines that remove the instrumental noise/systematics and stellar variability, resulting in highly processed light curves that should presumably only include planetary signals. Planet candidates are then manually vetted by human operators or run through automated pipelines based on previous human judgements. This can result in biased or incomplete samples that are not ideal for constraining population-level planet occurrence rates, especially for rocky planets that produce the smallest signals. Machine learning offers a new approach to this problem that is capable of automatically and rapidly identifying nuanced exoplanet signals in noisy data.

Objective & Scope: Our challenge was to increase the efficacy and yield of exoplanet transit searches from *TESS* using machine learning techniques. To address this challenge, we were provided with Google Cloud compute resources (machines with up to eight NVIDIA Tesla K80, V100, or P100 GPUs) and storage space capable of holding many TBs of data. For training the machine learning models, we had real data from *Kepler* and simulated data for *TESS*, which was still undergoing commissioning at the time of this project. Thus the scope of the project was to develop/improve machine learning models using the *Kepler* dataset and then perform initial application to the *TESS* simulated dataset in preparation for real data from *TESS* in the future.

Significance: Developing a machine learning algorithm to automatically classify (and one day detect) transiting exoplanets will be extremely important for maximizing the scientific return from *TESS*, which will produce large influxes of data each month that need to be quickly and uniformly vetted for promising exoplanet candidates that can be followed-up with ground-based facilities. Humans will struggle to keep up with this influx of data, but machine learning can be used to match the pace and therefore maximize the scientific return from *TESS*.

MAIN TEXT

Identified Need

The *Kepler* and *TESS* Science Processing Pipelines detect transit-like signals, some of which are exoplanet candidate signals, but many of which are false positives (e.g., eclipsing binaries or instrumental effects). These transit-like signals, called “threshold crossing events” (TCEs), therefore require vetting before they are confirmed as planet candidates or determined to be false positives. *TESS* vetting must be performed very rapidly and uniformly due to the short timeframe between the public release of the data (which occurs monthly) and the end of the ground-based observing window.

Machine learning is ideal for this problem as it is capable of automatically as well as rapidly identifying nuanced exoplanet signals in noisy data. The state-of-the-art machine learning model for classifying exoplanet transit signals is from Shallue & Vanderburg (2017), who developed a deep convolutional neural network for binary classification of the *Kepler* TCEs. Here we improve upon this model and test its application to *TESS* data, while also exploring new and innovative machine learning approaches.

Data Description

For the *Kepler* data, we used the Q1-Q17 Data Release 24 (DR24) light curves. These light curves consist of a time-series of integrated flux measurements extracted via aperture photometry from Target Pixel Files (TPFs) at 30-min intervals spanning up to 4 years per target. Each light curve has one or more TCEs associated with it, all of which were identified by the *Kepler* Science Processing Pipeline. We used all ~20,000 TCEs in DR24, which are available from the Mikulski Archive for Space Telescopes (MAST). This amounted to ~90 GB of data, despite being only a small subset of the entire *Kepler* dataset. The *Kepler* labels come from the DR24 TCE table (a CSV file) available from the NASA Exoplanet Archive; specifically, they are the “av_training_set” column in the DR24 TCE table, which corresponds to the human-vetted classes assembled from multiple papers published over several years (e.g., Batalha et al. 2013). The “av_training_set” column has four possible values---planet candidate (PC), astrophysical false positive (AFP), non-transiting phenomenon (NTP), and unknown (UNK). However, the UNK TCEs were ignored and the labels were binarized as “planet" (PC; 3,600 entries) and “not planet" (AFP + NTP; 12,137 entries). The data were then randomly divided into training (80\%), validation (10\%), and test (10\%) sets. We used the training set to train the model, the validation set to choose model hyperparameters, and the test set to evaluate the final model performance. The Kepler dataset is well vetted but the labels are noisy (e.g., it is difficult to say for sure that there is no planet in a given light curve given the lack of ground truth).

For *TESS*, we utilized Lilith simulated data (Jenkins et al. 2018), version TSOP-301. Lilith produces end-to-end simulated data, meaning that it includes instrument/optics behavior as well as stellar astrophysics. The exoplanets, eclipsing binaries, and background eclipsing binaries are all injected at the pixel level; because the signals are injected, the labels are “perfect” although the simulated data is less complicated than real data and may also include noise effects that are not in the real *TESS* data (since the simulated data was produced prior to launch and commissioning of the spacecraft). The TSOP-301 version covers four *TESS* sectors (1–4) each of which include ~16,000 targets (some targets appear in multiple sectors due to overlap of the sector field of view on the sky). The simulated targets were based on the current *TESS* Target Input Catalog (TIC), making it the most realistic target selection for Lilith simulated data to date. However, it greatly over-populated detectable planets, meaning it has an unrealistic exoplanet distribution that does not reflect nature, which is not ideal for machine learning.

