Identify Light-curve Signals with Deep Learning Based Object Detection Algorithm. I. Transit Detection

Kaiming Cui[1,2,4] , Junjie Liu[3], Fabo Feng[1] , and Jifeng Liu[2,4,5]

[1] Tsung-Dao Lee Institute, Shanghai Jiao Tong University, 800 Dongchuan Road, Shanghai 200240, People's Republic of China; cuikaiming@sjtu.edu.cn
[2] CAS Key Laboratory of Optical Astronomy, National Astronomical Observatories, Chinese Academy of Sciences, Beijing 100101, People's Republic of China
[3] Megvii Research Insitute, Megvii Technology, Chengdu, Sichuan 610095, People's Republic of China
[4] School of Astronomy and Space Sciences, University of Chinese Academy of Sciences, Beijing 100049, People's Republic of China
[5] WHU-NAOC Joint Center for Astronomy, Wuhan University, Wuhan, Hubei 430072, People's Republic of China

Abstract

Deep learning techniques have been well explored in the transiting exoplanet field; however, previous work mainly focuses on classification and inspection. In this work, we develop a novel detection algorithm based on a well-proven object detection framework in the computer vision field. Through training the network on the light curves of the confirmed Kepler exoplanets, our model yields about 90% precision and recall for identifying transits with signal-to-noise ratio higher than 6 (set the confidence threshold to 0.6). Giving a slightly lower confidence threshold, recall can reach higher than 95%. We also transfer the trained model to the TESS data and obtain similar performance. The results of our algorithm match the intuition of the human visual perception and make it useful to find single-transiting candidates. Moreover, the parameters of the output bounding boxes can also help to find multiplanet systems. Our network and detection functions are implemented in the `Deep-Transit` toolkit, which is an open-source Python package hosted on Github and PyPI.

1. Introduction

The increased amount of data in astronomy makes it necessary to apply machine-learning algorithms for regression, classification, detection, clustering, forecasting, etc. As a typical branch of machine-learning methods, deep learning algorithms have been rapidly evolving in recent years, possessing better performance compared to traditional methods in image classification (Krizhevsky et al. 2012), visual recognition (Girshick et al. 2014), and many other areas. Some featured algorithms, such as convolution neural network (CNN; Lecun et al. 1998), recurrent neural network (RNN; Rumelhart et al. 1985), long short-term memory (LSTM; Hochreiter & Schmidhuber 1997), and generative adversarial networks (GAN; Goodfellow et al. 2014), are also used in astronomy. For example, Leung & Bovy (2019) develop the `astroNN` package for spectra analysis, and Charnock & Moss (2017) and Muthukrishna et al. (2019) use RNN for transients classification on multiband photometric time series. Liu et al. (2019) use LSTM to predict solar flares based on some active region information and flare histories. Schawinski et al. (2017) use GAN for recovering galaxy morphology.

In the field of signal detection and classification of light curve, 1D CNN, RNN, and LSTM are commonly used (e.g., Hinners et al. 2018; Feinstein et al. 2020). As a very important and attractive project, finding transiting exoplanets from light curves also makes extensive use of deep learning techniques, with 1D CNN being the primary method. Pearson et al. (2018) first use 1D CNN to search for exoplanets based on iterative use of a binary classifier, and Zucker & Giryes (2018)

independently develop a similar work with different data set. Meanwhile, a series of work on exoplanet vetting based on 1D CNN starting from Shallue & Vanderburg (2018), who work on Kepler data with local and global views for folded light curves of transit candidates. Following works introduce external information (e.g., Ansdell et al. 2018, includes centroid curves and stellar parameters) or applied it to K2 data set (Dattilo et al. 2019), TESS data set (Yu et al. 2019; Osborn et al. 2020), Wide Angle Search for Planets program (Schanche et al. 2019), and the Next Generation Transit Survey (Chaushev et al. 2019). Olmschenk et al. (2021) develop a systematic pipeline for detecting and vetting transiting exoplanets from TESS full-frame image (FFI) light curves. They apply a 1D CNN for outputting the confidence of existing potential transiting exoplanets in a given FFI light curve.

In this work, we explore a two-dimensional (2D) object detection algorithm to identify transiting signals. 2D means the detection is performed on the image. Compared with previously 1D CNN, our approach is straightforward and matches human visual intuition. We implement our neural network based on Kepler, then transfer learning to TESS. Section 2 introduces the Kepler training data preparation (Section 2.1), network architecture (Section 2.2), detailed training steps (Section 2.3), and test and comparison results (Section 2.4). In Section 3, we transfer the model trained on Kepler to TESS data. After we have a trained model, further applications and inspections can be developed based on our results. Section 4 shows several application examples, and Section 5 discusses features and limitations of our method. Our training and detection algorithms are implemented in the `Deep-Transit`, which is an open-source Python package.[6]

[6] https://github.com/ckm3/Deep-Transit

2. Implementation and Training

Different from the commonly used 1D CNN and RNN (LSTM) algorithm, we plot the light curve to an image, so that the 1D time series is converted into a 2D image. This allows the rapidly developing computer vision algorithms to be applied to light-curve data. In terms of data preparation and network structure, our approach has two main advantages over previous algorithms.

1. The sampling frequency requirement for light curves is relatively lenient. For 1D CNN and RNN (LSTM), the input dimension is a fixed number of data points, so that different data set require interpolation or binning. In contrast, our method is less sensitive to the sampling frequency and preserves the original information of the data.
2. Our network has an output combining three scales: large, medium, and small. Thus it can be adapted to different information densities generated by different data sets without modifying network structure.

2.1. Data Set Preparation

As one of the most successful light-curve products, Kepler light curves are ideal for our work. Our data sets are selected from the Kepler light curves for confirmed transiting planets[7]. The time, durations, and periods of those transits are collected from the Kepler Threshold-Crossing Event catalog (Twicken et al. 2016). Compared with synthetic data sets, real data have more realistic stellar activity signal and systematics. Before creating our training and validation data, we randomly select the Kepler light curves for 100 confirmed exoplanets as the test data.

Similar to other transit detection algorithms, we also need to detrend instrumental and the photometric variation caused by stellar activity. To do so, we apply the Tukey's biweight algorithm (Mosteller & Tukey 1977) implemented by Wotan[8], whose performance is proven in blind search of Kepler and K2 data (Hippke & Heller 2019). The parameters we use are the default parameters in Wotan. Then, we have a sigma clipping for the detrended light curves. The upper clipping limit is three standard deviations and the lower clipping limit is 20 standard deviations.

Since our method relies on the visual presentation of the data, the method of conversion from light-curve data to images is critical. In our work, we first split the light curve into 30 day segments (less than 30 days are not split) and make the next window overlap the previous window by 5 days (i.e., a 30 day sliding window size with a 25 day step). Then, in each 30 day segment, the light curve is split into 10 day segments, and the next segment overlaps the previous one by 3 days (i.e., a 10 day sliding window with a 7 day step). We generate images for each of the 10 day light-curve segments. We choose the window length based on our experience and data propriety of Kepler and TESS, and it can be modified for different type of signals. Actually, after training, a slight modification of the window length has little effect on the results. For convenience, our image size is chosen to be 416×416, which is a commonly

used size for object detection. Larger size could has better performance but cost more computation time.

Based on our experience in light-curve identification, we plot data in line with point markers to enhance the characteristics of the light curve. The line width is set to 1 pixel, and the data point is the circle with a 2 pixel radius. These choices are flexible, because after training is complete, the network has the ability to generalize and can handle slightly different plotting styles. To keep things as simple as possible, all the lines and points are black, so the image is a single-channel bitmap with the shape of (416, 416, 1). The flux range for plotting is the flux range of 30 day segments, rather than within 10 days. This is because we want our model to have some perception of a wider range of time and avoid some local false positives. Also, we extend the lower limit of 30 day segments' flux by 2% to avoid the lowest flux of a transit being at the edge of an image.

