University of Amsterdam

MSc Artificial Intelligence
Master Thesis

Detecting Exoplanet Transit Signals in Light Curves Using Recurrent Neural Networks

---

by
Yke Jan Rusticus
11306386

August 10, 2021

48 ECTS
October 2020 – July 2021

Supervisor:
MSc D. Ruhe

External co-supervisors:
Prof. Dr. B. H. Foing
Dr. A. M. Heras

Assessor:
Dr. P. D. Forré

European Space Agency

[TODO: write Acknowledgements and Abstract]

Contents

1 Introduction **4**
- 1.1 Problem . . . 4
- 1.2 Research aims . . . 4
- 1.3 Common approaches and limitations . . . 5
- 1.4 Proposed method . . . 5
- 1.5 Summary of contributions . . . 6
- 1.6 Outline . . . 6

2 Background **7**
- 2.1 Transit method . . . 7
- 2.2 Exoplanet-hunting missions . . . 8
- 2.3 Challenges in detecting transit signals . . . 9
  - 2.3.1 Stellar variability . . . 9
  - 2.3.2 Instrumental and photon noise . . . 9
  - 2.3.3 Transit signal shapes . . . 9
  - 2.3.4 Weak and sparse signals in big data . . . 9
  - 2.3.5 Astrophysical false positives . . . 10
- 2.4 Transit detection methods in literature . . . 10
  - 2.4.1 Box and Transit Least Squares . . . 10
  - 2.4.2 Simultaneously modelling background and signal . . . 11
  - 2.4.3 Kepler and TESS pipeline . . . 11
  - 2.4.4 Other classical detection methods . . . 12
  - 2.4.5 "Intelligent" algorithms . . . 12
- 2.5 Recurrent neural network (RNN) . . . 13
  - 2.5.1 Theory . . . 13
  - 2.5.2 Applications . . . 14

3 Methodology **16**
- 3.1 Network . . . 16
  - 3.1.1 Architecture . . . 16
  - 3.1.2 Training . . . 16
- 3.2 Detection algorithm . . . 18
  - 3.2.1 PTS-Peak . . . 19
  - 3.2.2 PTS-Fold . . . 19
- 3.3 Handling data gaps . . . 20
- 3.4 Increasing interpretability with confidence estimation . . . 21
- 3.5 Separating signals from different planets . . . 21
- 3.6 Data simulations . . . 22
  - 3.6.1 LCSim . . . 22
  - 3.6.2 Lilith-4 . . . 23

4 Experiments and results **26**
- 4.1 Preprocessing and performance . . . 26
- 4.2 Model comparison . . . 29
- 4.3 Monotransit retrieval . . . 31
- 4.4 Single planet retrieval . . . 32
- 4.5 Multiplanet retrieval . . . 33
- 4.6 Success and failure cases . . . 34

5 Discussion **41**
5.1 Answers to research questions 41
5.2 Limitations and suggestions 42
5.3 Future directions 43

6 Conclusions **45**

References **49**

Appendix **50**
A.1 Conference and workshop contributions 50

Chapter 1

Introduction

The discovery of exoplanets is the first step in answering a broad range of fundamental questions in astronomy. Is our solar system one of a kind? Is there life elsewhere in the universe? Exoplanets, short for extrasolar planets, are planets orbiting stars other than our Sun. The problem of detecting exoplanets is an active region of science and is described in more detail in the first section of this chapter, i.e. Section 1.1. In Section 1.2, we state our general research aims. Commonly used approaches and their limitations are briefly discussed in Section 1.3, after which we outline our proposed method to address these limitations in Section 1.4. The structure of this thesis as a whole is described in Section 1.6.

1.1 Problem

Different approaches to exoplanet discovery exist. This work focuses on the transit method, the most successful method in terms of number of discoveries. The transit method relies on the rare event that a planet moves in front of its host star in our line of sight, i.e. the so-called "transit". Transits express themselves in the form of periodic dips in the observed brightness of a star over time. The detection of these dips in stellar data is therefore essential for the discovery of new exoplanets.

The observed brightness, or flux, of a star over time is referred to as the star's "light curve". Generally, transit signals have a duration in the order of hours and light curves have a time span in the order of days or months. A transit signal may repeat itself if the orbital period of the exoplanet is shorter than the time span of the light curve. In case the orbital period is longer, a transit signal may occur only once, in which case we speak of a "monotransit". Moreover, multiple planets might orbit the same star, in which case we can have multiple transit signals from different planets in the same light curve.

The physics of transits is well-understood, but the detection of their signatures in light curves is hindered by stellar and spacecraft induced noise. In addition, transit signals may vary in depth, duration and general shape, depending on stellar and planetary parameters. Furthermore, transit signals are in general weak and sparse, so the chance of any given light curve to contain detectable transit signals is low. A detailed description of the challenges and their origins is given in Section 2.3.