Following Shallue & Vanderburg (2017), we performed additional processing of the light curves before inputting them into the machine learning model. First, we “flattened" the light curve by iteratively fitting a basis spline to the light curve (with the in-transit points excluded), then divided the light curve by the best-fit spline linearly interpolated over the transit points. We implemented a different spline fitting routine than used in Shallue & Vanderburg (2017), which reduced data processing times by a few factors. Specifically, we used the LSQUnivariateSpline interpolation in SciPy rather than bsplin in PyDL (PyDL is a library of Python replacements for IDL built-in functions). Second, we created “global" and “local" views of the transit by folding the light curve on the TCE period following the description in Shallue & Vanderburg (2017). In short, both views are scaled so that the continuum is at zero and the transit depth is at -1; the global view encapsulates the full view of the folded light curve (e.g., including secondary transits of eclipsing binaries) at the cost of long-period TCEs having poorly sampled transits. Thus the local view, which depends on the TCE duration, provides a more detailed view of the transit shape. This is illustrated in Figure 1.

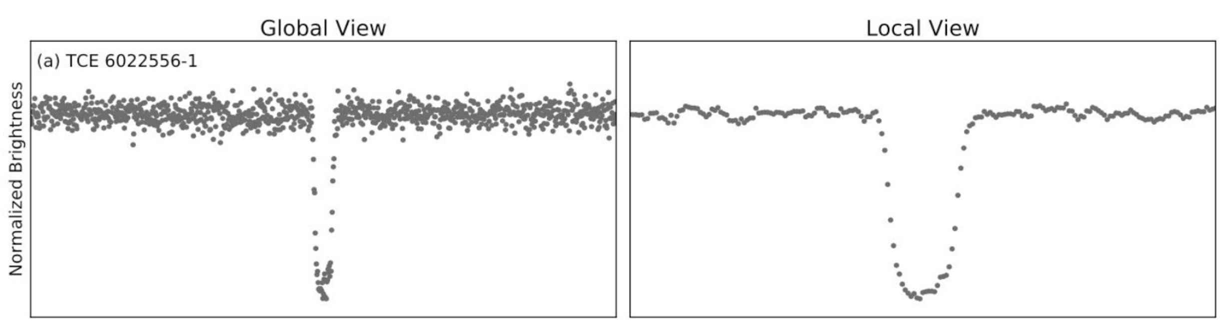


Figure 1: An example “global” and “local” views of a light curve folded on its TCE period. These light curve views were the inputs into the machine learning models.

Methodology

We took a multi-pronged approach to this challenge. Since Shallue & Vanderburg (2017) already developed a deep convolutional neural network to classify exoplanet transits, called *Astronet*, we started with their model and added more domain knowledge then also significantly reduced the model architecture size. We simultaneously pursued two much more experimental approaches that were high-risk-high-reward. These are all described in turn below.

Astronet: This baseline model from Shallue & Vanderburg (2017) was implemented in TensorFlow with a model architecture (see Figure 2) that has two one-dimensional convolutional neural networks (for the global and local views) with max pooling, the results of which are concatenated and then fed into a series of fully connected layers ending in a sigmoid function that produces an output in the range (0,1) that loosely represents the probability of a positive class (i.e., a true exoplanet transit). *Astronet* used the Adam optimization algorithm to minimize the cross-entropy error function over the training set. The training data were augmented by applying random time reflections to the light curves during training. The Google-Vizier system was leveraged to automatically tune the model and input hyperparameters. The model was therefore trained with a batch size of 64 for 50 epochs; the Adam optimization algorithm was implemented with alpha=1e-5, beta_1=0.9, beta_2=0.999, and eta=10e-8.

Augmented Astronet: This is our version of *Astronet*, re-written in PyTorch to be more accessible to astronomers. It also integrated new scientific “domain knowledge” to improve performance. First, we added a second channel to both neural networks in order to input the respective centroid time series for the global and local views. These centroid time series are the location of the target brightness center at each flux measurement; we processed them in the same way as the light curves (see above). Centroid information is important for identifying background eclipsing binary false positives that are offset in location from the main target star but within the same aperture. Second, we added stellar information when concatenating the results of the convolutional layers; stellar information can help to identify false-positives, for example related to giant stars with inflated stellar radii. Third, we included three more data augmentation techniques; in addition to the flipping implemented in the original *Astronet*, we added random Gaussian noise as well as randomly “jittered” the location of the transit center and transit bottom to simulate errors on the flux data points (which otherwise are not considered). We retained the use of the Adam optimization algorithm and cross-entropy loss function. The model was trained with a batch size of 64 for 250 epochs using the same Adam optimization parameters as before.