The format of our bounding box label is (x_{middle}, y_{middle}, w, h, S/N). The x_{middle} and y_{middle} are the normalized coordinates of the center of a transit (i.e., convert the coordinates to 0–1, the coordinate of the left edge is 0, the right edge is 1, the top edge is 0, and the bottom edge is 1). The width w of our bounding box is set to twice the normalized duration, and if the width is less than 0.02, then enforce it to 0.02. The top value of the bounding box is 95% of the flux of the 10 day segment. To ensure that the bounding box can cover the whole transiting region, we extend the height h down by 4 pixels. We also calculate the signal-to-noise ratio (S/N) for each transit. The median and standard deviation are derived from the out-of-transit region of the 10 day segment. The depth of transit is estimated from the median value of the middle half. In our training set, only transits with S/N > 3 are generated. Figure 1 is a schematic diagram of our training data set generation.

After the data set preparation, we have more than 140,000 images, and each of them has at least one transit event. Then, we randomly split the data into a training set and a validation set at a ratio of 85% and 15%. Different from the previously preserved test set, our training and validation sets are split on the image level to make the parameters of the transits evenly distributed in the training and validation sets as much as possible.

2.2. Network Architecture

We do not need to invent a new object detection network because there are already a lot of well-proven algorithms and tools in the computer vision field. We choose the well-established You Only Look Once, Version 3 architecture (YOLOv3; Redmon & Farhadi 2018), which has a wide range of applications (e.g., Choi et al. 2019; Tian et al. 2019; Yurtsever et al. 2020) and performs better than previous algorithms (e.g., R-CNN; Girshick et al. 2014; Faster R-CNN; Ren et al. 2017) in terms of small object recognition and computational speed. Our network architecture is shown in Figure 2. It has one input head and an output of three scales for large, medium, and small objects. The final output combines the results of three scales through the Non Maximum Suppression (NMS) algorithm, which is a standard way for only keeping best bounding boxes (Rosenfeld & Thurston 1971). The main procedure of NMS is to iteratively preserve the boxes with the highest confidence score and then remove other boxes depending on a threshold.

Original YOLOv3 network is designed for detecting daily common objects, so they set their prior anchor boxes based on

---

[7] https://exoplanetarchive.ipac.caltech.edu/docs/counts_detail.html
[8] https://github.com/hippke/wotan
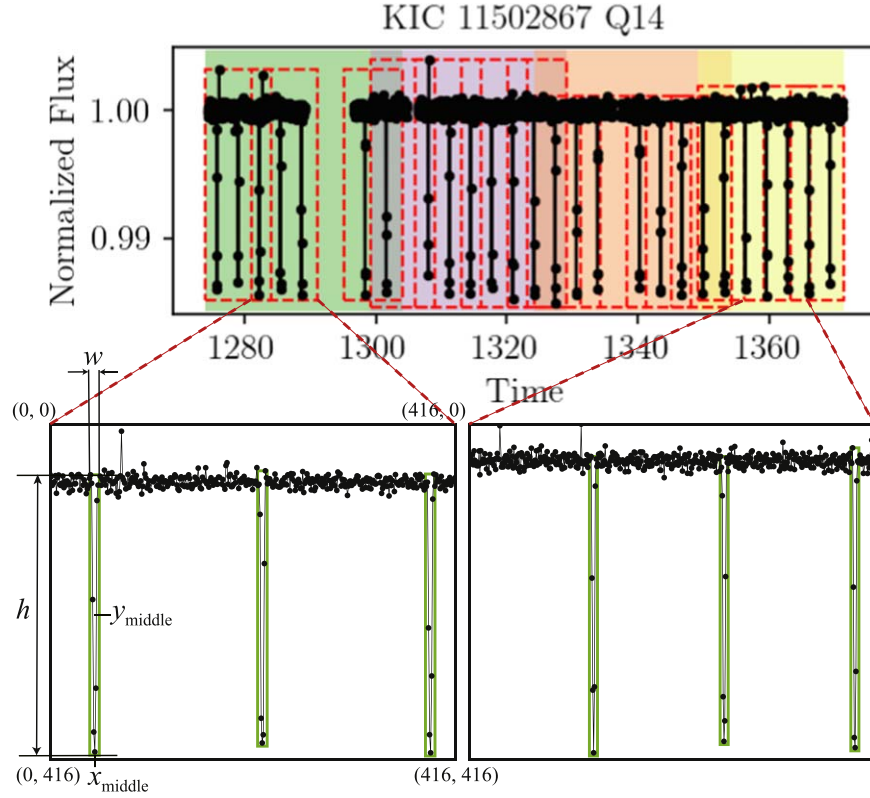
Figure 1. Schematic diagram of training data set generation steps. The top panel shows a part of a detrended light curve. The color shaded sections indicate 30 day windows, and they have 5 day overlapped areas between adjacent windows. The dashed red lines are 10 day selection ranges, and they have 3 day overlapped areas between adjacent ranges. The bottom two panels are the zoom-ins of two 10 day selection ranges, and the bounding boxes are also plotted with green rectangular boxes. The coordinates of a bounding box (x_{middle}, y_{middle}, w, h) and four corners are indicated in the bottom left panel.

some competition data sets (e.g., PASCAL Visual Object Classes Challenge and Microsoft Common Objects in Context). Anchor boxes are crucial parameters in YOLO, they are a set of predefined bounding boxes of a certain height and width. However, in our work, we are aiming at identifying transits, the size of anchors should be determined by the characteristics of the signals, and since we have only one class, too many anchor boxes are overkill for our detection. Thus, we measured the height and width distribution of all the boxes in our data set, and then we manually selected three anchors to roughly get close to the size of most of the transiting signals, while considering assigning different sizes of anchors for different scales. Finally we reset the width and height of anchor boxes to (0.1, 0.9), (0.05, 0.7), and (0.02, 0.3) for three scales accordingly. The width is significantly lower than the height, consistent with the basic pattern of a transit. We try adding more anchors but it does not improve the performance. We also compare the original YOLOv3 anchors with our modified version and find that the modified one is better in terms of transit identification.

Usually, the confidence score of a bounding box is trained to equal the intersection over union (IOU) between a predicted box and the ground truth. IOU is defined as a ratio of the areas of intersection and union between the predicted bounding box and the ground-truth bounding box. IOU describes the accuracy of a predicted box; the higher the IOU, the more accurate the overlap with the target. However, the range of the target box is artificially selected by us, so the high overlap does not have much physical sense. Instead, we consider a more physical expression of the confidence score to be the S/N. We can easily calculate the S/N for a single-transit event, and an the S/N can be naturally converted to a confidence score with a normalization method. In this work, our confidence scores $C_{\mathrm{S/N}}$ of training data are calculated by $C_{\mathrm{S/N}} = 1 - \exp(-0.15\ \mathrm{S/N})$. The $C_{\mathrm{S/N}} \approx 0.6$ when the S/N is 6, and then rapidly increase to 1, matching our naked eye's intuitive perception and previous research thresholds (e.g., Kovács et al. 2002; Kunimoto et al. 2018). A different normalization method is also acceptable for different requirements.