The problem we address is thus the detection of dips in light curves of stars that could indicate the presence of an orbiting exoplanet. In literature, however, some ambiguity exists in the naming of different steps in the process of discovering transiting exoplanets. In the detection step, i.e. the focus of this thesis, we search for potential signals in a light curve, and provide parameters such as the epoch t_0 and orbital period P of the detected candidates. The epoch is a reference time, e.g. the time of the first transit event in the light curve, which together with P allows for retrieving the signal that was found. Detection should not be confused with identification or vetting, the step in which the detected signals are classified as false positive, planet candidate, or candidate of another type of object.

1.2 Research aims

The field of exoplanet science has seen a rapid increase in the amount of data over the last decades. For this reason, artificial intelligence (AI) has come to play a more important role in the discovery of exoplanets. Most applications of AI, however, focus on the identification step, e.g. *Autovetter* (Catanzarite, 2015), *Robovetter* (Coughlin, 2017) and *AstroNet* (Shallue and Vanderburg, 2018). Few works exist on the application of AI for the detection of transit signals, an example of which is the work of Pearson et al. (2018). Following their line of research, we aim at contributing to the development of AI-based algorithms for transit signal detection, with the goal to overcome limitations encountered in previous approaches.

1.3 Common approaches and limitations

Most commonly used in the task of transit detection is the Box Least Squares (BLS) algorithm (Kovács et al., 2002). This algorithm uses a box-shaped model of a strictly periodic signal. In order to search for transits, the model is fitted to the data at different signal durations, periods and epochs. Variants to this algorithm exist, for example Transit Least Squares (TLS) (Hippke and Heller, 2019), or Sparse Box Least Squares (SBLS) (Panahi and Zucker, 2021).

One limitation of these methods is their requirement of a detrended input light curve. Detrending is the process of removing irrelevant time-dependent noise from the light curve. In the process, however, information is altered at short time scales, so the signal that we wish to find may get reduced or altered (Hippke et al., 2019). Alternatives have therefore been proposed, for example simultaneously modeling the background while searching for transit signals Foreman-Mackey et al. (2015).

However, what these methods still have in common is their brute-force approach to searching for signals. They rely on iterating through trial configurations of parameters that define the signal's shape and timing. In particular for transit models more realistic that the box-function, this iteration over potential signals can be costly. In addition, in case the search is directed towards multiple planets in the same light curve, this process has to be repeated several times, each time masking the previous signal that was found. Furthermore, the model configurations to be iterated over are generally specified by the user, which might cause the search to be biased towards specific signal shapes.

An approach based on AI may overcome several of these limitations, as shown by Pearson et al. (2018). In their work, the use of a one-dimensional convolutional neural network (CNN) is proposed for the task of transit detection. On the one hand, neural network outputs can be more difficult to interpret than outputs of box-fitting algorithms, because a neural network generally has complex hidden dynamics which make it function like a "black box". On the other hand, they come with several benefits. The CNN does not iterate over transit shapes, and detrending the input light curve is not necessary. In their proposed method, however, the CNN is only capable of providing a single output for an input light curve segment of fixed size. This means that it cannot directly be applied to a full-length light curve of arbitrary size, and we need to take additional steps to determine the timing of potential signals. One option is to apply the CNN to overlapping segments in the light curve to obtain outputs at each time step. However, this approach makes the CNN less intuitive and efficient in its use for detection.

In the task of transit signal detection, the use of recurrent neural networks (RNNs) remains unexplored. The main property of the RNN is that it takes into account temporal information and can therefore generally handle sequential data well, depending on the task. Light curves consist of sequential data, so the RNN would be a suitable for our task. The questions therefore remain whether and how RNNs can be used for the task of transit detection; whether they can provide an efficient and competitive alternative to classical transit detection methods; and whether they can improve over CNNs in terms of their applicability. Additionally, we raise the question of how the outputs of RNNs can be made more interpretable for the task of transit detection.

1.4 Proposed method

We evaluate the use of RNNs for the task transit signal detection. As opposed to CNNs, RNNs can (i) handle arbitrary input sizes and (ii) provide an output at every time step of a given light curve. We make use of property (i) by training the network on light curve segments and applying it to full-length light curves for detection. We make use of property (ii) by training the model to identify exactly those time steps that are part of a transit signal. In other words, the RNN is trained to classify each data point as signal or non-signal.

An important factor in the network's performance is the way in which the input data is preprocessed. Therefore, the question arises which preprocessing steps can or should be considered when using RNNs for this task, and how they affect the model's performance. In this work, we evaluate several different preprocessing steps concerning the scaling of the input data and the way data gaps are handled. For example, to address the problem of non-uniform time intervals between measurements, we explore the use of a generative RNN, which is trained to fill in data gaps as it processes through a light curve.

