The Astronomical Journal, 159:109 (9pp), 2020 March https://doi.org/10.3847/1538-3881/ab6aa7

Detrending Exoplanetary Transit Light Curves with Long Short-term Memory Networks

Mario Morvan, Nikolaos Nikolaou, Angelos Tsiaras, and Ingo P. Waldmann
Department of Physics and Astronomy, University College London, Gower Street, London, WC1E 6BT, UK; mario.morvan.18@ucl.ac.uk, n.nikolaou@ucl.ac.uk, angelos.tsiaras.14@ucl.ac.uk, ingo.star@ucl.ac.uk
Received 2019 October 30; revised 2019 December 13; accepted 2020 January 7; published 2020 February 17

Abstract

The precise derivation of transit depths from transit light curves is a key component for measuring exoplanet transit spectra, and henceforth for the study of exoplanet atmospheres. However, it is still deeply affected by various kinds of systematic errors and noise. In this paper we propose a new detrending method by reconstructing the stellar flux baseline during transit time. We train a probabilistic long short-term memory (LSTM) network to predict the next data point of the light curve during the out-of-transit, and use this model to reconstruct a transit-free light curve—i.e., including only the systematics—during the in-transit. By making no assumption about the instrument, and using only the transit ephemeris, this provides a general way to correct the systematics and perform a subsequent transit fit. The name of the proposed model is TLCD-LSTM, standing for transit light-curve detrending-LSTM. Here we present the first results on data from six transit observations of HD 189733b with the IRAC camera on board the *Spitzer Space Telescope*, and discuss some of its possible further applications.

Unified Astronomy Thesaurus concepts: Exoplanet atmospheres (487); Photometric systems (1233); Astronomy data analysis (1858); Extrasolar gas giants (509); Neural networks (1933)

1. Introduction

Since the first exoplanet atmosphere observation 20 years ago (Charbonneau et al. 2000), more than 3000 transiting extrasolar planets have been discovered. Transit spectroscopy—i.e., multiwavelength transit observations—has opened the way for the characterization of atmospheric content and properties of exoplanets. In effect, this can be done by first reconstructing the transmission or emission spectrum from the transit depth measurements at various wavelengths, and at a typical precision level of just a few parts per million (ppm) for hot gaseous planets. This is to be contrasted with the imprints left in the stellar light curve by various instrumental and astrophysical effects that make the measurement of the transit depths extremely challenging. Given the shift of the field toward increasingly smaller planets, the need for efficient detrending methods is thus ever growing. Here we present a long short-term memory (LSTM) neural network approach to effectively model and detrend instrument and astrophysical systematics in-transit light curves.

The total flux $F(t)$ received by a detector at time t can be broken down as follows:

1. star flux: $F_s(t)$;
2. planetary signal: $\delta(t) = (R_P(t)/R_S)^2$ in the case of primary transit obstruction with no limb darkening, where R_S is the stellar radius and R_P the apparent planetary radius;
3. background stars and transient events: $F_b(t)$; and
4. noise and instrumental systematics: $G(.)$.

The total flux received by each pixel of the detector can then be written as $F(t) = G(1 - \delta(t))F_s(t) + F_b(t)$, where F_s and F_b may vary depending on the position on the detector and are then subject to instrumental systematics. We will refer to individual pixel time series as pixel light curves, and to the summed contribution of pixels over time as a raw light curve.

Essentially, the main instrumental systematics trend observed both with the *Hubble Space Telescope* WFC3 and the *Spitzer* IRAC cameras are the so-called ramp effect (Knutson et al. 2007), hypothesized to be due to the charge trapping in the detector (Agol et al. 2010), and intra-pixel and inter-pixel variations, which are correlated with the position of the source on the detector that shows variations in quantum efficiency across different pixels.[1]

Footprints of these entangled variability sources can be found in additional instrumental data collected besides the detector raw flux. In particular, the center and scale of the stellar point-spread function (PSF) can be processed to give valuable information on the systematics while being mostly uncorrelated with the planetary signal itself.

Considering the analysis of time-correlated light curves with the end goal of detrending transit light curves and extracting the transit parameters as precisely as possible, one can approach the problem in several ways. Indeed, the disentanglement of various independent signals might naturally guide one toward blind source separation techniques, which have been applied on this problem (Waldmann 2012; Morello et al. 2014, 2016) using the pixel light curves as correlated components. In a complementary way, signal processing analysis techniques have also been used to denoise the raw or pixel light curves, with Gaussian processes (GPs; Gibson et al. 2012), pixel level decorrelation (Deming et al. 2015), or wavelet analysis (Carter & Winn 2009; Thatte et al. 2010; Morello et al. 2016). Here we choose the angle of interpolation, i.e., we want to provide predictions for the raw light curve during the transit time provided the out-of-transit parts of the light curves. The interpolation method we propose is nonlinear and thus capable of capturing complex long-term dependencies in the light curve.

The use of artificial neural networks (ANNs) is burgeoning in various fields including astronomy. In particular, Charnock & Moss (2017) presented one of the first uses of recurrent neural networks (RNNs) in astronomy for supernovae classification. Yet, in the subfield of exoplanetary sciences, only a

[1] This effect has been described in the IRAC instrument handbook: http://irsa.ipac.caltech.edu/data/SPITZER/docs/irac.

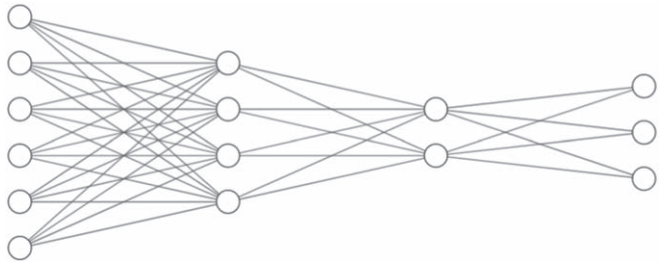

Figure 1. Example of a feed-forward neural network with two hidden layers. When evaluating the underlying function h, the information is flowing from the input on the left toward the output layer on the right.

few studies have been using ANNs so far, with namely Hinners et al. (2018) who predicted stellar and planetary parameters from *Kepler* light curves using RNNs and representation learning, Zingales & Waldmann (2018) on exoplanetary spectra retrieval, Shallue & Vanderburg (2018), Ansdell et al. (2018), and Osborn et al. (2020) for the supervised classification of transit candidates, and Gomez Gonzalez et al. (2018) and Yip et al. (2019) for planet detection in direct imaging.

Here we make use of an LSTM neural network (Hochreiter & Schmidhuber 1997) to interpolate the flux of a raw light curve during the transit, given additional time-series data coming from the PSF centroid. The LSTM network learns to predict the next value of the light curve at each timestep. The predictions of future timesteps are then performed in a probabilistic manner using ancestral sampling, i.e., by injecting the current prediction as input to the subsequent prediction and so on. We thus assume that the pre-transit and post-transit information, along with additional data such as centroid time series, are sufficient to predict the flux that the detector would have received in the absence of a planet transit.