Then, we modify the original YOLOv3 loss function because of the different number of anchor boxes and confidence scores. As shown in Equation (1), loss function of each scale is a combination of three parts and is defined as

$$
\begin{aligned}
L &= L_{\mathrm{coord}} + L_{\mathrm{obj}} + L_{\mathrm{noobj}}, \\
&= \lambda_{\mathrm{coord}} \frac{1}{N} \sum_{i=0}^{S^2} \mathbb{1}_i^{\mathrm{obj}}[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \\
&\quad + \lambda_{\mathrm{coord}} \frac{1}{N} \sum_{i=0}^{S^2} \mathbb{1}_i^{\mathrm{obj}}[(w_i - \hat{w}_i)^2 + (h_i - \hat{h}_i)^2] \\
&\quad - \frac{1}{N} \sum_{i=0}^{S^2} \mathbb{1}_i^{\mathrm{obj}}[C_{\mathrm{S/N}} \log(C_i) \\
&\quad + (1 - C_{\mathrm{S/N}}) \log(1 - C_i)] \\
&\quad - \lambda_{\mathrm{noobj}} \frac{1}{N'} \sum_{i=0}^{S^2} \mathbb{1}_i^{\mathrm{noobj}} \log(1 - C_i).
\end{aligned}
\tag{1}
$$
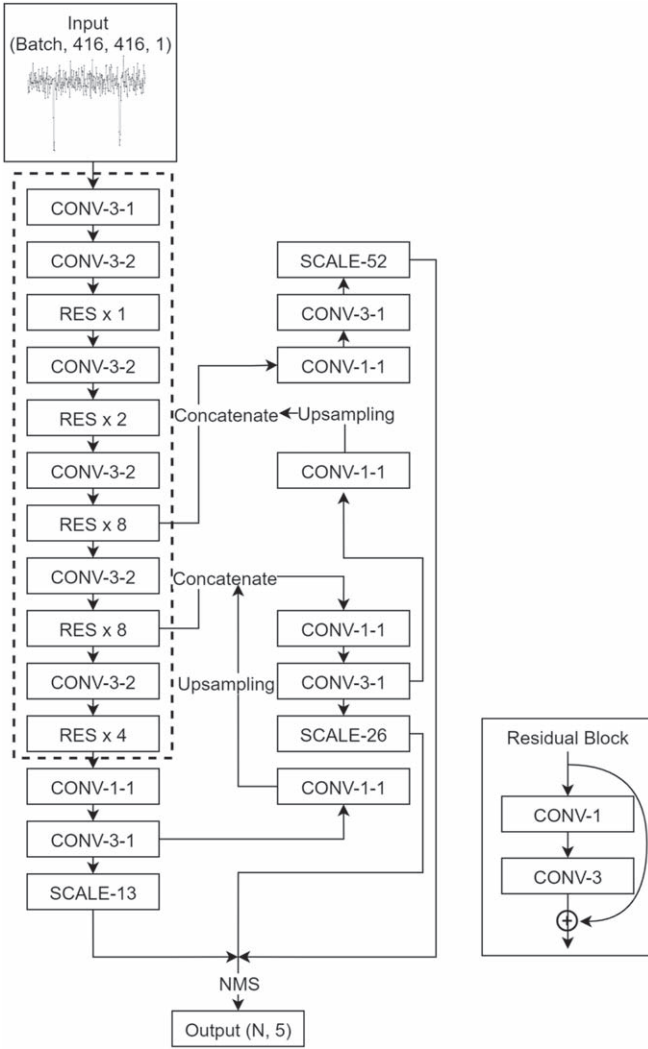
Figure 2. Network architecture. Our network structure is based on a standard YOLOv3 architecture with a slightly modified input channel and output dimension. The dashed line region indicates the Darknet-53 backbone. Each convolution block is shown as a CONV, it is combined by a 2D convolution layer, a batch normalization layer, and a Leaky ReLU function (Maas et al. 2013). The numbers split by dashes of a convolution block show the size and stride of the convolution layer. Each residual block is abbreviated as RES. As shown on the right side of the figure, a residual block is combined by two convolution blocks with a shortcut. The scale block is shown as SCALE, and the following number is the size of each scale. Similar to the residual block, each scale block has two convolution blocks but without a shortcut. Combining the results of the three scales to calculate the NMS, the final output is the coordinates and confidence level of N-detected bounding boxes.

The sum of the first two terms is the coordinate mean squared error (squared L2 norm) loss. x_i, y_i, w_i, h_i denote the predicted coordinates of boxes in the cell i, similarly, $\hat{x}_i$, $\hat{y}_i$, $\hat{w}_i$, $\hat{h}_i$ denote the target coordinates of boxes. S is the scale size. $\mathbb{1}_i^{obj}$ equals 1 when there is an object in the cell i, otherwise 0; $\mathbb{1}_i^{noobj}$ equals 1 when there is no object in the cell i. N is the number of object cells for calculating the mean value. The third and fourth terms show the binary cross-centropy losses of the object and no object cells. Binary cross entropy is a commonly used cross-entropy loss for binary classification applications. C_i is the predicted confidence score of whether there is an object or not. Similar to N, N' is the number of cells without object. λ_{coord} and λ_{noobj} are the constants to increase the attention of certain parts. In our work, we choose $\lambda_{coord} = \lambda_{noobj} = 10$ to ensure the
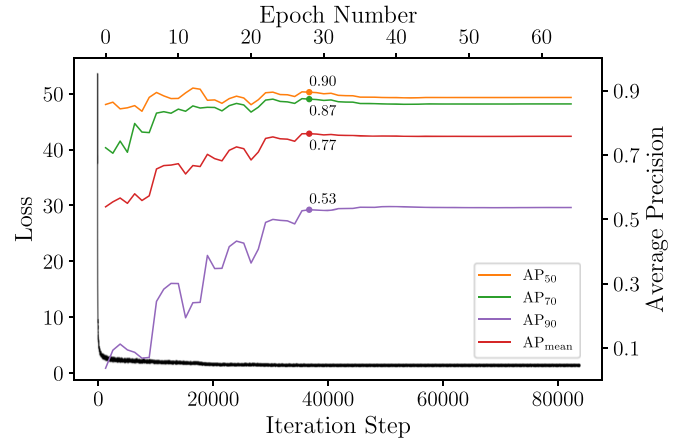
Figure 3. Training steps on the Kepler light curves. The black line is the loss and the APs are plotted as different colors for different IOU thresholds. The AP values of the best model are marked and written on the figure.

balance of those three losses. Compared with original YOLOv3 loss function, we do not have the class loss, because we only focus on transit signals.

2.3. Training Steps

There are two key metrics precision and recall. Precision is defined as $TP/(TP + FP)$, and recall is defined as $TP/(TP + FN)$. The TP, FP, and FN indicate true positive, false positive, and false negative. Average precision (AP) is calculated by the area under the precision-recall (PR) curve. AP_{50} means AP with the IOU threshold greater than 0.5, AP_{70} means AP with the IOU threshold greater than 0.7, and so does AP_{90}. We choose the mean value of AP_{50}, AP_{70}, and AP_{90} as our training evaluation metric (AP_{mean}). The AP_{50} is the baseline of many algorithms because IOU larger than 0.5 is sufficient for most ordinary target detection tasks. We consider AP_{70} and AP_{90} because we want the parameters of the box to provide better help for subsequent applications.

After several trials based on the performance on the validation set, we choose the Adam optimizer to optimize the total loss L. The initial learning rate is 10^{-4}, and the weight decay is 10^{-4}. A lower learning rate would increase the training steps significantly. We also reduce the learning rate by a factor of 2 when our metric has stopped improving for two epochs. The epoch is the number of passes of the entire training data set. In our work, each epoch has 1265 iteration steps. To improve training efficiency, we input a batch of training examples in each iteration step. Our batch size is 96, lower batch size performs worse. The NMS threshold is 0.1, the IOU threshold is 0.5, and the confidence score threshold is 0.6. Each epoch takes about 20 minutes on an NVIDIA Tesla V100 GPU.