Smaller Astronet: This includes everything in augmented *Astronet* but with a drastically reduced model size (see Figure 2). We removed most of the layers in the convolutional neural networks, introduced global max pooling, and removed the fully connected layers. The model was trained with a batch size of 64 for 250 epochs; the Adam optimization algorithm was implemented with alpha=5e-4, beta_1=0.9, beta_2=0.999, and eta=10e-8.

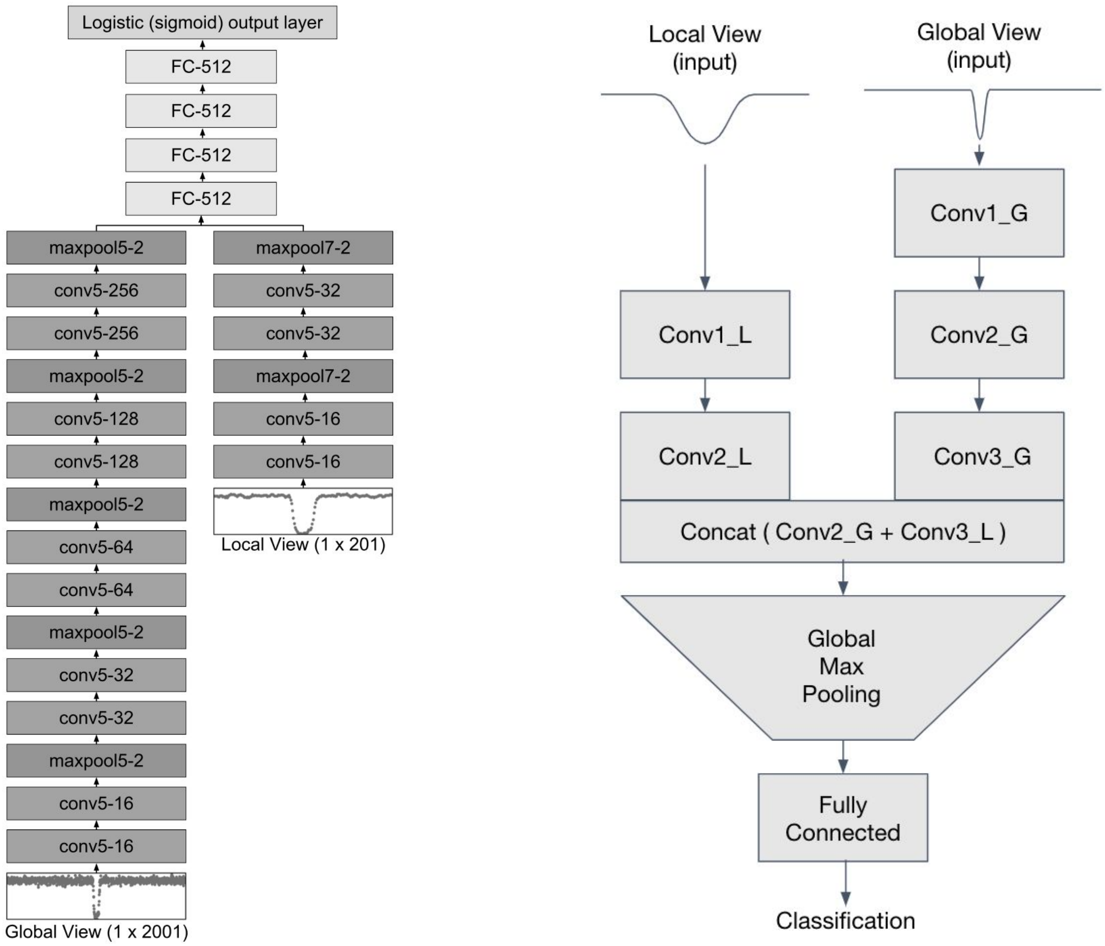


Figure 2: The original *Astronet* architecture (left) compared to our reduced model (right).

Target Pixel Files: This high-risk-high-reward approach aims to replace the Photometric Analysis (PA) and Transit Planet Search (TPS) sections of the *Kepler* and *TESS* Science Processing Pipelines by utilizing the calibrated target pixel files (TPFs), which are "postage stamp" time series of images (i.e. videos), rather than the highly processed light curves used in *Astronet*. To learn to classify periodic signals, we created a novel architecture to learn a set of time-domain basis filters, somewhat like the wavelet basis used in the TPS pipeline. The first few layers of the network were typical convolutional layers, reducing the spatial domain of the images to a singleton spatial dimension with successive convolution/pooling layers. This is designed to learn something similar to the foreground/background separation using the "optimal aperture" in the TPS pipeline, learning to separate the flux of the star from the background. The output of these spatial layers is a 1x1xT featuremap, where T is the number of time steps. This featuremap is fed into layers that learn 1D temporal filters. Each basis filter had roughly double the previous filter's width (e.g. basis filters were of length 1, 3, 7, …), much like a wavelet basis.