This paper is organized as follows. Section 2 contains background information about neural networks, Section 3 presents the interpolating model and how it can be used for transit light-curve fitting, and finally Section 4 is dedicated to an application on *Spitzer* data.

2. Recurrent Networks and LSTMs

In a typical supervised statistical learning task, the goal is to learn a model $h(x) \simeq y$ that maps an input x to an output y[2] given examples of pairs (x, y) in such a way that the expected error of future predictions is minimized.

Feed-forward neural networks or multilayer perceptrons (MLPs) represent the simplest architecture of deep neural networks. An example of this type of architecture is shown on Figure 1. No feedback connections exist in these models. Every layer consists of a set of neurons and the neurons of the input layer represent each of the original input variables x. The output of each neuron is a scalar value and is used as input for the neurons of the next layer. Each subsequent layer transforms a linear combination of the outputs of the neurons of the previous layer using an activation function σ: $h_{l+1} = \sigma(W_l h_l + b_l)$, where W_l is a matrix of multiplicative weights, b_l the bias vector, h_l the vector of units, and σ_l the activation function, all at layer l. If we interchangeably write h_l for the function represented at layer l as well as its output, the full function represented by a feed-forward network can then be written as $y = h_D(h_{D-1}(...h_1(X)))$ where D is the depth of the network. Note that the nonlinearity of at least one of the layer's activation functions is key to obtaining a nonlinear predictor.

[2] Note that x and y can be scalars, or more generally n-dimensional vectors.

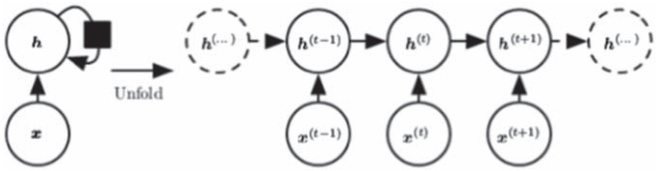

Figure 2. Example of an RNN with no output from Goodfellow & Aaron Courville (2016).

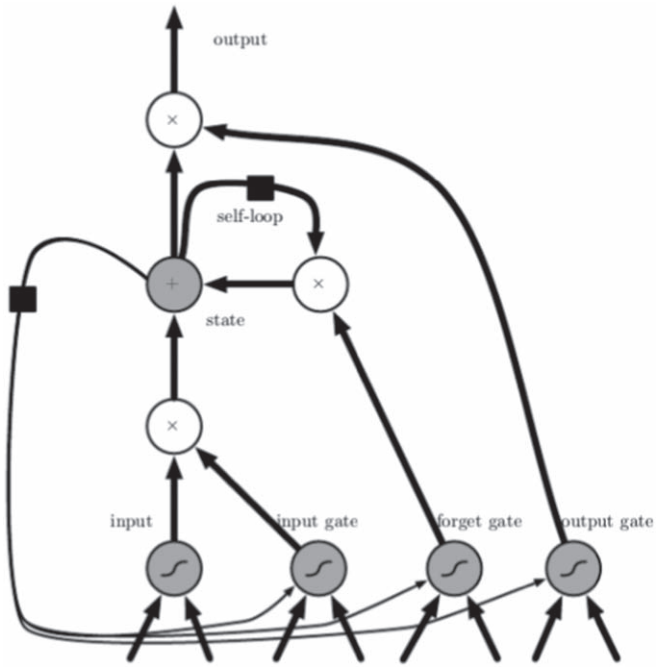

Figure 3. LSTM cell from Goodfellow & Aaron Courville (2016), which replaces a usual hidden unit (i.e., neuron) in a feed-forward neural network. The input, forget, and output gating units enable the cell to accumulate or shut off, respectively, the current input, long-term dependencies, and output through a sigmoidal activation function. The square here indicates a delay of one timestep, and operation symbols in the circles indicate the operation involving the gates' outputs.

The main characteristic of RNNs is that they allow for recurrent connections.[3] If we consider an input sequence $\{x_1, x_2 ...\}$ of vectors, a recurrent hidden layer will thus process it sequentially, receiving at step t both the input x_t as well as other previous hidden state(s) in order to compute the current state h_t. A typical example is shown on Figure 2, where the recurrence occurs between the hidden units of the same layer: $h_t = h_t(x_t, h_{t-1})$. Compared to MLPs, RNNs allow us to reduce the number of parameters of the network by sharing weights between timesteps while seeking temporal patterns in the data.

In practice, several more sophisticated recurrent architectures are often more effective than basic RNNs, with most being variants of the LSTM (Hochreiter & Schmidhuber 1997) architecture whose cell is shown in Figure 3. LSTM networks have proven successful in a large range of applications including unconstrained handwriting recognition (Graves et al. 2009), speech recognition (Graves et al. 2013), and machine translation (Sutskever et al. 2014), to cite only a few.

[3] This means that—unlike in feed-forward neural networks—in RNNs the output of neurons from one layer can be used as input for neurons of the same or a previous layer.

An LSTM cell contains four different gates (see Figure 3), allowing the network to either retain or forget information from the past of the input sequence. This enables the relevant long-term time dependencies to be picked up more easily. The main addition in LSTMs compared to the basic RNNs has been to introduce self-loops, which are conditioned on the context and controlled by the gates. Below we state the detailed update formulae for the gates and states composing each LSTM unit:

1. the input gate: $i_t = W_{ix}x_t + W_{ih}h_{t-1} + b_i$;
2. the forget gate: $f_t = W_{fx}x_t + W_{fh}h_{t-1} + b_f$;
3. the output gate: $o_t = W_{ox}x_t + W_{oh}h_{t-1} + b_o$;
4. the cell state: $c_t = \sigma(f_t) \odot c_{t-1} + \sigma(i_t) \odot \tanh(j_t)$; and
5. the output vector: $h_t = \sigma(o_t) \odot \tanh(c_t)$.

Where t denotes the timestep, W_{ab} the matrix of weights relative to the vectors a and b, b_a the bias vector relative to a, $\odot$ the Hadamart (i.e., entrywise) product, and σ is the activation function, typically a logistic sigmoid or tanh function.

Incidentally, these types of gated RNNs also have the advantage of being easier to train than basic RNNs, by alleviating the well-known vanishing or exploding gradient issue[4] (Kolen & Kremer 2001).

3. TLCD-LSTM

Here we describe the proposed model to interpolate a time series on a predefined prediction range. As the final goal of this paper is to study the transit signal contained in the interpolation range after correction of the systematic errors, we name the method transit light-curve detrending-LSTM (TLCD-LSTM).