Figure 3 shows the loss, AP_{50}, AP_{70}, AP_{90}, and AP_{mean}, vary during the training. Those APs are calculated on the validation data, and we can see them increase and then keep constantly after longer epochs. Due to overfitting, the loss on the training set continues to drop, but the metrics on the validation set are no longer rising. Therefore, we choose the model with the largest AP_{mean} on the validation set as the best model. The PR curve of the best model is shown in Figure 4. Our final best model has an $AP_{50} = 0.9$, which is an excellent performance
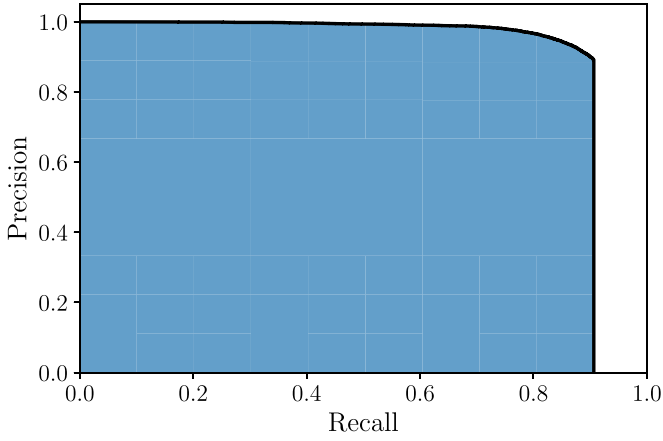
Figure 4. The precision-recall curve for IOU > 0.5 of the best model. The blue shaded area is the AP_{50}. The sharp drop at 0.9 is due to our relatively high confidence threshold (0.6), making recall only go up to 0.9.
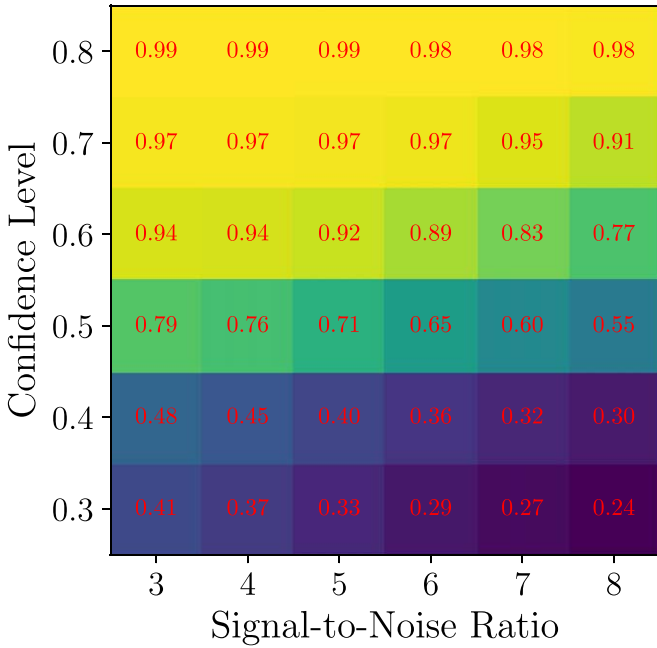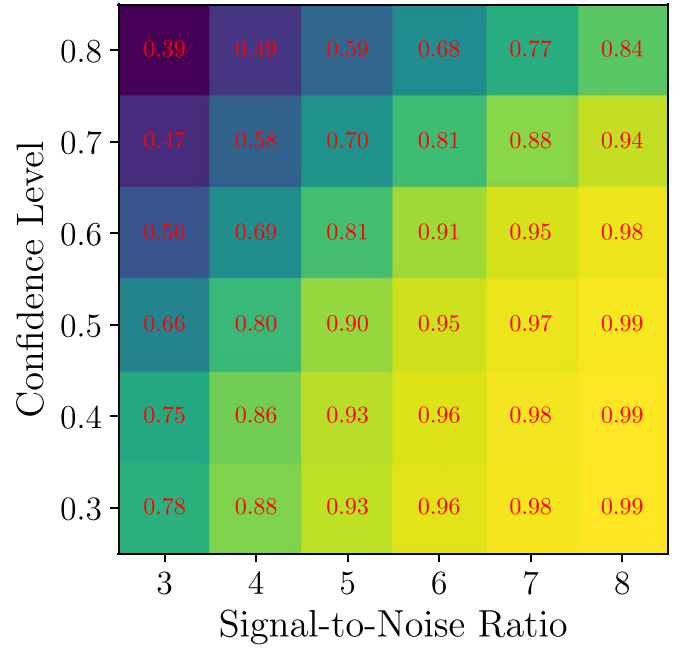

Figure 5. Precision matrix of the best model trained from Kepler on the test set.

2.4. Model Tests and Comparisons

After finishing the training, we need to test the performance of the model. Although our high AP_{mean} on the validation set gives us confidence, we still want to know its performance on real detection when facing transits with different S/Ns. To do so, we apply our best model on 100 previously reserved exoplanet hosts. Consistent with the real detection task, we use the detection function implemented in our `Deep-Transit` package with default parameters. To use more physical criteria, we treat a detected box as a true positive when the time of the transiting midpoint locates in the box range. We apply our test on different confidence levels and different S/Ns of transits.
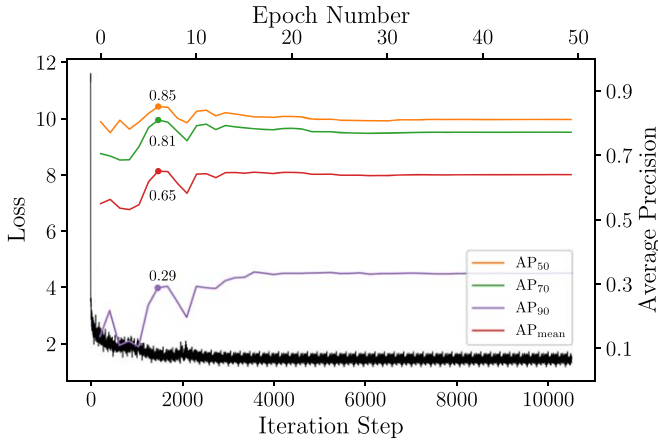

Figure 6. Recall matrix of the best model trained from Kepler on the test set.