On the next layer a set of singleton filters (1xC, where C is the number of channels) is learned, effectively learning a linear combination of the basis filters.
Period-Phase Images: With this approach, we processed the *Kepler* light curves as period vs. phase "images," as shown in Figure 3. This is similar to the method originally presented in Jenkins et al. (1996) to visualize "sparse" periodic signals. The data is transformed in a representation designed to facilitate detection of the planet signals in the processed light curves. The light curves are "whitened" by removing the mean and normalizing by the variance of the signal. We fold the time series by all physically possible periods, sampled in log space. The folded time signals are then re-sampled in units of period. This provides a 2D representation (t/p, ln(p)) of the original 1D signal. Now planets look like sharp dots on a flat background. Other forms of stellar variabilities look like wider fluctuations. To test the usefulness of this data representation, we cropped the images around the ground truths from the DR24 dataset. We trained simple 2D convolutional neural network for classification. We tried a small five-layer network with random initializations as well as a heavier ResNet50 architecture pre-trained on ImageNet. If also detection in this data representation is proven possible, the approach would allow us to skip large part of the *Kepler* and *TESS* pipelines.

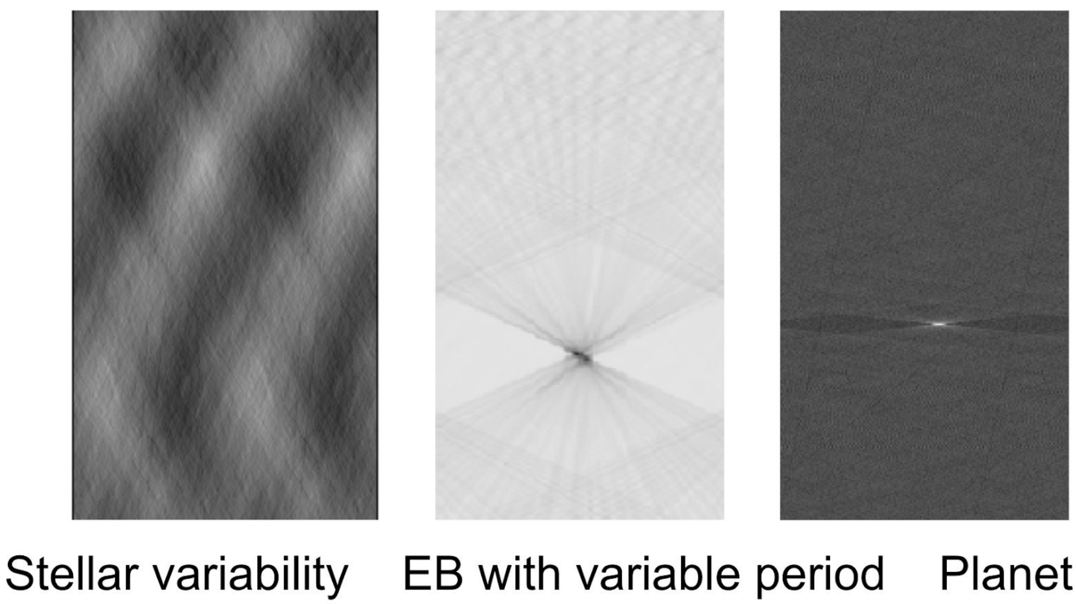


Figure 3: the period-phase "images" of planet signals and false positive signals.

Tools, Compute, and Software

GPUs: The augmented and reduced *Astronet* models were individually trained using one NVIDIA V100 GPU provided by Google Cloud. This was the same for the period-phase image approach. However, four of these GPUs were required to train the TPF approach.

Storage: Google Cloud provided the required storage space capable of holding ~10 TBs of *Kepler* and *TESS* data as well as the pipeline outputs (e.g., target pixel files, light curves, DV reports). We utilized Solid State Disk (SSD) drives to enable faster data read-in during training.

Software: The original *Astronet* was written with TensorFlow, thus we re-wrote it in PyTorch to make it more accessible to astronomers. We then continued using PyTorch when developing the augmented/reduced versions of *Astronet*. For the more experimental approaches, we used Keras for the period-phase image approach, and PyTorch for the TPF approach. The development of these codes were often performed using open source Jupyter notebooks. The pre-processing of the data and extraction of labels employed custom Python codes that utilized standard Python astronomy packages, such as Astropy.

Tests & Results

Astronet: The baseline model gave 94.5% accuracy as well as precision/recall vs. MES curves shown in the far left panel of Figure 4. We compare the other models to these values below.