The model is based on the deep autoregressive neural network model described in Salinas et al. (2017). It assumes that temporal relations exist in the time series and learns to predict the next step in the training range of the input time series. It can also make use of additional data available for prediction contained in the so-called covariate time series, which is to be distinguished from the main time series. In general, one can consider both the main and covariate time series to be multivariate, i.e., to be composed of several time series each.

TLCD-LSTM is specifically adapted for interpolation within a given range, and therefore differs from Salinas et al. (2017) mainly in that the values it tries to predict are not in the future (i.e., the end of the time series) but in timesteps somewhere within the time series.

3.1. Model Description

Let us denote with $\{x_1, x_2, .., x_T\}$ (abbreviated $\{x_t\}$) the main time series of length T we ought to interpolate on the prediction range $[t_1 .. t_2]$ with t_1 and t_2 integers in $[1..T]$, and $\{z_1, z_2, .., z_T\}$ (abbreviated $\{z_t\}$) the time series of covariates, which constitute additional data available for prediction on the whole time range. Finally, let us also denote with $\{y_1, y_2, .., y_T\}$ (abbreviated $\{y_t\}$) the target time series, identical to the main time series in the training range but which may differ in the prediction range. In the case of $\{x_t\}$ being a transit light curve, $\{y_t\}$ is the hypothetical light curve without any transit signal.

[4] Neural networks are trained via gradient-based minimization of a loss function. In each iteration of training, each parameter of the model (weight) receives an update proportional to the partial derivative of the loss with respect to the current weight. Allowing these gradients to grow vanishingly small or too large can cause numerical instabilities, slow down training, or stop it prematurely.

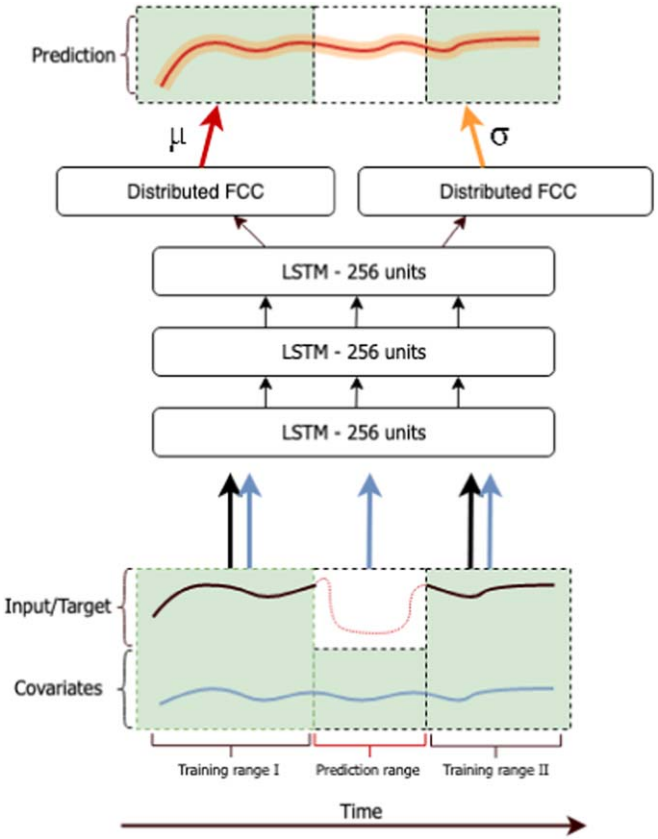

Figure 4. Sketch of the interpolating probabilistic LSTM neural network. The main and covariate time series are processed through three LSTM layers consisting of 256 units each, and then decoded into two outputs for each of the interpolated points: the mean and the standard deviation.

As sketched in Figure 4, each value of the input time series passes through a stack of LSTM layers, the output of which branches into two distinct feed-forward layers outputting two parameters $\widehat{\mu_t}$ and $\widehat{\sigma_t}$ at each timestep, which are the predicted mean and standard deviation for the distribution of the current value x_t, respectively. The details and hyperparameters of the architecture are presented in Appendix C. The same network is used both for the training and prediction ranges with only the inputs differing in each case.

At each timestep t, the network predicts the current value x_t from all past timesteps x_1,.., x_{t-1} as well as from the current covariate z_t. While the actual previous time-series value x_{t-1} is used as input in the training ranges, in the prediction range the previous prediction μ_{t-1} is injected as an input instead of it (see Table 1).

3.1.1. Training the Model

We assume each value z_t is sampled from a normal distribution:

$$z_t \sim \mathcal{N}(\mu, \sigma^2).$$

The loss function is then computed as the product of individual likelihoods outside the prediction range of

$$\mathcal{L}(\mu_{t_1..t_2}) = \prod_{t \in [1..t_1] \bigcup [t_2..T]} \frac{e^{(x_t - \widehat{\mu}_t)^2/2\widehat{\sigma}_t}}{\sqrt{2\pi\sigma_t}}.$$

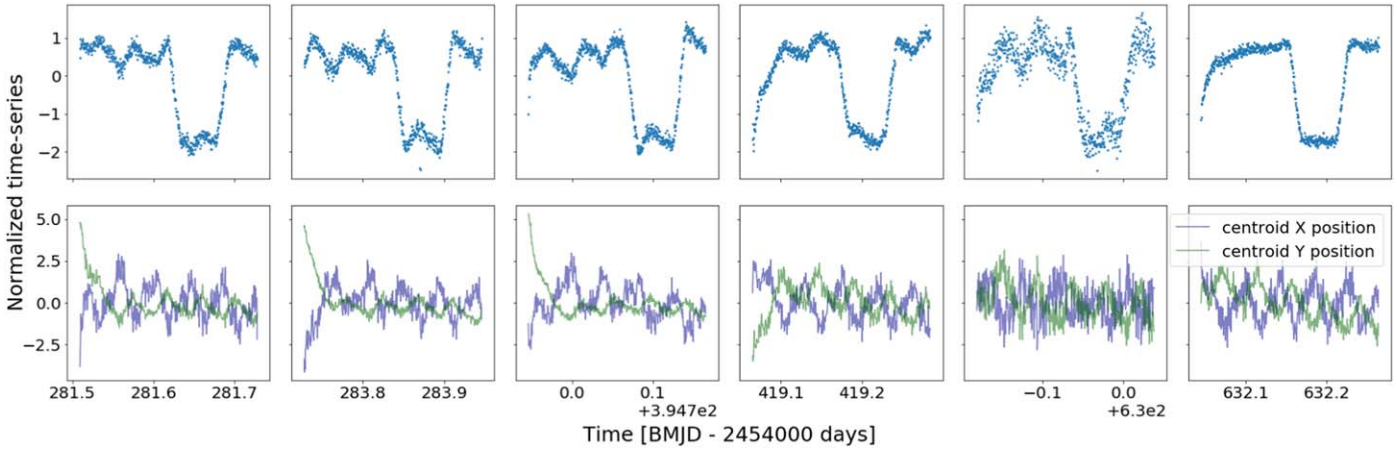

Figure 5. Top: six *Spitzer*/IRAC 8μm raw transit light curves of HD 189733b after preprocessing. Bottom: X/Y centroid positions of the PSF.