Figure 5 and Figure 6 show the precision and recall matrices accordingly. These two matrices present a comprehensive evaluation of the detection ability of our model. We can also easily recognize that the trade-off between precision and recall. For the detected confidence scores higher than 0.6, our model's precisions can achieve high to 0.9 in most cases. Higher confidence scores allow precisions close to 1. For the confidence scores lower than 0.5, the recalls are higher than 0.8 in most cases. Lower confidence scores also make recalls close to 1 for S/Ns higher than 6. Therefore, our model has high practicality and can control the precision and recall of detection by adjusting the confidence threshold. To have a balance between precision and recall, 0.6 can be a Goldilocks score in most cases. To perform an extensive search, 0.5 is a practical choice.
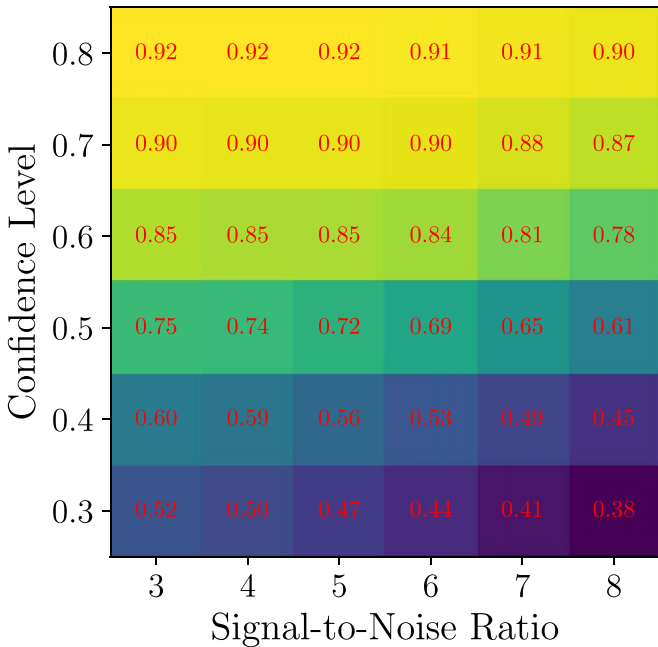
We also compare our model with two practically used baselines. Unlike other commonly used exoplanet detection algorithms (periodogram-based algorithms such as box-fitting least squares), our model lacks period information. Therefore, a more appropriate baseline is the algorithm of single-transit detection. One of the most straightforward and conventional single-transit detection algorithms is a transit fitting with a moving window, which is applied by most single-transit event searches of Kepler and K2 with only minor differences (e.g., Foreman-Mackey et al. 2016; Osborn et al. 2016; Kawahara & Masuda 2019). We implement this algorithm by following Osborn et al. (2016). Our transit model is an analytic model with quadratic limb darkening (Mandel & Agol 2002), and it is generated with the `Batman`, which is an open-source Python package for fast calculation of exoplanet transit light curves. The window sizes and transit durations are exactly adopted from Osborn et al. (2016). We applied this algorithm to our test set. For transits with S/N > 6, the precision is 0.59 when the recall reaches 0.91 by tuning the reduced χ^2 threshold. The higher number of false positives is also confirmed in Kawahara & Masuda (2019), this is also why single-transit detection usually requires visual inspections to rule out false positives.
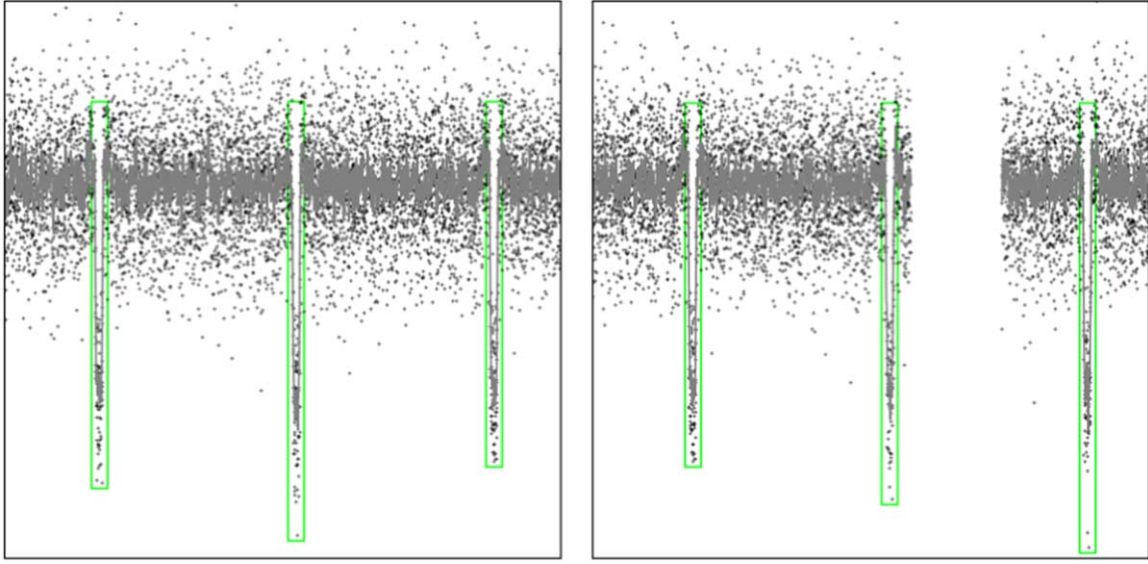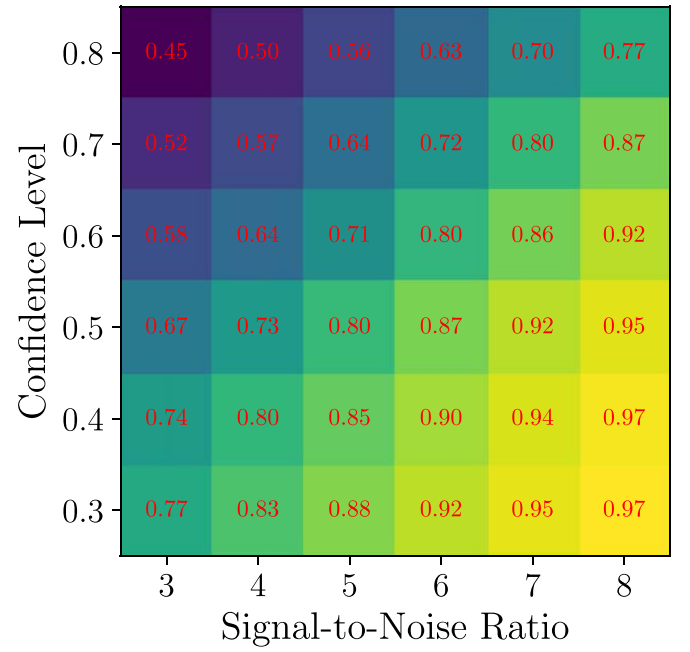
Figure 7. Two examples of TESS training and validation data. The bounding boxes are shown as green boxes. Note that the actual sample does not have a border.

**Figure 8.** Training steps of transfer learning on TESS data. Same as Figure 3.

Figure 10. Recall matrix of the best model trained from TESS on the test set.

Table 1
Table of Parameters of Transits of Kepler-297 from Rowe et al. (2014)

Parameters			
KOI	1426.01	1426.02	1426.03
Kepler ID	Kepler-297 b	Kepler-297 c	
Period (days)	38.871826	74.920137	150.018303
R_p/R_*	0.02876	0.06543	0.59798
Transit Depth	940.4	4150.1	4437.6
Transit Duration	6.125	4.794	4.400

Figure 9. Precision matrix of the best model trained from TESS on the test set.

Our network is based on computer vision techniques, thus we have fewer false positives.

Another baseline is given by the naked eyes. Many exoplanet detection pipelines require visual vetting to remove false positives (e.g., Batalha et al. 2013; Huang et al. 2013;
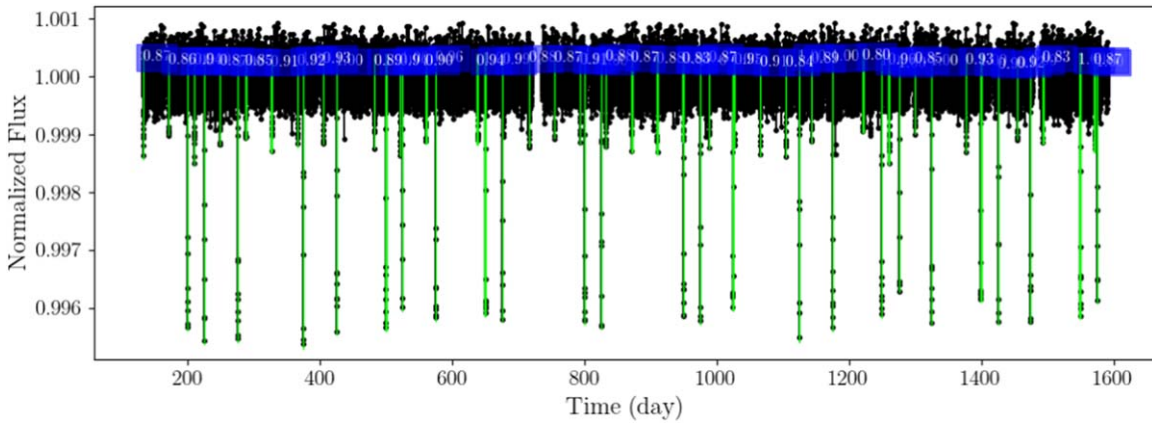
Figure 11. Light-curve and transit detection results of Kepler-297. The green boxes are the detected bounding boxes, and their top square blue labels indicate the confidence scores.
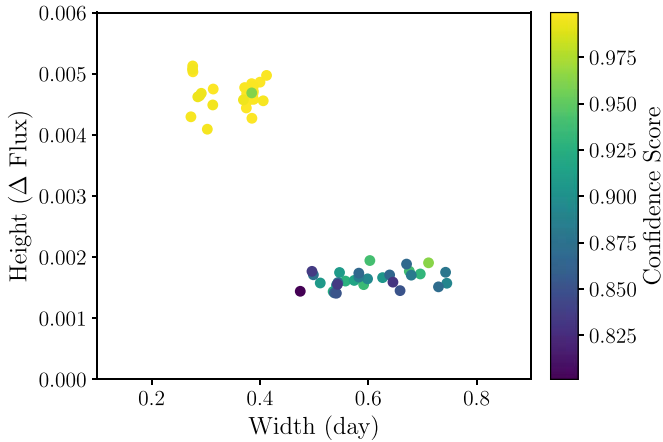
Figure 12. Scatter plot of the width and height of the detected boxes. The color of the points indicate the confidence level.