Augmented Astronet: The augmentations to the baseline *Astronet* increased accuracy to 96.0% and significantly improved the precision/recall vs. MES curves, seen by comparing the far left panel to the middle panel of Figure 4. Essentially this model was able to better recover low-MES planets (e.g., smaller rocket planets in the habitable zone) compared to the original *Astronet.*

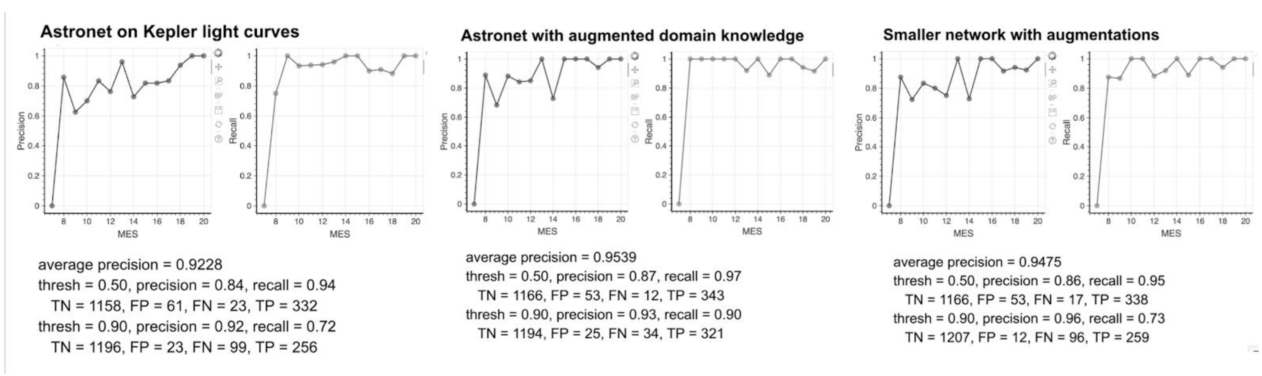


Figure 4: precision/recall vs. MES plots for the baseline *Astronet* model (left), our augmented version of *Astronet* (middle), and our smaller version of *Astronet* (right). MES stands for Multiple Event Statistic and can be thought of as a signal-to-noise ratio of the transit signal.

Smaller Astronet: Our reduced model (which also had the augmentations) was able to obtain 95% accuracy, better than the baseline model, though not as good as the augmented full model. This indicates that the augmentation was balanced by the smaller model size. Nevertheless, this model trained 60% faster than the full *Astronet*, allowing for faster development timescales. We therefore used this smaller model for the cross-validation tests to show the improvements in precision-recall curves as we added each of the augmentation of the scientific domain knowledge areas. This is shown in Figure 5.

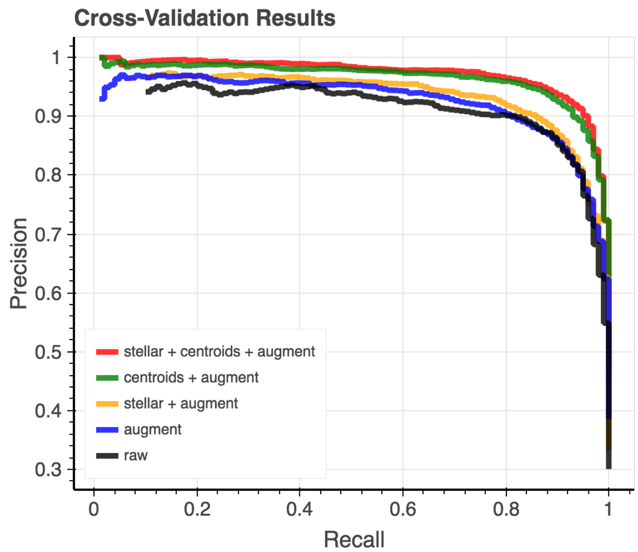


Figure 5: Cross-validation results showing the improvement in the precision-recall curves as we incrementally added the domain knowledge in our augmented but smaller version of *Astronet.*

Target Pixel Files: Because of the novelty of this method, a lot of engineering of infrastructure to load and process the input data was necessary. Assembling the TPF dataset required much effort due to the large TPF files and need to extract the appropriate labels for each file. Searching for transits in TPF files is unlike any prior task we know of that has been approached with deep learning, and required developing a completely new architecture and may yet require novel learning methods. In the final few days of the challenge we realized some of the basic code for loading the images from the TPF files was incorrect invalidating the few results we had. Perhaps this method was a bit too challenging given the time constraints of the FDL, but we now have the infrastructure and experience to give ourselves or anyone following up on this method a great head start. Given more resources following FDL, we plan to explore this method further. In summary, for the approach of learning directly from TPF files, we simply did not have enough time during FDL to get any conclusive results.