Table 1
Differences of Inputs at Each Timestep t between the Training and Prediction Ranges

Mode	Range	Inputs at t	Output at t
Training	$[1\,..\,t_1] \cup [t_2\,..\,T]$	x_{t-1}, z_t	$(\widehat{\mu_t}, \widehat{\sigma_t})$
Prediction	$[t_1..t_2]$	$\widehat{\mu_{t-1}}$, z_t	$(\widehat{\mu_t}, \widehat{\sigma_t})$

Note that the log-loss is only computed in the training ranges. However, the last output of the prediction range is taken as the first input of the second training range, thus providing a way to link together the outputs in the different ranges.

3.1.2. Predicting the Time Series

There are several ways one can generate predictions in $[t_1..t_2]$ once a model is trained. Since the inputs of the network consist of parameters of a probability distribution, the simplest one is to directly take the vector of predicted means $\widehat{y_t} = \mu_t$. However, one can also generate a trace by drawing every value from the Gaussian distribution at every timestep in the prediction range $\widehat{y_t} \sim \mathcal{N}(\mu_t, \sigma_t^2)$, and injecting each of these predictions as input for the next timestep. Multiple traces obtained with this process then represent the joint predicted distribution (of which they are samples) in a more general way than merely using the means vector. To generate a single vector of predictions from multiple traces, one can—for instance—select the median or mean value at every timestep to construct the median trace or mean trace on the prediction range. In Section 4, we focus on the simplest approach, i.e., selecting the output means and standard deviations.

3.1.3. Covariant Features

The covariates time series $\{z_t\}$ can consist of single-dimensional or multi-dimensional data available both in the training and prediction ranges. It is used by the network as additional information besides the target time series. This works merely by concatenating x_t (conversely $\widehat{x_t}$ in prediction mode) to the covariate data z_t to construct the new input to the network at every timestep. Ideally, one wants $\{z_t\}$ to be correlated with the target time series. Several time series might be related to the time-correlated noise we intend to correct, and therefore can be used as covariate data in the model. In the application presented in Section 4 we suggest the use of PSF-related time series, namely the instrument's PSF centers and widths of a two-dimensional Gaussian fit on the images at every timestep. One could also think of other potentially relevant information such as simultaneous host star activity, calibration data relative to the detector, and estimations of background flux. For ground-based applications, information about airmass, seeing, and weather patterns could be included.

3.2. Application to Transit Light Curves

Here we discuss the use of the interpolating model specifically to transit light curves.

The transit signal must be contained within the prediction range. This requires either to know beforehand when the transit occurs, or to adapt the prediction range during the first phase of the training. Pre-transit and post-transit data are used for training the network, and are assumed to not be contaminated by any transit event. They can however contain any sort of variability coming from the star, the background, or the instrument. In fact, the model aims at picking up variations due to all sources other than a transit event in order to predict the flux due to these sources alone during the transit time.

We perform a transit fit at each evaluation step even though our model does not strictly require it for the training. This is done for two main reasons. (1) The transit fit can be used as a proxy to evaluate the quality of the prediction and provide us with a criterion for stopping the training of our model early. The transit fit is performed on the detrended light curve normalized with respect to the star $(1 - \delta_t)$. For details, see Appendix A. (2) We can use the transit fit to adapt the prediction range $[t_1..t_2]$ during training so that it better matches the actual transit range of the data. This can be done by extracting the fitted mid-transit time and transit duration to compute the times for the beginning and end of transit.

4. Application

We present an application to six transit observations of planet HD 189733b from the *Spitzer*/IRAC detector at 8 μm, collected in 2007 and 2008 (PI: E. Agol, program: 40238), and shown in Figure 5. This hot-Jupiter planet has been extensively studied and makes a good candidate for benchmarking our method. In this wavelength channel, the ramp effect can be heavily pronounced (Agol et al. 2010), while the intra-pixel variations due to pointing jitter are less important than at

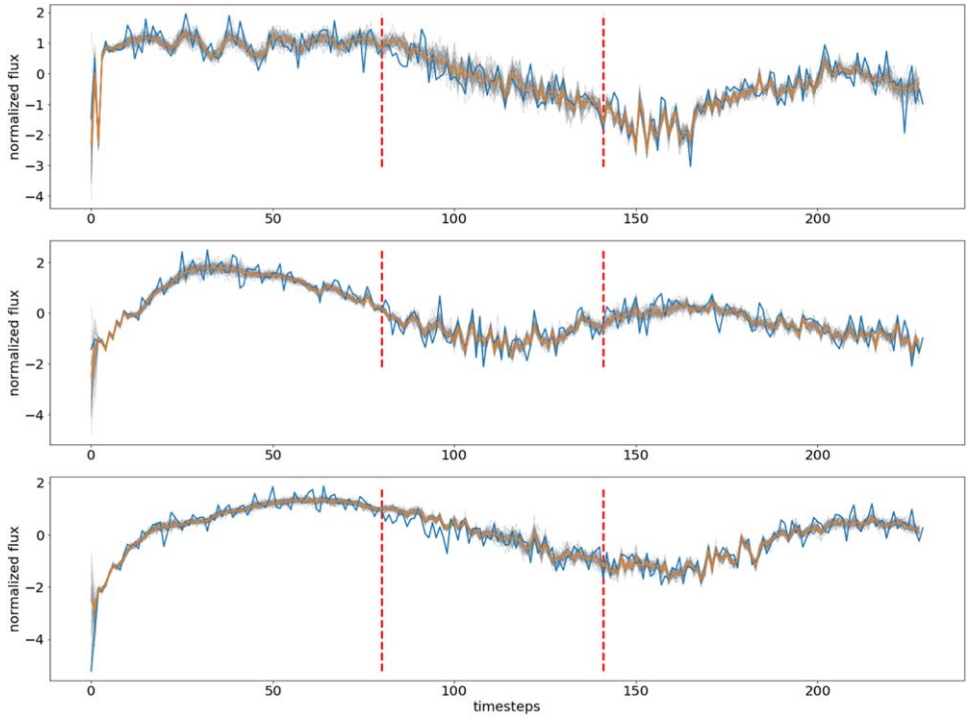

Figure 6. Example of interpolations on three light curves containing no transit. The prediction range is located inside the vertical dashed lines. The raw light curve is displayed in blue, the predicted traces in gray, and the median prediction in orange.

shorter wavelengths. A few preprocessing steps are applied to the data[5] and detailed in Appendix B. These include outlier removal, raw light-curve extraction and normalization, centroids fitting, and background light estimation.