In experiments using simulated data, we assess whether an RNN-based detection algorithm is a viable and competitive alternative to existing approaches. In the task of monotransit detection, we investigate the question of whether the outputs of the RNN can be made more interpretable, in order to set better thresholds and reduce the amount of false detections. To do so, we extend the network to produce confidence outputs in addition to the standard classification outputs at each time step, as proposed by DeVries and Taylor (2018). The confidence outputs are supposed to indicate the confidence over the corresponding standard outputs at each time step.

In the case of detecting repeating signals, rather than monotransits, additional steps are required. This work investigates two approaches to determine the periodicity in such a case. In combination with the RNN, both of

these approaches combine to a novel RNN-based transit detection algorithm, which is able to output t_0 and P of candidate transit signals in any given light curve.

The first approach, referred to as PTS-Peak, is inspired by the work of Pearson et al. (2018). The probability time series (PTS) refers to the outputs of the RNN over time. Peaks in the PTS would indicate a potential transit event at the corresponding time steps. PTS-Peak uses the average distance between subsequent peaks as a period estimate of the signal. In the search for multiple planets, one could apply this method iteratively, each time removing the most prominent peaks from the PTS. To help resolve ambiguities in determining which PTS peaks belong to which signal, we explore the use of learned signal representations by the RNN.

The second approach, referred to as PTS-Fold, builds on top of existing approaches, where the input, in this case the PTS, is folded over a set of trial periods. For each trial period, a score is given based on the aggregated values within the folded PTS. The best scoring value for the period is used as a period estimate.

1.5 Summary of contributions

This work reports the first attempt in literature of using RNNs for the task of transit detection. The research presented revolves around the question of whether and how RNNs could be applied to compete with conventional transit detection algorithms. We provide a detailed analysis of what preprocessing steps could be considered when applying this network to light curves to search for signals. Successes and failure cases of our method are identified, and the general performance is evaluated in comparison to existing approaches. Our approach does not require detrending of input light curves as opposed to commonly used methods. During search, the RNN does not iterate over transit durations, depths or other parameters that define the signal's shape, since the model is implicitly trained to recognize these features prior to application.

In short, we found the RNN to show most potential in the task of monotransit detection, because it does not rely on periodicities of signals and it can be applied highly efficiently. It outperformed the box-fitting algorithm of Foreman-Mackey et al. (2016) in the task of retrieving single transit events in simulated light curves. In the case of multiple transit signals in a single light curve, we found the BLS algorithm to benefit more from the periodicity of the signal, as it outperformed our RNN-based algorithms. In both cases however, the algorithms tested were able to detect signals that the other one was not able to find. In other words, the methods performed complementary. This result suggests that an RNN-based transit detection method could open the door to detecting exoplanets that would otherwise be overlooked by conventional algorithms.

1.6 Outline

This thesis is structured as follows. In Chapter 2, the necessary background is given, such as the transit method and existing approaches to detect transit signals in light curves. The details of our proposed methods are described in Chapter 3. In Chapter 4, we present the experiments and results, which are discussed in Chapter 5. Finally, the work is summarized and concluded in Chapter 6. Appendix A can be referred to for additional information, for example on conference contributions that were based on this work.

Chapter 2

Background

The following sections provide necessary information to understand the research and the context in which this research was conducted. The transit method is further described, as well as the missions which are responsible for the available data in the field. Challenges and their origins are described in Section 2.3, after which we further discuss existing methods for transit detection in Section 2.4. The theory and applications of RNNs are described in the last section of this chapter.

2.1 Transit method

Among all exoplanet discovery methods, the transit method has most exoplanet discoveries on its name. At the time of writing, over 75% (3343 in total) of the known exoplanets (4424 in total) were discovered using this method[1]. Of the other methods, the radial velocity method is the most successful, covering about 20% of the exoplanet discoveries. The radial velocity method works by measuring the shifts in the spectrum of a star which are due to the gravitational pull of an orbiting planet. While this used to be the prominent method for exoplanet discovery, nowadays it is mostly used to complement the transit method as a means of confirming exoplanet candidates, or characterizing planetary systems. Anther method closely related to the transit method uses transit timing variations (TTVs) of transiting planets to infer the presence of an additional planet in the same system. TTVs are deviations from the expected periodicity of a signal, which could be caused by a disturbing planet in a different orbit around the same star. Other methods to exoplanet discovery exist, but for the purpose of this thesis we do not go into depth for these methods.

Most planets in our Solar System can be directly observed with a small telescope or even the unaided eye. For the observation of exoplanets, distances between stars plays a role, which are large compared to the distances between stars and planets. The distance between Earth and the Sun is 1 Astronomical Unit (1 AU = 1.496×10^8 km) and the distance to the nearest star is about 10^5 AU. Exoplanets are thus far away and form a compact system together with their host star as observed from Earth. This makes direct imaging of exoplanets difficult. Therefore, most methods rely on indirect measurements of potential exoplanets. This is also the case for the transit method, for which only the brightness of a star is monitored over time. Dips in the observed brightness are used as indication of an exoplanet passing in front of the stellar disk. A requirement for this method is therefore that the system is seen edge-on.