Period-Phase Images: This method achieved an accuracy of 91% when training on the *Kepler* light curves. We did not use any data augmentation since this adds to the already computationally heavy requirements of this method. The accuracy quoted here thus cannot be directly compared to the 94.5% accuracy of our baseline *Astronet* model (see above), which uses time flip augmentation. Indeed, the 91% accuracy will likely improve with the addition of data augmentation. We did not try to search over the entire period-phase space. This would increase computational requirements and it would not be directly comparable with the *Astronet* baseline. We instead restricted our search to regions around the TCE labels; extending the search area will be a key area to investigate in future studies. We explored different methods for producing the images (mean, median, etc.) and initial results suggest to add all of these input folding techniques as multiple channels in the network. Ideally, this preprocessing should be done by filters learned from the data in an architecture trained end to end.

Application to TESS

TESS dataset: As no flight data existed at the start of the project, we relied on multiple end-to-end simulations performed by the *TESS* team. Three such runs were performed: an initial 1-sector run named ETE-6, a 2.5-sector run named TSOP-280 and a 4-sector run called TSOP-301. All three runs had simulated populations of planets, eclipsing binaries, background eclipsing binaries, and stellar variability injected, as well as an array of expected detector systematics (e.g. rolling bands and cosmic rays). We focused on the TSOP-301 run, which had the most data and realistic target catalogue.

TESS labels: Unlike *Kepler* data, the ground truth of our simulated *TESS* dataset is known precisely. However, the injected signals are never recovered perfectly–some may be found at the wrong period, or with the wrong durations, etc. Therefore, the degree of correlation between the injected signal and the recovered TCE must be computed - we adapted the *TESS* team's own code which sums in quadrature the overlap between the in-transit (or in-eclipse) points from an injection and from the detection, setting a threshold of 0.75. We split eclipsing binaries into their primary and secondary dips, therefore recovering both signals. We also searched for injections recovered at an integer multiple of the real period, finding a handful of equal-depth eclipsing binaries detected at half the real period. Although complex labels were generated for each target (for example, EB_secondary or BEB_at_2P), we collated all labels from the same source to give between four (Planet, Eclipsing Binary, Background Eclipsing Binary, Non-Astrophysical signal) and two (Planet, not planet) labels.

Balancing TESS data: Training a neural network using a dataset with an unbalanced class distribution is difficult, since the learning algorithm inevitably biases the model towards the majority class. In the case of the *Kepler* dataset, the two classes (planet and non-planet) were roughly balanced since the "unknown" transit label was ignored (see "Data Description"); hence unbalanced data was not an issue for *Kepler*. However, in the *TESS* dataset only 14% of the TCE dataset are planets, and so we found it was necessary to perform dataset balancing in order to train the network. We took the approach of balancing the mini-batches used in training such that each minibatch has an equal number of samples from all the classes. This was also important in multi-label classification.

Models & Methodology: We applied the smaller *Astronet* model developed for the *Kepler* data, repeating the same pre-processing steps and the same training, validation and test set balances. Stellar parameters were taken from the Tess Input Catalogue (TIC).

Multi-label classification results: rather than binary classification, multi-label classification can provide information on the nature of the non-planetary signals. This is important when considering follow-up observations, as different types of false positives require different techniques confirm their nature. For example, an object that is shared between planet and background eclipsing binary classes may require high-contrast imaging and so-called “on-off” photometry, whereas for planets confused with simple eclipsing binaries only radial velocity measurements would suffice. Hence, while multi-class classification may provide lower precision, it may prove more useful for follow-up.

Using the four-label classification model, we achieved an average precision of 0.888 (Figure 6). This precision is lowered by the confusion between eclipsing binary and background eclipsing binary, which is expected given their eclipse shapes are nearly identical. Interestingly, the model performs best on the “unknown” (e.g., non-astrophysical signal) class, suggesting the neural network is able to detect repeated patterns of systematic noise in the dataset. Planets were accurately classified in 84% of cases. We also combined the eclipsing binary and background eclipsing binary classes into a single label, finding both higher average precision (94%) and higher precision on classifying planets (90%) than the four-label version (Figure 6).