In this section, we present predictions on the pre-transit range (on intervals not used in the training of the model) as an initial evaluation of the model, and then show results on the real transit ranges on which we derive the detrended light curve and subsequent transit fit.

4.1. Testing

As the ground truth, i.e., the predicted stellar and instrumental flux, is not available in the transit range, we chose to first test the interpolating model on the pre-transit range instead, where its predictions can be evaluated more directly. In practice, three prediction ranges are selected in the first 250 timesteps of the time series, where no transit signal is present, and the mean squared error (MSE) metric is used to evaluate and compare various models. An example of prediction is shown on Figure 6, where the prediction is obtained by averaging 50 sampled traces.

4.1.1. Hyperparameter Optimization

We perform a grid search over different types of inputs and hyperparameters. More specifically, we vary the aperture width of the subarray used for computing the raw light curve between 5 and 7 pixels; we experiment with including and excluding covariate features, namely: (1) excluding covariate features altogether, (2) including centroid time series, and (3) including centroid and PSF width time series. Furthermore, we vary the number of layers (between 1 and 4), units per layer (powers of 2 up to 1024), and dropout rate (between 0% and 50% in steps of 1%)[6] values for the LSTM block and a unidirectional or bidirectional network.[7] We train each different model on the six light curves and three different prediction ranges, monitoring the average MSE for these 18 predictions and using it as a criterion for early stopping and comparison between the different models.

From these tests we observe the following:

1. including the centroids information improves the quality of the prediction by a factor of ∼2, and including the PSF widths time series besides the centroids brings a further increase in MSE;
2. dropping 3% of the recurrent units improves the predictions slightly, especially when the number of parameters of the network increases; and
3. using a bidirectional network slightly decreases the quality of the predictions.

More information on the hyperparameters and model training used is presented in Appendix C.

4.1.2. Performance

We present in Table 2 the results of the best tested model in the explored grid. As a reference for the performance of the interpolation, we include a baseline model, which is a linear composition of the centroid X/Y time series $\{z_t^X\}$ and $\{z_t^Y\}$,

$$\widehat{y_t} = a + bz_t^X + cz_t^Y,$$

[5] The data used are publicly available and were downloaded from https://sha.ipac.caltech.edu/applications/Spitzer/SHA/.

[6] Dropout is a common regularization technique in deep learning consisting in randomly reinitializing a fraction of the neurons of a given layer. The dropout rate refers to this fraction.

[7] By unidirectional network we mean one that uses just past timesteps to infer the current one. With bidirectional we mean using timesteps from both past and future to infer the current one.

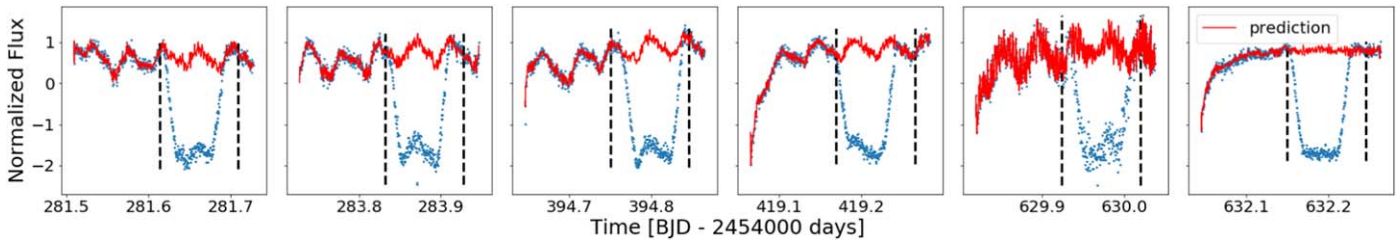

Figure 7. Raw data (blue) and model output, i.e., interpolated light curve in the absence of transit (red) for the light curves. Dashed vertical lines indicate the initial prediction ranges.

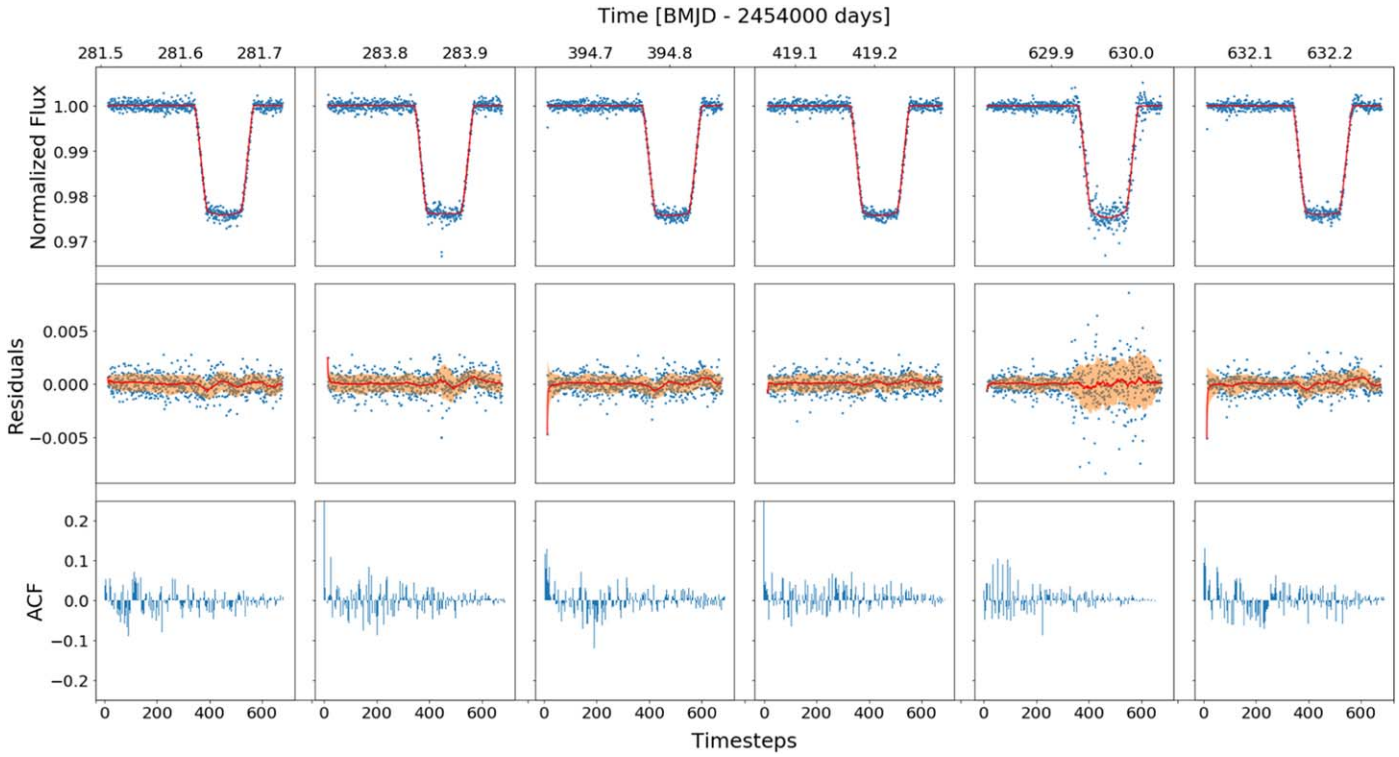

Figure 8. Top: best transit fit (red curve) to the detrended light curve (blue points) normalized with respect to the stellar flux. Center: fit residuals (blue points) along with the moving average (red curve) and standard deviation (orange) of the residuals. Bottom: ACF of the residuals.