Guerrero et al. 2021). Also, as one of the most famous citizen science projects, Planet Hunters[10] attracts volunteers to visually detect exoplanets. According to their papers (Fischer et al. 2012; Eisner et al. 2021), the practical recovery rate of Planet Hunters is about 67% for Kepler data, and 53% for TESS objects of interest (TOI) data. The spurious transit boxes rate is about 10%. Therefore, compared with the Planet Hunters, our network performs similar precision but higher recall. The result is reasonable because our trained model is more sensitive to transit signals compared to amateur volunteers.

3. Transfer Learning to TESS

As an ongoing survey, TESS provides higher duty cycle light curves than Kepler. Therefore, we also wish to apply our well-trained model to the TESS data. Usually, deep learning algorithms can only achieve high performance on a specific task. It means that our model trained on Kepler can not be directly applied to TESS. However, since the main difference between TESS and Kepler is their photometric accuracy and sampling rates, we can create a well-performing model for TESS data based on the pretrained Kepler model. To do so, a transfer learning technique can be helpful.

Transfer learning inherits existing relevant data or models while building a new machine-learning model. In practice,

transfer learning is a standard way when there are not enough training data, or the source and target domains have some similarities but are not identical (Pan & Yang 2009). Therefore, when considering the short observation time of TESS, resulting in a small amount of data, transfer learning is an appropriate choice. To achieve this, we need to prepare the data set for TESS and then load the model already trained on the Kepler data as a pretrained model. The training on TESS data only requires a fine-tuning of the pretrained model.

3.1. Training and Evaluation

First, we create the TESS data set similarly as we create the Kepler data set (described in Section 2.1). Since there are only over 100 confirmed exoplanets detected by TESS, our training and validation data set are created from the TOI catalog (Guerrero et al. 2021). The main difference in the data set preparation is that the TESS images are plotted using scatter points with a 30 day moving average gray line overlaid. The points are 1 pixel points and the line width is 1 point. Figure 7 shows two examples of the TESS training data set.

Then, we use the same network structure and load our pretrained Kepler model to initialize the network without freezing any layers. As the model converges quicker, we reduce the learning rate by a factor of 2 when our metric has stopped improving for one epoch. The performance during training steps is shown in Figure 8. Similarly, as described in Section 2.3, we choose epoch 6 as the best performance model. Also, the best model can be downloaded online.[11]

Finally, we test the performance of our model using light curves of confirmed TESS exoplanet hosts. Due to the relatively small amount of data in TESS, we manually adjust some parameters of the detrending method. Like with the previous Kepler model, the test results are shown as two matrices (see Figures 9 and 10). Their general performance is good but relatively lower than Kepler, which might be caused by the lower data quality of TESS. The recall is better than precision, makes it more suitable for a large-scale transit search.

---

[10] https://www.zooniverse.org/projects/nora-dot-eisner/planet-hunters-tess

[11] The states of the network trained from TESS data are stored to a model file, which can be directly downloaded from https://nadc.china-vo.org/article/20200722160959?id=101079.

(a)

(b)

(c)
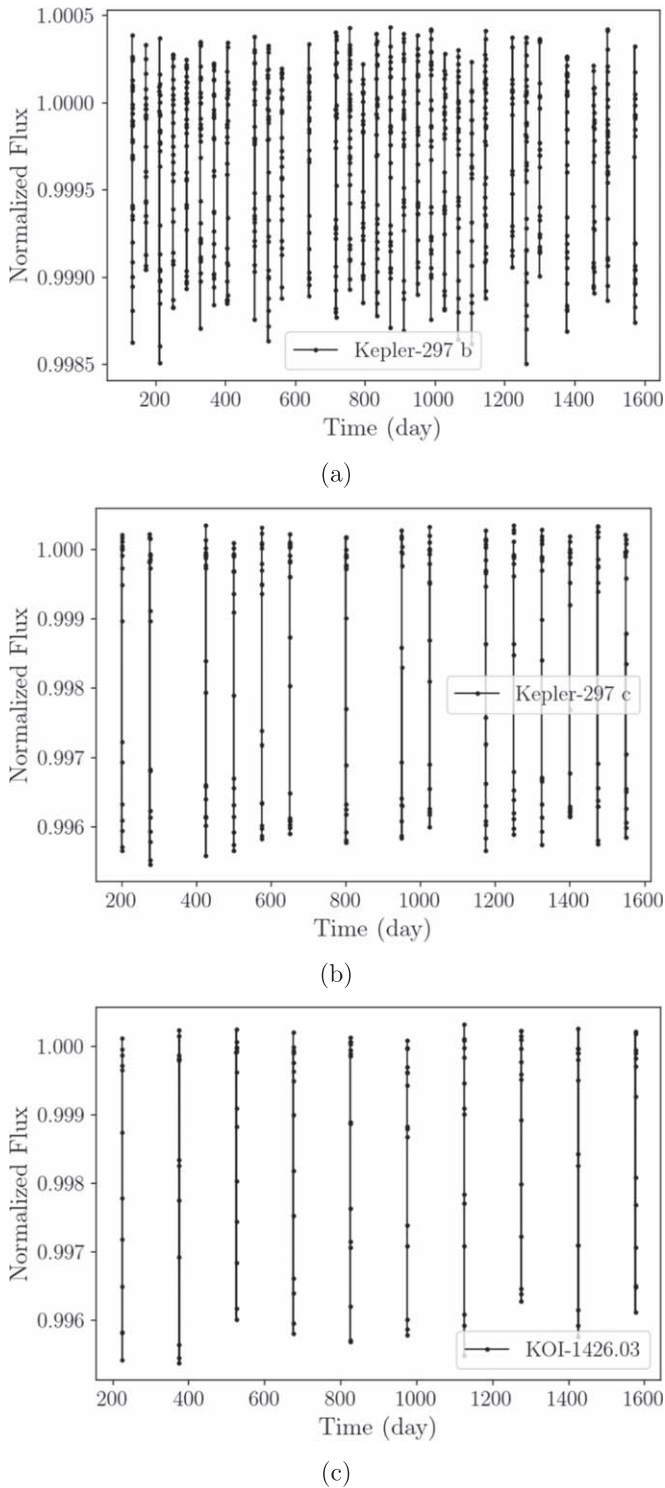
Figure 13. Selected light curves from our detected bounding boxes. (a) is the selection of detected transiting events of Kepler297b. (b) is the selection of detected transiting events of Kepler297c. (c) is the selection of detected transiting events of KOI-1426.03.

4. Application

After the training and evaluation of the model is completed, our algorithm can have some good practical applications.

Once the bounding boxes with confidence scores of transits are successfully detected, the width and height of boxes can be used as a rough estimation for the duration and depth of the

transits. For example, Kepler-297 (KIC 11122894) has three transit candidates and two of them are confirmed (Lissauer et al. 2011; Rowe et al. 2014). Table 1 list some of their parameters from Rowe et al. (2014).

We can apply our detection method to the 4 yr light curve of Kepler-297 to reproduce its multiplicity. Figure 11 shows the flattened light curve and our detected bounding boxes. The confidence threshold is 0.8. After we put the width and height of the detected boxes into a scatter plot, we can see two obvious clusters in Figure 12. If we look closely, the yellow points in the upper-left corner also seem can be divided into two parts. Actually, they have similar depth and width but different periods. By selecting the light curve in the detected bounding boxes, we can perform a more focused period detection. As shown in Figure 13, the periods of those three clusters can be estimated from the middle time directly.