Figure 2.1 illustrates the effect of a transit event on a simplified light curve of a star. The exoplanet does not contribute to the light that is observed, only to the absence of light. When the planet moves in front of the stellar disk, i.e. during "ingress", the observed light from the star drops. Around the mid-transit time, the amount of observed flux only changes slightly over time, depending on the distribution of light emitted from the stellar surface (see Section 2.3.3). When the planet moves away from the stellar disk, i.e. during "egress", the observed flux increases again to its normal level. The depth of the signal can be described by the fraction of light that is blocked by the planet, which in the case of a simplified edge-on system is $\delta = R_{pl}^2/R_*^2$. The duration of the signal depends on the orbital period P of the planet. In turn, the period of the planet depends on the semi-major axis a of its orbit, according to Kepler's third law:

$$P^2 = \frac{4\pi^2}{GM} a^3, \tag{2.1}$$

where G is the gravitational constant ($G = 6.67408 \cdot 10^{-11} \mathrm{m^3 kg^{-1} s^{-2}}$), and M the star's mass. In case the eccentricity of the planet's orbit is 0, i.e. the orbit describes a circle, a is simply the distance of the planet from its host star. For increased values for the eccentricity, the orbit becomes more elliptical. Another parameter

[1] https://exoplanetarchive.ipac.caltech.edu/docs/counts_detail.html

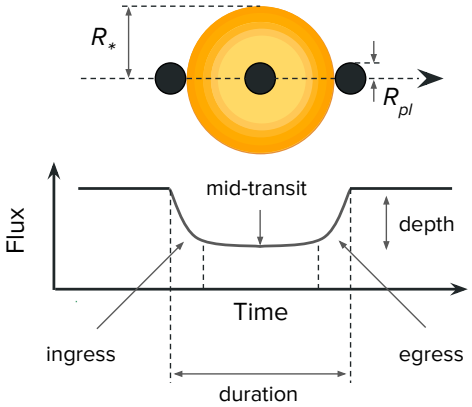


Figure 2.1: An illustration of a planet (black) transiting in front of its host star. The planet blocks a fraction of the star's emitted light, so we observe a dip in the light curve of the star. The depth of this transit signal depends on the planet radius R_{pl} relative to the star's radius R_*.

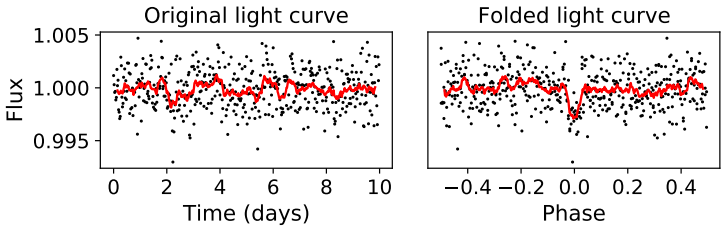


Figure 2.2: An example of "folding" a light curve as a means increasing the signal-to-noise ratio (SNR) of a transit signal. In the original light curve, five transit events with $P = 2.03$ days, $t_0 = 0.21$ days are hidden in the noise. These become visible when the light curve is folded over the correct period, as shown on the right. The red line indicates the running mean over 15 data points.

that is relevant is the inclination i of the orbit. If $i = 90°$, then the system is seen edge-on, meaning that the planet would transit its host star. For values $i < 90°$ or $i > 90°$, the planet crosses the stellar disk further away from its center, up to a point where the planet does not transit the star at all.

Figure 2.2 shows a more realistic, but simulated light curve with five transit signals of a single planet. The light curve is dominated by white noise and the transit signals are therefore difficult to recognize. For this reason, the transit method often works better if the input light curve is "folded" over the period of the planet. In the resulting phase curve, the signal-to-noise ratio (SNR) of the signal is then increased compared to the individual signals in the original light curve. The problem in detection, however, is that the correct period of a planet is not known beforehand, let alone the fact that there may not be a planet in the first place.

2.2 Exoplanet-hunting missions

Since the first exoplanet discoveries [TODO: cite], the rate at which exoplanets are discovered has grown exponentially. Ground-based surveys such as WASP, HAT, TRAPPIST and KELT [TODO: references] are responsible for the discovery of several exoplanets. However, the major turning point was when the search for transiting exoplanets was taken into space.