where $a, b, c \in \mathbb{R}$. The model is trained on the training ranges[8] and evaluated in the prediction ranges. The metrics computed for both models include the MSE, the mean absolute error (MAE), and the mean signal-to-noise (S/N) ratio, defined as

$$\mathrm{MSE} = \frac{1}{N}\sum_{t_1}^{t_2}(\widehat{y_t} - y_t)^2.$$

$$\mathrm{MAE} = \frac{1}{N}\sum_{t_1}^{t_2}|\widehat{y_t} - y_t|.$$

$$\mathrm{S/N} = \frac{1}{N}\sum_{t_1}^{t_2}|\widehat{y_t} - y_t|/\sigma_{\mathrm{noise}},$$

where N is the number of observations, and σ_{noise} is an estimate of the noise level computed by taking the mean value of the running standard deviation of width 15 over each input light curve.

[8] The model was fitted using scikit-learn's linear regression module: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html.

Table 2
Comparison of the Performance on the Six Light Curves for Each Model

LC Instance	Baseline	This Model
MSE #1	0.192	0.156
MSE #2	0.782	0.196
SEE #3	0.606	0.138
MSE #4	0.0921	0.0529
MSE #5	0.275	0.249
MSE #6	0.688	0.0837
Mean MSE	0.439	0.124
Mean MAE	0.503	0.367
Mean S/N	1.430	0.807

Note. Every value is averaged between the prediction and actual value over three different ranges of length 60 and starting, respectively, at timeseteps 80, 100, and 120. The three last lines show the mean performance over all light curves and ranges in terms of MSE, MAE, and S/N.

Given its simplicity, this baseline model does a remarkably good job at interpolating $\{x_t\}$, and this is why it was chosen here as a reference for the MSE. Furthermore, since

Table 3
Fitted Physical Parameters for Each of the Six Transits

t_c (BJD-2454,000)	R_P/R_S	i (deg)	a/R_S	u
281.655329 ± 0.000046	0.15489 ± 0.00018	85.7682 ± 0.0502	8.971 ± 0.045	0.141 ± 0.020
283.873934 ± 0.000049	0.15477 ± 0.00024	85.7277 ± 0.0698	8.901 ± 0.069	0.051 ± 0.018
394.802829 ± 0.000045	0.15564 ± 0.00020	85.5926 ± 0.0771	8.799 ± 0.065	0.093 ± 0.037
419.206955 ± 0.000070	0.15520 ± 0.00015	85.8120 ± 0.0881	8.992 ± 0.075	0.129 ± 0.026
629.971770 ± 0.000097	0.15523 ± 0.00042	85.9760 ± 0.1263	8.999 ± 0.106	0.248 ± 0.028
632.190498 ± 0.000046	0.15488 ± 0.00019	85.5862 ± 0.0747	8.782 ± 0.057	0.097 ± 0.028

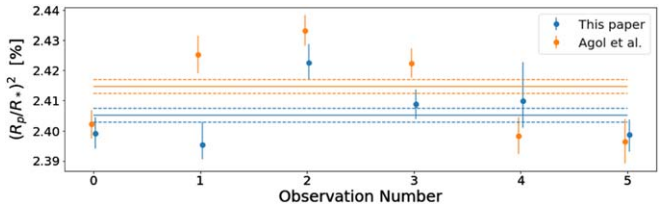

Figure 9. Comparison of fitted transit depths between this work and Agol et al. (2010), for the six transit observations of HD 189733b. The horizontal lines show the means of the observations from both papers weighted by their respective standard deviations. The dotted lines show the standard deviations of these weighted means.

the TLCD-LSTM also uses the centroid time series, the increase in performance seen on Table 2 can directly be interpreted as the improvement brought by the LSTM's ability to identify temporal dependencies in the raw light curve.

4.2. Prediction on Real Transit Ranges

Using the optimized hyperparameters listed in Section 4.1 and after training the model for 3000 epochs, we extract the output of the network for the whole time ranges, shown in red on Figure 7. Note that the decreasing learning rate used guarantees the convergence of the network toward a stable solution. Visually, the model seems to be able to pick up the trends and variability of each time series, while joining smoothly the pre- and post-transit ranges where the ground truth is known.

The last step is to perform the transit fit on the detrended light curve $\{1 - \delta_t\}$, normalized with respect to the stellar flux $F_s(t)$. Since the limb-darkening effect is minor at 8 μm, we chose a transit model with linear limb darkening (bound between 0.05 and 0.25), and compute the best fit using a Markov Chain Monte Carlo optimization procedure[9] (Tsiaras et al. 2016). The fitted parameters are R_p/R_s, the mid-transit time t_c, a linear limb-darkening coefficient u, the orbit inclination i, and orbital semimajor axis relative to the stellar radius a/R_s. The fitted model, residuals, and autocorrelated functions (ACF) are shown in Figure 8 and the fitted parameters are presented in Table 3. The higher variance present in the residuals of the fifth light curve is due to a higher noise level in the input data for this light curve.

We compare the retrieved transit depths with the results published in Agol et al. (2010) for the same data set and preprocessing steps (Figure 9). Although slightly smaller, the scatter of the predictions is still present with a standard deviation of 91.7 ppm instead of 144 ppm. The mean weight by the standard deviations of the six transit depths is also found to be slightly smaller in our case by 94 ppm $\approx 4\sigma$.

[9] The transit model was fitted using the PylightCurve package: https://github.com/ucl-exoplanets/pylightcurve.

5. Discussion and Conclusions

We presented a deep learning model suitable for interpolating time series, and showed how it can be used to predict the variability of stellar light curves for subsequent transit fits. This approach has the advantage of not making any assumption on the types of noise, systematics, or transit shape.