Since our algorithm is complete and accurate to a single transit with high $S/N > 7$, it has a strong potential for single-transits detection. Kawahara & Masuda (2019) yields 67 single-transit candidates in Kepler light curves from previous work (e.g., Uehara et al. 2016; Foreman-Mackey et al. 2016). Nearly all of them can be detected by our model with proper detrending (the default window size 0.5 day will remove some long duration transits). Figure 14 shows an example of KOI-1174, which has a single-transiting exoplanet candidate. Our detection algorithm can easily find the only transit with a confidence score higher than 0.8. Finding single transits becomes even more important due to the observation mode of TESS (e.g., Cooke et al. 2018; Villanueva et al. 2019; Yao et al. 2021). Our model can be a powerful tool for future single-transits detection work.

5. Discussion

From the human perception, our method is intuitive and can help some visual searches and inspections. For example, the individual vetting and group vetting of TOIs could apply methods similar to ours to reduce manual effort and increase efficiency. Also, some citizen science projects like Planet Hunters and Citizen ASAS-SN[12] can be partly replaced by our method. An alternative approach is to use the results of citizen projects as the training set for the model.

From the view of data analysis, our methods of converting 1D time series into 2D images can be seen as a kind of feature engineering, which is widely used in the field of signal processing. For example, the discrete cosine transform converts data from time domain to frequency domain and it is commonly used in many traditional and deep learning areas, such as computer vision (e.g., Zhang et al. 2020), voice recognition (Bae et al. 2016), and electroencephalogram signal detection (Rundo et al. 2019). Our approach implicitly includes some prior and feature enhancements from our own visual perception. This provides the direction for the optimization of our algorithm: clearer and more intuitive illustration or presentation of data for humans can yield better results on the computer.

Considering that the observation windows of ground-based telescopes are often affected by weather, season, etc., the sampling interval is more complicated. Our method can be seamlessly extended to the ground-based photometric survey, like ASAS-SN, ZTF, Vera C. Rubin Observatory, etc. In

---

[12] https://www.zooniverse.org/projects/tharinduj/citizen-asas-sn/classify

(a)

(b)
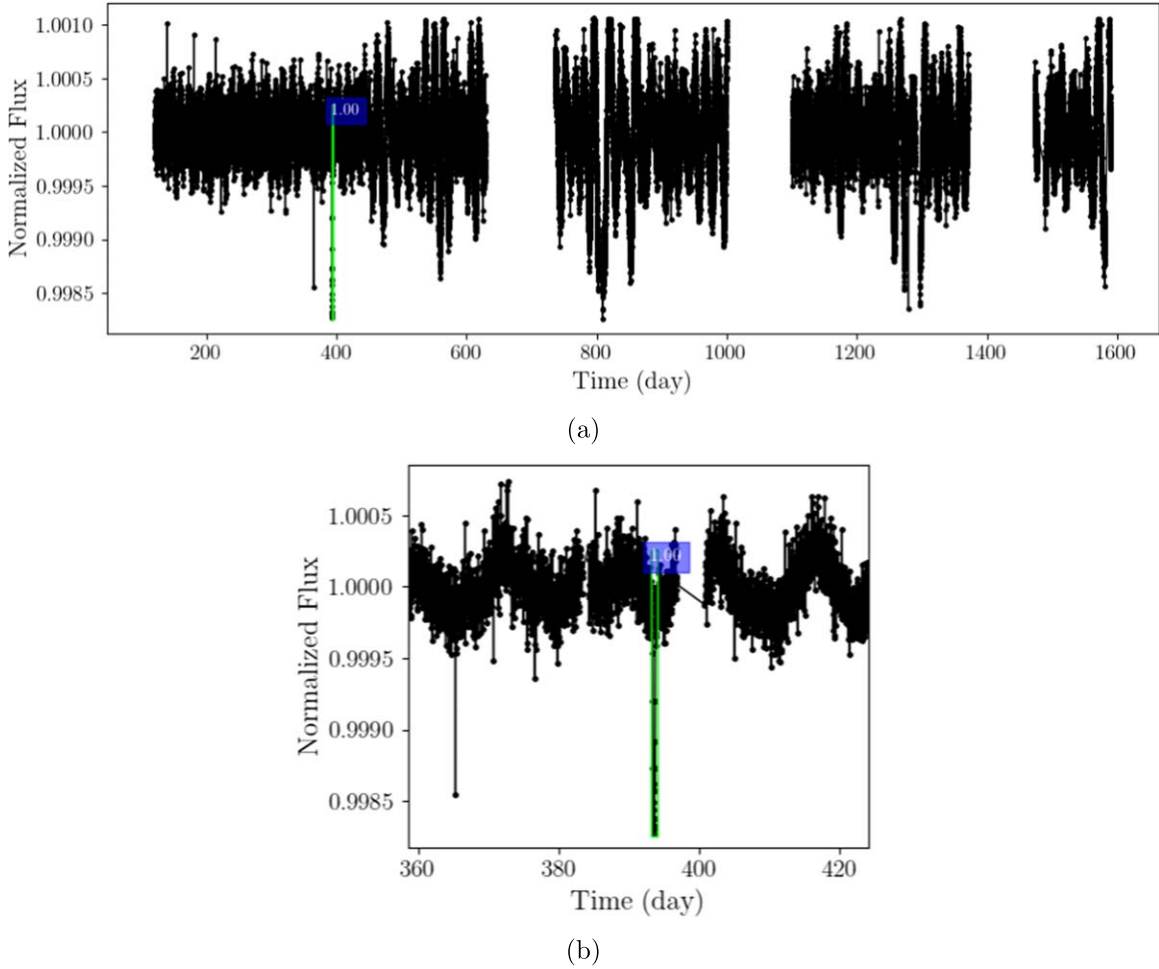
Figure 14. Light curve of KOI-1174. (a) is the 4 yr light curve of KOI-1174. (b) is the zoom-in of the detected region. The green box is the detected bounding box and the confidence score is written in the semitransparent blue box.

particular, multiband photometry can be naturally implemented as the multiple channels of an image, providing richer information.

However, there are three known limitations in our current algorithm.

1. Periodicity determination. Estimate period directly from the neural network is a quite difficult problem. Traditional algorithms like the box-fitting least squares are still more robust.
2. The method of the detrending light curve has a significant impact on the S/Ns of transits. Though Tukey's biweight algorithm performs fairly robust in blind searches, its parameters still need to be adjusted for different types of trends and systematics.
3. Technically, our algorithm only detects transit-like signals, we do not introduce other possible false positives (e.g., eclipsing binaries) in our data set. Considering that many works are using deep learning algorithms to do the transit inspection (e.g., Astronet of Shallue & Vanderburg 2018; Exonet of Ansdell et al. 2018), our results can be the inputs of their networks to classify signals.

We have plans to solve the above difficulties in our future work. The lack of period information, plus the intrinsic and external noises of the light curve, mainly limit the performance of our model for low S/N objects. An alternative solution is to simultaneously include the result of the box-least-squares periodogram and corresponding folded light curve. Some time-series modeling methods such as autoregressive moving average and its extensions (e.g., autoregressive integrated moving average and autoregressive fractionally integrated moving average; Carter & Winn 2009; Feigelson et al. 2018; Caceres et al. 2019) can be helpful for removing stellar variations and systematics.

We will also extend our detection target to more classes of light-curve signals (e.g., rotation, pulsation, and eclipsing binary). By modeling those various types of detected objects, detrending algorithms can be more accurate, and false positives can also be identified. In addition, the data within the output box of the network can be used as the start of further detection and inspection of the signal, resulting in a complete imitation of human operation. However, some blended eclipsing binaries can be hardly discriminated from planetary transits only based on light curves. Usually, these contaminations are removed with some pixel-level inspection. Thus, we are trying to add some pixel vetting modules in our network, and they can be naturally applied in our 2D network. With the development of the computer vision field, more advanced algorithms and structures (e.g., YOLOv5 and Transformer) can be a promising choice in our future work.