CoRoT was the first space-based mission designed to search for transiting exoplanets and added 33 exoplanets to the list. Kepler and its continued mission K2 hold the discovery of over half of the known exoplanets (> 2500). Kepler observed a small patch of the sky and monitored over 500,000 stars during its mission. Most targets were observed with an observation interval of 30 minutes, i.e. a 30-minute cadence. TESS, Kepler's successor, monitors brighter and closer stars in a full-sky survey. After each 27.4-day sector, the telescope points at a different patch of sky. Midway each sector, the spacecraft transmits data back to Earth, during which no new data is gathered. With over 200,000 stars observed and "only" 131 exoplanet discovered, thousands discoveries more are expected [TODO: cite]. The amount of data in the field will only increase more rapidly, as future telescopes will get bigger and better. Planned for launch in 2026, PLATO will monitor the brightness of around up to a million stars for the detection and characterization of transiting exoplanets.

2.3 Challenges in detecting transit signals

In the task of transit detection, the presence of an exoplanet is not known beforehand, so we need to search the light curves for potential signals. However, many noise sources hinder the search and could trigger false detections. This and other challenges are described in the following subsections.

2.3.1 Stellar variability

Stars do not emit photons at a constant rate. A star's surface is dominated by cell-like structures called granules. Energy transport within the star causes hotter and brighter cells of plasma to rise, and cooler and dimmer plasma to descent. This granulation effect leaves the stellar surface constantly changing, and results in fluctuations in the star's light curve. Granulation has timescales of about a half hour up to several days (Kallinger et al., 2014). At larger time scales, starspots could leave their imprint on a light curve. These visually dark regions on the star's surface are relatively low in temperature, and thus decrease the flux. As the star rotates, the starspot could come in and out of view, which results in large scale fluctuations in the star's light curve. This effect is called rotation modulation. Additionally, other phenomena could be at play such as stellar oscillations or sudden outbursts of energy called flares. In short, a light curve may exhibit quasi-periodic patterns caused by complex mechanisms.

2.3.2 Instrumental and photon noise

All data used in this work is assumed to be collected by space-based observatories. Although space offers a good environment for astronomical observations, the instruments still suffer from imperfections. For example, temperature changes in the spacecraft might alter the pointing of the telescope, which could cause jumps or drifts in the observed flux from stars in the field of view. Incorporating information on the telescope's pointing, or centroids, in addition to the flux values seems beneficial in some exoplanet-related tasks [TODO: cite centroid examples]. Stray light, for example reflected off the moon, might also influence the quality of observations. Poor quality data may be flagged by the spacecraft's pipeline, which could then be filtered out. However, this leaves missing data points in the light curve, which might pose a problem to certain processing algorithms. Another source of noise, although not due to instrument imperfections, is photon noise. This noise originates from the arrival of different amounts of photons at the detector between different observations. In general, this type of noise can be seen as the time independent Gaussian distributed noise.

2.3.3 Transit signal shapes

The basic shape of a transit signal is determined by, among others, the planet's orbital period, radius relative to its host star, distance from its host star, and inclination and eccentricity of its orbit. Each parameter affects the shape of a transit differently. The stellar surface also plays a role in the shape of the signal. The limbs of the star seem darker than its central region. This is because at the central region, we look deeper into the star where temperatures are higher. When a planet passes in front of the star, it therefore blocks a varying portion of the star's total emitted light. For this reason, a box-shaped transit model is not accurate. More realistic transit models take this limb darkening effect into account. Other phenomena can also cause transit depths to vary over time, for example when a planet passes in front of starspots during its transit. Although it is not the task of the transit detection algorithm to determine all the parameters that cause a certain transit shape to be observed, it should be robust against different signal shapes during its search.

2.3.4 Weak and sparse signals in big data

The requirement for the transit to work is that the planetary system is seen edge-on. However, nothing restricts a system from being oriented randomly. For this reason, even if all stars would host exoplanets, a large portion would still go undetected. Moreover, the farther the planet from the host star, the smaller the chance that the planet transits the star. For example, the chance of earth to transit the sun if it were observed from outside the solar system is only 0.47% (Borucki and Summers, 1984). Even if it does, Earth transits the Sun only once per year and leaves dip of 80 ppm in the Sun's observed flux for 13 hours. In other words, transit signals are weak and sparse, especially those we are most interested in. The reason why still many exoplanets are discovered by the transit method, is mainly due to large amount of available data. For example, TESS alone collects over 20 GB every day [TODO: cite TESS data amount], which alone could also be a challenge to work with.

2.3.5 Astrophysical false positives

Sometimes, a false positive detection is not due to stellar or spacecraft induced noise, but due to other phenomena. For example, a large portion of the stars exists in a binary configuration. In case an eclipsing binary (EB) system is seen edge-on from Earth, the stars transit each other just like an exoplanet would transit its host star. Generally, EB signals are larger than for exoplanets, because stars are larger than planets. However, if the stars have grazing transits, the signals can become smaller and more similar to exoplanet transits. EB systems lurking in the background of an observed star can also produce disturbing signals. These background EBs (BEBs) are much farther away than the star under consideration, but contribute to the star's observed light curve. The BEB transit signals will thus be smaller than EB signals and could mimic exoplanet transit signals in the light curve of the foreground star. In the detection step, however, it is not the main goal to discern (B)EBs from exoplanet transit signals. Preferably, most EB signals are filtered out from the start, but restricting our detection algorithm too much on what kind of signals it is allowed to pick up, may exclude various interesting signals from being detected. For example, the detection step could return a signal that is in fact caused by a large exocomet or asteroid instead of an exoplanet, which would still be of great astrophysical value to find and study. Furthermore, such detected signals may well be filtered out in the vetting step. For this reason, we exclude astrophysical false positives from this work, and leave them for future research.