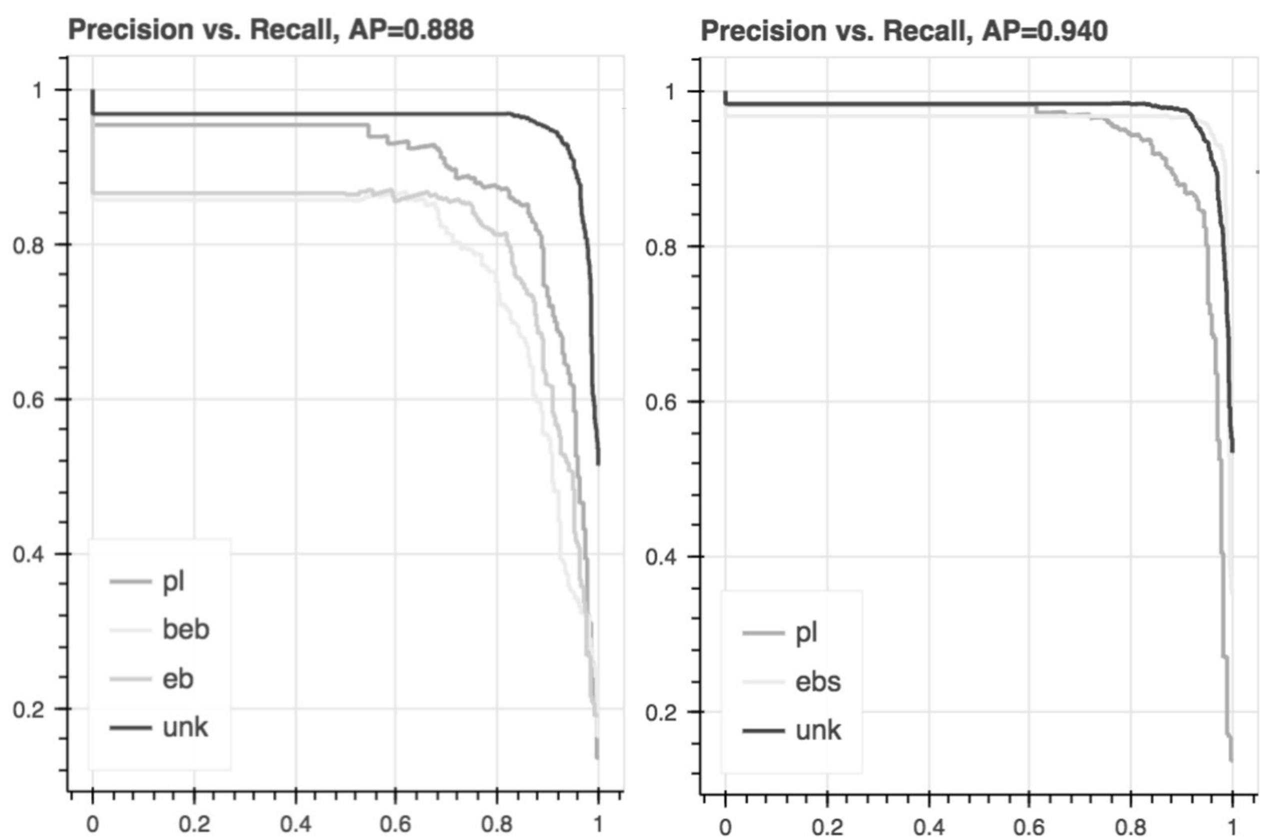


Figure 6: Precision vs. Recall curves for the best four-class and three-class *TESS* datasets. Average precision is shown at the top. ‘pl’, ‘beb’,’eb’ and ‘unk’ stand for ‘planet’,’background eclipsing binary’, ‘eclipsing binary’ and ‘unknown’ respectively, while ‘ebs’ is the combination of

both eclipsing binary and background eclipsing binary classes.

In the current *TESS* pipeline, a large number of detections by the brute-force transit search (TPS; transiting planet search) are whittled down by statistical tests to a smaller population of “TCEs” which we have been classifying previously. We tested our neural network trained on TCEs on the full TPS dataset (~70,000 objects), and found that even without specific training it was able to recover transiting planet signals as well as the statistical tests currently performed (with a “precision” of ~10%; see Figure 7), and was also able to recover the transits of some real, small-radius planets that were otherwise missed by the TPS. This could be vastly improved by training our classifier on the “upstream” TPS dataset instead of the TCEs.