9

ORCID iDs

Kaiming Cui ● https://orcid.org/0000-0003-1535-5587
Fabo Feng ● https://orcid.org/0000-0001-6039-0555

References

Ansdell, M., Ioannou, Y., Osborn, H. P., et al. 2018, ApJL, 869, L7
Astropy Collaboration, Robitaille, T. P., Tollerud, E. J., et al. 2013, A&A, 558, A33
Astropy Collaboration, Price-Whelan, A. M., Sipőcz, B. M., et al. 2018, AJ, 156, 123
Bae, H.-S., Lee, H.-J., & Lee, S.-G. 2016, in 2016 IEEE 11th Conf. Industrial Electronics and Applications (ICIEA) (New York: IEEE), 1542
Batalha, N. M., Rowe, J. F., Bryson, S. T., et al. 2013, ApJS, 204, 24
Caceres, G. A., Feigelson, E. D., Jogesh Babu, G., et al. 2019, AJ, 158, 58
Carter, J. A., & Winn, J. N. 2009, ApJ, 704, 51
Charnock, T., & Moss, A. 2017, ApJL, 837, L28
Chaushev, A., Raynard, L., Goad, M. R., et al. 2019, MNRAS, 488, 5232
Choi, J., Chun, D., Kim, H., & Lee, H.-J. 2019, in Proc. IEEE/CVF Int. Conf. on Computer Vision (New York: IEEE), 502
Cooke, B. F., Pollacco, D., West, R., McCormac, J., & Wheatley, P. J. 2018, A&A, 619, A175
Dattilo, A., Vanderburg, A., Shallue, C. J., et al. 2019, AJ, 157, 169
Eisner, N. L., Barragán, O., Lintott, C., et al. 2021, MNRAS, 501, 4669
Feigelson, E. D., Babu, G. J., & Caceres, G. A. 2018, FrP, 6, 80
Feinstein, A. D., Montet, B. T., Ansdell, M., et al. 2020, AJ, 160, 219
Fischer, D. A., Schwamb, M. E., Schawinski, K., et al. 2012, MNRAS, 419, 2900
Foreman-Mackey, D., Morton, T. D., Hogg, D. W., Agol, E., & Schölkopf, B. 2016, AJ, 152, 206
Girshick, R., Donahue, J., Darrell, T., & Malik, J. 2014, in 2014 IEEE Conf. on Computer Vision and Pattern Recognition (New York: IEEE), 580

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., et al. 2014, Adv. Neural Inf. Process. Syst., 27, 2672
Guerrero, N. M., Seager, S., Huang, C. X., et al. 2021, ApJS, 254, 39
Harris, C. R., Millman, K. J., van der Walt, S. J., et al. 2020, Natur, 585, 357
Hinners, T. A., Tat, K., & Thorp, R. 2018, AJ, 156, 7
Hippke, M., & Heller, R. 2019, A&A, 623, A39
Hochreiter, S., & Schmidhuber, J. 1997, Neural Comput., 9, 1735
Huang, X., Bakos, G. Á., & Hartman, J. D. 2013, MNRAS, 429, 2001
Hunter, J. D. 2007, CSE, 9, 90
Kawahara, H., & Masuda, K. 2019, AJ, 157, 218
Kluyver, T., Ragan-Kelley, B., Pérez, F., et al. 2016, in Positioning and Power in Academic Publishing: Players, Agents and Agendas, ed. F. Loizides & B. Schmidt (Amsterdam: IOS Press), 87
Kovács, G., Zucker, S., & Mazeh, T. 2002, A&A, 391, 369
Kreidberg, L. 2015, PASP, 127, 1161
Krizhevsky, A., Sutskever, I., & Hinton, G. E. 2012, Adv. Neural Inf. Process. Syst., 25, 1097
Kunimoto, M., Matthews, J. M., Rowe, J. F., & Hoffman, K. 2018, AJ, 155, 43
Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. 1998, Proc. IEEE, 86, 2278
Leung, H. W., & Bovy, J. 2019, MNRAS, 483, 3255
Lightkurve Collaboration, Cardoso, J. V. D. M., Hedges, C., et al. 2018, Lightkurve: Kepler and TESS time series analysis in Python, Astrophysics Source Code Library, ascl:1812.013
Lissauer, J. J., Ragozzine, D., Fabrycky, D. C., et al. 2011, ApJS, 197, 8
Liu, H., Liu, C., Wang, J. T. L., & Wang, H. 2019, ApJ, 877, 121
Maas, A. L., Hannun, A. Y., & Ng, A. Y. 2013, in ICML Workshop on Deep Learning for Audio, Speech, and Language Processing
Mandel, K., & Agol, E. 2002, ApJL, 580, L171
MegEngine Collaboration 2014, MegEngine, https://github.com/MegEngine/MegEngine
Mosteller, F., & Tukey, J. 1977, Data Analysis and Regression: A Second Course in Statistics, Addison-Wesley Series in Behavioral Science (Boston, MA: Addison-Wesley)
Muthukrishna, D., Narayan, G., Mandel, K. S., Biswas, R., & Hložek, R. 2019, PASP, 131, 118002
Olmschenk, G., Ishitani Silva, S., Rau, G., et al. 2021, AJ, 161, 273
Osborn, H. P., Armstrong, D. J., Brown, D. J. A., et al. 2016, MNRAS, 457, 2273
Osborn, H. P., Ansdell, M., Ioannou, Y., et al. 2020, A&A, 633, A53
Pan, S. J., & Yang, Q. 2009, IEEE Trans. Knowl. Data Eng., 22, 1345
Paszke, A., Gross, S., Massa, F., et al. 2019, Adv. Neural Inf. Process. Syst., 32, 8024
Pearson, K. A., Palafox, L., & Griffith, C. A. 2018, MNRAS, 474, 478
Redmon, J., & Farhadi, A. 2018, arXiv:1804.02767
Ren, S., He, K., Girshick, R., & Sun, J. 2017, ITPAM, 39, 1137
Rosenfeld, A., & Thurston, M. 1971, ITCmp, C-20, 562
Rowe, J. F., Bryson, S. T., Marcy, G. W., et al. 2014, ApJ, 784, 45
Rumelhart, D. E., Hinton, G. E., & Williams, R. J. 1985, Learning Internal Representations by Error Propagation, Technical Report, Univ. of California San Diego
Rundo, F., Rinella, S., Massimino, S., et al. 2019, Comput., 7, 13
Schanche, N., Collier Cameron, A., Hébrard, G., et al. 2019, MNRAS, 483, 5534
Schawinski, K., Zhang, C., Zhang, H., Fowler, L., & Santhanam, G. K. 2017, MNRAS, 467, L110
Shallue, C. J., & Vanderburg, A. 2018, AJ, 155, 94
Tian, Y., Yang, G., Wang, Z., et al. 2019, Comput. Electron. Agric., 157, 417
Twicken, J. D., Jenkins, J. M., Seader, S. E., et al. 2016, AJ, 152, 158
Uehara, S., Kawahara, H., Masuda, K., Yamada, S., & Aizawa, M. 2016, ApJ, 822, 2
Villanueva, S. J., Dragomir, D., & Gaudi, B. S. 2019, AJ, 157, 84
Yao, X., Pepper, J., Gaudi, B. S., et al. 2021, AJ, 161, 124
Yu, L., Vanderburg, A., Huang, C., et al. 2019, AJ, 158, 25
Yurtsever, E., Lambert, J., Carballo, A., & Takeda, K. 2020, IEEE Access, 8, 58443
Zhang, J., Liao, Y., Zhu, X., Wang, H., & Ding, J. 2020, ISPL, 27, 276
Zucker, S., & Giryes, R. 2018, AJ, 155, 147