The presented method is similar to the GP approach (Rasmussen & Williams 2005; Gibson et al. 2012) in that they both construct highly nonlinear models, avoid explicit physical modeling of the systematics, and provide probabilistic predictions. However, they differ in various aspects. (1) The neural network light-curve interpolation approach we propose does not require any transit model in the first hand, whereas it is included in the mean function of the GP. This makes the TLCD-LSTM approach generally more applicable as it does not depend on pre-chosen mean and kernel functions. (2) The GP approach requires fewer parameters to train and provide fully Bayesian predictions compared to our LSTM-based approach. The smaller number of free parameters may make GPs the preferred choice for short time series. However, GPs computation scales more poorly with the number of data points, preventing them to be applicable to data sets of more than ≈1000 timesteps without binning of the time series.[10] The proposed interpolating LSTM can, on the other hand, be applied to longer or multiple light curves as commonly found in *Kepler* and *Transiting Exoplanet Survey Satellite* (*TESS*) time series allowing for even very long-period variability to be captured in the predictive LSTM model. This is because the computational complexity in the case of GPs mainly depends on the number of data points, while in the case of the deep neural networks in the architecture chosen (i.e., the number of layers, number of nodes per layer, and type of layers in our case).

While the current implementation still relies on a few preprocessing steps such as computing the raw light curve or centroids fits, it constitutes a first step toward the ultimate goal of developing an end-to-end detrending pipeline where the input would be the raw pixel light curves or focal plane images. Furthermore, while we trained our network on data from six real light curves only, taking advantage of a large number of light curves, real or simulated, would allow for the development of a more general detrending approach for each

[10] Although more scalable GPs implementations have recently been proposed, hence allowing for processing increasingly longer time series (see for instance Foreman-Mackey et al. (2017) and Farr et al. (2018) for fast GP implementation in astronomical time series).

instrument. LSTMs allow for efficient transfer learning between data sets and instruments (e.g., *Kepler* to *TESS*). This may become important in modeling common systematics such as stellar noise between planet–star systems observed by multiple instruments.

As we have firmly entered the era of big data in planet detection (e.g., *Kepler*, *TESS*, and ground-based surveys) and with upcoming characterization missions and instruments (e.g., the *James Webb Space Telescope*, the Atmospheric Remote-sensing Infrared Exoplanet Large-survey, CHaracterising ExOPlanets Satellite, and the Extremely Large Telescopes), the opportunities for data detrending and modeling with scalable deep learning methods capable of processing large numbers of high-dimensional data will become increasingly prevalent in the future.

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 758892, ExoAI), under the European Union's Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement numbers 617119 (ExoLights) and the European Union's Horizon 2020 COMPET programme (grant agreement No 776403, ExoplANETS A). Furthermore, we acknowledge funding by the Science and Technology Funding Council (STFC) grants: ST/K502406/1, ST/P000282/1, ST/P002153/1, and ST/S002634/1.

Software: The data and code used in this paper are available on GitHub under a Creative Commons Attribution 4.0 International License (https://github.com/ucl-exoplanets/deepARTransit, which is archived on Zenodo (Morvan 2019)), or a MIT License (https://github.com/ucl-exoplanets/pylightcurve).

Appendix A
Transit Fit

To obtain a light curve normalized with respect to the star, three steps are required: transformation to the original units $y_t \rightarrow y_t'$, subtraction of background flux $F_b(t)$, and division of the background subtracted raw light curve by the predicted star flux,

$$1 - \delta(t) = \frac{F_{\text{received}}(t) - F_b(t)}{F_s(t) - F_b(t)}.$$

With the time-series notations, where x_t' and $\hat{y}_t'$ are the input and mean prediction of the neural network in the original units,

$$1 - \delta_t = \frac{x_t' - F_{b,t}}{\hat{y}_t' - F_{b,t}}.$$

Note that during training, we use a simple piecewise-linear transit model with four parameters described in Carter et al. (2008) optimized by least-square fitting and neglect the contribution of the background $F_b \ll x_t', \hat{y}_t'$.

Appendix B
Preprocessing

Here we describe the different preprocessing steps applied to the raw subarray data.

Outlier removal. Due to a number of causes, such as remaining cosmic rays or bad pixels, the flux on individual pixels can exhibit great fluctuations within short timescales (≈1s). These abnormal values are identified by computing the absolute difference of the pixels' flux with their corresponding median within a time window of width 5 (2s exposure). The values of the median-subtracted time series greater than 4σ are then replaced by the median values, where σ is the standard deviation of the time series.

Raw light-curve extraction. In order to limit the influence of background light and focus on the brightest pixels of the stellar PSF, 3×3, 5×5, and 7×7 pixel regions are extracted around the brightest pixel. The raw light curve is then obtained by summing all the individual pixel light curves.

Centroid fitting. As mentioned earlier the centroid position time series are highly correlated with the flux received by the detector. In order to compute the centroids, we perform a two-dimensional Gaussian fit with an offset to the data at every timestep, and hence extract four useful time series, two of which are monitoring the position of the center on the detector and two for the width of the Gaussian. As discussed in Agol et al. (2010), this method provides by far a better estimate of the centroids over other methods such as the flux-weighted ratio extraction.

Background extraction. The background flux contribution to the total flux, although minor, increases with the aperture size used for the light-curve extraction. We estimate it here by taking the median flux value of the pixels located in the four corners of each frame, corners delimited by the complement of a circular aperture of radius 16. It accounts for 0.67%–1.2% in our analysis, and should therefore be taken into account. However, as the background estimation is necessarily approximate, we advocate to still interpolate on the raw light curve directly, and only correct for it before the transit fit.

Normalization. The raw light curve and centroid time series are all locally standardized, i.e., individually centered around a mean value of zero and rescaled to have their standard deviation equal to one.

The preprocessed raw light curves and centroid X/Y positions are shown on Figure 5. Note the diversity of effects among them, showing more or less stochastic noise, ramps, or jitter.

Appendix C
Hyperparameters

Training parameters. The network hyperparameters are shown in Table 4. Training was performed using the Adam optimizer (Kingma & Ba 2014) with parameter values $\beta_1 = 0.9$, $\beta_2 = 0.99$, $\epsilon = 108$. The learning rate was decreased from 0.01 to 0.0001 using a polynomial decay law with exponent 20. We train the model using a batch size of six (all of the light curves) for faster training.