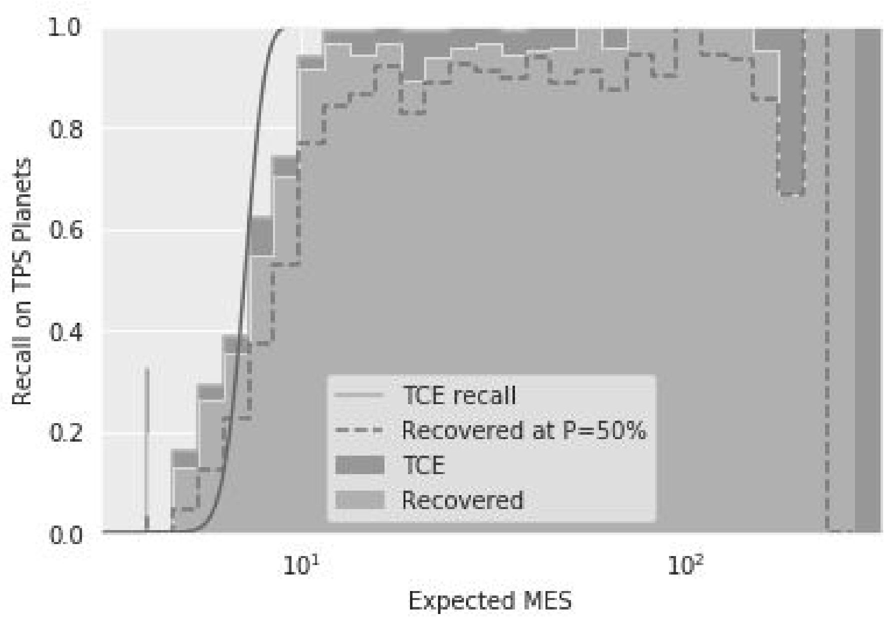


Figure 7: The plot shows the fraction of all injected planet signals that were flagged by a brute-force transit search (TPS) that were recovered (i.e., the “recall”) as a function of signal strength (i.e., the expected Multi-event statistic, or MES). This plot shows the TCE (simple statistical tests), our neural network classifier (at 0.5) and our neural network with a threshold such that the total precision is 50%. Also plotted is the statistical noise threshold expected for a perfect classifier (an error function centred at 7.1-sigma).

Future: To apply this model to the real *TESS* data in time for it to be scientifically useful to the mission, which nominally only lasts two years, we will need to quickly develop a large labeled dataset. The only feasible approach to accomplishing this early in the mission is to create injected light curves in the first few sectors of real *TESS* data that will become publicly available in January 2019. These injections must be accurate representations of planet signals, eclipsing binaries, background eclipsing binaries, and instrumental effects (i.e., unlike for the *Kepler* injected light curves, where eclipsing binaries were simply two injected planet signals).

CONCLUSION

We significantly improved upon the existing state-of-the-art deep convolutional neural network for classification of *Kepler* transit signals, known as *Astronet*. These improvements were enabled by including additional scientific “domain knowledge” when crafting the model architecture and inputs. In short, we added stellar properties and centroid time-series as inputs into the model, and implemented additional data augmentation techniques.

We also drastically reduced the size of the previous neural network architecture while maintaining comparable results, which will likely improve with hyperparameter optimization. This smaller model was a only 0.06% the size of *Astronet*, thereby producing a much more efficient model that cut training time by ~60%.

We extended our improved machine learning model to *TESS* simulated data, achieving slightly worse results in performance, which is expected given the data is simulated and also different from *Kepler* data in terms of cadence and duration as well as planet occurrence. Additionally, we were able to implement multi-class classification (rather than the binary classification used in *Astronet*) and apply innovative data re-balancing techniques (since exoplanets are rare among the candidate transit signals) to improve performance.

These models are ready to be re-trained and applied to real *TESS* data when it becomes available starting in January 2019. However, reliable labels for large training datasets are needed immediately; we cannot wait until the end of the mission, as was done for *Kepler*, to develop labeled datasets. The best way forward is to quickly produce injected light curves into real *TESS* data as soon as it becomes available; in order for this training set to be effective, these injections must be reliable representations of planet signals, eclipsing binaries, background eclipsing binaries, and instrumental effects (i.e., unlike for the *Kepler* injected light curves, where eclipsing binaries were simply two injected planet signals).

Finally, we explored two exciting new avenues for exoplanet transit detection and classification using deep learning. These are notably different from the above methods, not only because they will be able to detect (not just classify) transit signals, but also because they use new and innovative methods not previously explored. One method aimed to detect and classify transits in period-phase “images”, while the other attempted to detect and classify transits from the target pixel files (rather than the extracted light curves). Both of these methods would revolutionize the field of exoplanet transit searches, but require further exploration.

References

Batalha, Rowe, Bryson, et al. (2013), “Planetary Candidates Observed by Kepler. III. Analysis of the First 16 Months of Data,” *The Astrophysical Journal Supplement*, 204, 24.

Borucki, Koch, Basri, et al. (2010), "Kepler Planet-detection Mission: Introduction and First Results," *Science*, 1185402.

Jenkins, Tenenbaum, Caldwell et al. (2018), “A Simulated Data Set for the Transiting Exoplanet Survey Satellite,” *Research Notes of the AAS*, Volume 2, Number 1.

Jenkins, Doyle, and Cullers (1996), “A Matched Filter Method for Ground-Based Sub-Noise Detection of Terrestrial Extrasolar Planets in Eclipsing Binaries: Application to CM Draconis,” *Icarus*, 119.

Ricker, Winn, Vanderspek, et al. (2014), "Transiting Exoplanet Survey Satellite," *Journal of Astronomical Telescopes, Instruments, and Systems*, 014003.

Shallue & Vanderburg (2017), “Identifying Exoplanets with Deep Learning,” *The Astronomical Journal,* 155, 94.

van een Oord, Dieleman, Zen, et al. (2016), "WaveNet: A generative model for raw audio." arXiv:1609.03499.

Other Contributors: We thank Chris Shallue for his useful input during the course of this project regarding his experiences building *Astronet*.