2.4 Transit detection methods in literature

The following subsections describe previously proposed methods for transit detection in further detail. The pipelines of the TESS and Kepler missions are included in the description, because they have a built-in module for transit detection. Classical methods are described first, after which we describe the more recent developments in AI in the task of transit detection.

2.4.1 Box and Transit Least Squares

The Box Least Squares (BLS) algorithm, proposed by Kovács et al. (2002), is widely adopted in the task of transit signal detection [TODO: cite examples]. Its simplified model of the transit signal makes it a relatively efficient algorithm, because the model is only parametrized by the transit depth, duration and timing of the signal. The model is fitted to the data by minimizing the squared error, or similarly, maximizing the log likelihood of the model with respect to the transit depth δ for a given configuration of orbital period P, transit duration τ and epoch t_0[2]. The log likelihood at different values for P, maximized over $\{\delta, \tau, t_0\}$ is proportional to the Signal Residue (SR) as defined by Kovács et al. (2002). The SR at each given trial period can be used to defined the BLS periodogram, i.e. SR over P. More commonly, the Signal Detection Efficiency (SDE) is used instead of SR, which is defined as

$$SDE = \frac{SR_{\text{peak}} - mean\,(SR)}{std\,(SR)}, \tag{2.2}$$

where SR_{peak} is the maximum SR in the periodogram, and $mean\,(SR)$ and $std\,(SR)$ are the mean and standard deviation of the periodogram respectively.

An example of the BLS periodogram is shown in Figure 2.3, in which we can see how the peaks in the periodogram can be used for detection. For example, a peak in the at a given period P could indicate the detection of a signal with corresponding parameters $\{\delta, \tau, t_0, P\}$, which are provided for each point in the periodogram. However, as is clear from the figure, the harmonics of the signal may be confusing and lead to wrong detections.

Instead of using the SDE to set a detection threshold, a detection threshold could be set on the signal-to-noise ratio (SNR) of transit candidates. Ignoring the presence of time dependend noise, the SNR can be defined as $\delta/\sigma_w \cdot \sqrt{n_t}$ with $n_t \propto \tau$ the number of measurements that belong to the transit signal and σ_w the estimated white noise in the detrended light curve. However, Pont et al. (2006) show that a detection threshold for BLS can be set better if time dependend noise is accounted for in the approximation of the SNR. This is likely because background patterns with time scales similar to the transit duration could remain intact after detrending.

Several variants to BLS exist. For example, Carter and Agol (2013) relax the assumption of strictly periodic signals, and use a quasi-periodic transit model that is fitted to the data. Foreman-Mackey et al. (2016) omit the periodicity assumption altogether and use an algorithm similar to BLS to search for monotransits. In their work, a box function is fitted to the data at multiple points in time, and a detection threshold for single events is set on the SNR above the noise floor determined by a sliding median filter.

[2] https://docs.astropy.org/en/stable/timeseries/bls.html

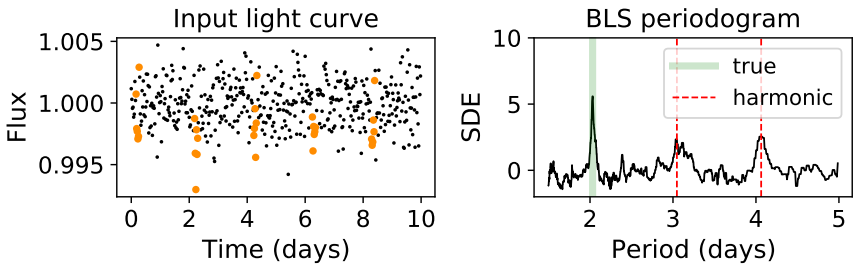


Figure 2.3: The BLS algorithm applied to the same light curve from Figure 2.2. The highest peak in the periodogram corresponds to the correct period P of the signal. However, the harmonics of the signal, e.g. $3/2P$ and $2P$ as shown here, also produce peaks.

More recently, the Transit Least Squares (TLS) algorithm was proposed by Hippke and Heller (2019). As opposed to BLS, TLS takes into account the effect of limb darkening on the transit shape, and is therefore more expressive than BLS. For this reason, TLS generally performs better at detecting transit signals than BLS, however at the cost of longer computation times.