Table 4
Table of the Network Hyperparameters

Parameter	Value
Number of LSTM layers	3
Number of units per layer	256
Recurrent dropout rate	3%
Initial bias values	0.0
Batch size	6

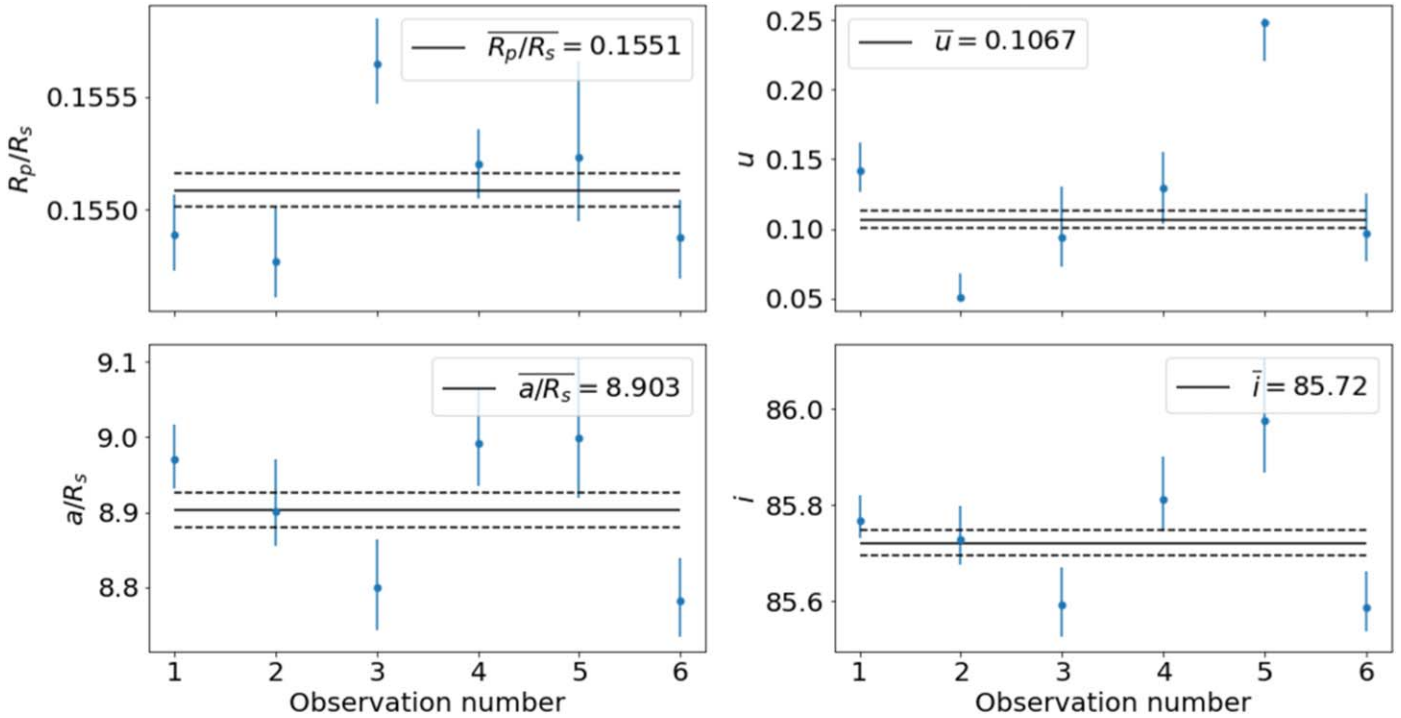

Figure 10. Additional plots showing the fitted parameters R_p/R_s, u, a/R_s, and i for each of the 6six light curves, as well as their weighted mean and associated standard deviation.

Appendix D
Plots of Fitted Transit Parameters

Figure 10 presents the results for the fitted transit parameters R_p/R_s, u, a/R_s and i.

ORCID iDs

Mario Morvan https://orcid.org/0000-0001-8587-2112
Nikolaos Nikolaou https://orcid.org/0000-0001-8453-7574
Angelos Tsiaras https://orcid.org/0000-0003-3840-1793
Ingo P. Waldmann https://orcid.org/0000-0002-4205-5267

References

Agol, E., Cowan, N. B., Knutson, H. A., et al. 2010, ApJ, 721, 1861
Ansdell, M., Ioannou, Y., Osborn, H. P., et al. 2018, ApJL, 869, L7
Carter, J. A., & Winn, J. N. 2009, ApJ, 704, 51
Carter, J. A., Yee, J. C., Eastman, J., Gaudi, B. S., & Winn, J. N. 2008, ApJ, 689, 499
Charbonneau, D., Brown, T. M., Latham, D. W., & Mayor, M. 2000, ApJL, 529, L45
Charnock, T., & Moss, A. 2017, ApJL, 837, L28
Deming, D., Knutson, H., Kammer, J., et al. 2015, ApJ, 805, 132
Farr, W. M., Pope, B. J. S., Davies, G. R., et al. 2018, ApJL, 865, L20
Foreman-Mackey, D., Agol, E., Ambikasaran, S., & Angus, R. 2017, AJ, 154, 220
Gibson, N. P., Aigrain, S., Roberts, S., et al. 2012, MNRAS, 419, 2683
Gomez Gonzalez, C. A., Absil, O., & Van Droogenbroeck, M. 2018, A&A, 613, A71
Goodfellow, I., & Aaron Courville. 2016, Deep Learning (Cambridge, MA: MIT Press), http://www.deeplearningbook.org
Graves, A., Liwicki, M., Fernández, S., et al. 2009, ITPAM, 31, 855
Graves, A., Mohamed, A.-R., & Hinton, G. 2013, arXiv:1303.5778
Hinners, T. A., Tat, K., & Thorp, R. 2018, AJ, 156, 7
Hochreiter, S., & Schmidhuber, J. 1997, Neural Computation, 9, 1735
Kingma, D. P., & Ba, J. 2014, arXiv:1412.6980
Knutson, H. A., Charbonneau, D., Allen, L. E., et al. 2007, Natur, 447, 183
Kolen, J. F., & Kremer, S. C. 2001, A Field Guide to Dynamical Recurrent Networks (Piscataway, NJ: IEEE), https://ieeexplore.ieee.org/document/5264952
Morello, G., Waldmann, I. P., & Tinetti, G. 2016, ApJ, 820, 86
Morello, G., Waldmann, I. P., Tinetti, G., et al. 2014, ApJ, 786, 22
Morvan, M. 2019, DeepARTransit: A Library for Interpolating and Detrending Transit Light Curves with LSTMs,v 1.1, Zenodo, doi:10.5281/zenodo.3574538
Osborn, H. P., Ansdell, M., Ioannou, Y., et al. 2020, A&A, 633, A53
Rasmussen, C. E., & Williams, C. K. I. 2005, Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning) (Cambridge, MA: MIT Press), http://www.gaussianprocess.org/gpml/
Salinas, D., Flunkert, V., & Gasthaus, J. 2017, arXiv:1704.04110
Shallue, C. J., & Vanderburg, A. 2018, AJ, 155, 94
Sutskever, I., Vinyals, O., & Le, Q. V. 2014, arXiv:1409.3215
Thatte, A., Deroo, P., & Swain, M. R. 2010, A&A, 523, A35
Tsiaras, A., Waldmann, I. P., Rocchetto, M., et al. 2016, ApJ, 832, 202
Waldmann, I. P. 2012, ApJ, 747, 12
Yip, K. H., Nikolaou, N., Coronica, P., et al. 2019, arXiv:1904.06155
Zingales, T., & Waldmann, I. P. 2018, AJ, 156, 268