The performance of BLS and TLS largely depends on the way the light curve is detrended prior to the search. Hippke et al. (2019) compare a wide range of detrending methods, before applying TLS to search for transits. They evaluate each method by the extend to which known transit signals are retrieved by the TLS algorithm, after applying the detrending method. The methods evaluated include sliding mean and median filters, Gaussian processes, the Savitzky-Golay filter, and more. The results showed that detrending could significantly alter or reduce the transit signals, for some methods more than others. A time-windowed median filter performed reasonably considering its computational efficiency. Furthermore, for window-based filters, it was found that a window size of three times the transit duration is the best choice for detection. The problem is however, that the presence of a transit is not known beforehand in the task of detection, let aside its duration.

2.4.2 Simultaneously modelling background and signal

In order to avoid the risks that are involved with detrending, one could simultaneously model background stellar activity with the transit signals. This approach was taken by Foreman-Mackey et al. (2015), where light curves were described as a combination of the 150 most representative components of the principle component analysis (PCA) of K2, while simultaneously fitting for a box-shaped transit model. Since the background is modelled side-to-side with the transit signal, the background model is less prone to overfit to the transit signal, and the transit signal is therefore expected to stay better intact.

In contrast, Kovács et al. (2016) argue that with simultaneous modelling, there are more degrees of freedom and thus more chance of false positives. They found better results by first detrending the light curve, and subsequently searching for transit signals. This approach was also found to be considerably more efficient.

2.4.3 Kepler and TESS pipeline

The pipelines of Kepler and TESS cover the process of reducing raw images of stars to systematics corrected light curves, which are subsequently searched for transit signals. The TESS pipeline is largely based on the Kepler pipeline (Jenkins et al., 2016), so we base our description of both pipelines on the Transiting Planet Search (TPS) module as described in the Kepler Data Processing Handbook Jenkins et al. (2017).

In the TPS module, the input light curve first undergoes a series of preprocessing steps, e.g. stitching sectors together and filling data gaps. Subsequently, a time-varying whitening filter is applied, which is supposed to clear the light curve of irrelevant time dependent noise. The use of a pre-whitening filter for transit search has been adopted more often (e.g. Carpano et al. (2003)) as it is considered to be the optimal detector in combination with a simple matched filter in the case for colored Gaussian noise, as explained in Jenkins (2002). However, Rodenbeck et al. (2018) note that a pre-whitening filter can introduce features that could be misinterpreted as signal. The pipeline therefore removes positive flux outliers which could introduce transit-like features in the whitened flux. Since the whitening filter can also alter transit signals, the same whitening is applied to the transit pulse train that is used as transit model. After single candidate transit events have been identified, a grid of periods, durations and epochs is searched through to find a least squares fit to each potential signal. A representative limb darkening model is used to fit with the data.

Although the pipeline is far in its development and is constantly being improved, it still relies on heavy preprocessing and an extensive search through parameter configurations of potential signals. Furthermore,

the TPS module in the pipeline is designed as general algorithm to detect most of the hidden transit signals. Therefore, this algorithm might still miss signals in special cases.

2.4.4 Other classical detection methods

In cases where a periodic signal resembles a sine function, the Fourier transform (FT) provides a good way to detect it. For a transit signal this is in general not the case as its duration is short relative to its period. For ultra-short-period (USP, < 1 day) planets on the other hand, the duration becomes larger relative to the period, and the FT will have similar detection performance as the BLS algorithm, as was shown by Sanchis-Ojeda et al. (2014). In their work, an FT-based detection method is applied to detect transits from USP planets in Kepler data. They argue that the FT produces less disturbing harmonics in its resulting spectrum compared to BLS, but BLS is more effective for longer period planets.

Another approach to transit detection is to use the dispersion of the phase folded light curve. Plavchan et al. (2008) fold a given light curve over several trial periods, each time evaluating the difference between the folded light curve and its boxcar-smoothed counterpart. A small set of periods for which the folded curve corresponds best to the smoothed phase curve, is further analysed for transits. Wheeler and Kipping (2019) also propose a method which aims to minimize the phase dispersion by folding the light curve at different trial periods. Since this method does not assume any specific signal shape, it is sensitive to strictly periodic signals of arbitrary shape. Their method was applied by Chakraborty et al. (2020) to find 377 previously unreported signals in the first year of TESS observations.

2.4.5 "Intelligent" algorithms

The human eye can function as an excellent tool for pattern recognition and can exceed algorithms in complex ways. For this reason, the citizen science project Planet Hunters was launched. Participants, or users, are presented with light curves from Kepler and asked to flag parts of the light curve that resemble transit signals. Six months after the launch, millions of classifications were made and later Fischer et al. (2012) reported the detection of the first two planet candidates that were flagged by participating volunteers. This process can also be viewed as a detection algorithm. The user is in this case the detector, which utilizes prior knowledge about the shapes of different transit signals to flag potential signals. Without requiring the light curve to be detrended or folded, individual events are flagged by the detector. If there is enough confidence about a certain signal, i.e. enough users have flagged the same events, the "detection" is passed to the vetting stage. For Planet Hunters, this stage consists of experts in the field who rule out clear false positives and conduct follow up observations to confirm the planets.

Similarly, one could use AI for the task of transit detection. Pearson et al. (2018) trained a one dimensional CNN to classify light curve segments as signal or non-signal. The CNN lends itself well for identification and forms the basis of several recent works in the field such as *AstroNet* (Shallue and Vanderburg, 2018) and variants (Ansdell et al., 2018; Dattilo et al., 2019; Koning et al., 2019; Yu et al., 2019; Osborn et al., 2020). For detection, however, the model is limited by the fact that is only allows relatively small inputs of fixed size. Therefore, Pearson et al. (2018) propose two approaches to use the CNN for detection. In the first approach, the network is applied to overlapping segments in the light curve to obtain outputs at each point in time, which are referred to as probability time series (PTS). Subsequently, they propose to use the average distance between peaks in the PTS can give an estimate of the periodicity of the signal. In the second approach, the light curve is folded over given trial periods, each time applying the CNN to overlapping segments in the resulting phase curve. A detection in the phase curve directly gives an estimate of the periodicity, but this approach requires the same parts of the light curve to be evaluated many times by the CNN. Zucker and Giryes (2018) tested the feasibility of using CNNs for detection in comparison with the BLS algorithm. Their experiments were based on simulated light curves with Gaussian processes to account for stellar variability. The task they evaluated both methods on was not entirely that of detection, but instead the binary classification task of classifying input light curves as signal or non-signal.

Extending on the work on CNNs, Chintarungruangchai and Jiang (2019) evaluated the use of two-dimensional CNNs for detection. Instead of folding the light curve to one dimension, they fold the light curve such that each folded segment is stacked on top of each other. Subsequently, they train the CNN to classify these two-dimensional "images" as signal or non-signal. Although multiple trial periods still need to be evaluated to search for a potential signal, this method is found to be more robust against deviations from the true period than the method proposed by Pearson et al. (2018). However, the problem remains that the methods evaluated by Chintarungruangchai and Jiang (2019) and Zucker and Giryes (2018) only provide a binary output for an input light curve, and do not allow to determine the exact timing of the individual signals.

2.5 Recurrent neural network (RNN)

RNNs seem to be missing from literature concerning transit detection. They have, however, been applied in related tasks. In the following subsections, we first describe what RNNs are, after which we discuss different application of this network in related fields.

2.5.1 Theory

The field of machine learning covers a broad range of techniques and methods, e.g. k-nearest neighbours, decision trees, linear regression. A subfield of machine learning is deep learning, which is based on artificial neural networks (NNs) such as the RNN and CNN. A simple NN architecture is the multilayer perceptron (MLP), which is a network consisting of fully-connected (FC) layers of computational units, often referred to as neurons or nodes. The output at each layer $h_i \in \mathbb{R}^n$ in an MLP can be obtained by transforming the preceding layer $h_{i-1} \in \mathbb{R}^m$ as:

$$h_i = \phi(W_{i-1,i} h_{i-1} + b_i), \tag{2.3}$$

where $W_{i-1,i} \in \mathbb{R}^{n \times m}$ is the weight matrix between layers $i-1$ and i and $b_i \in \mathbb{R}^n$ the bias vector. $\phi(\cdot)$ is the activation function, which is used to introduce non-linearities in the network. Between layers, $\phi(\cdot)$ is often chosen to be the ReLU activation function, i.e. $\phi(a) = \mathrm{ReLU}(a)$, where

$$\mathrm{ReLU}(a) = \max(0, a). \tag{2.4}$$

The network's output y for a given input $X \in \mathbb{R}^N$ is given by the function $y = g(X, W)$, which represents the neural network. In this notation, we use W to refer to the collection of all weight and bias parameters in the network. For example, for an MLP with one hidden layer we have

$$g(X, W) = \psi(W_{1,2}(\phi(W_{0,1} X + b_0)) + b_1), \tag{2.5}$$

where, for example, in the case of binary classification, the scalar output y is mapped between 0 and 1 by choosing $\psi(a) = \sigma(a)$, where $\sigma(a)$ is the logistic sigmoid function given by

$$\sigma(a) = \frac{1}{1 + \exp(-a)}. \tag{2.6}$$

For an RNN, the network function, say $f(X, W)$, is more complicated. This is because an RNN uses each element x_j in X to update its hidden state recursively. Consider X to be a time series, so we can speak of individual time steps j with corresponding data points x_j. The hidden state $h_j \in \mathbb{R}^K$ of the "vanilla"-RNN at time step j is given by

$$h_j = \phi(U h_{j-1} + V x_j + b), \tag{2.7}$$

where $U \in \mathbb{R}^{K \times K}$ and $V \in \mathbb{R}^{K \times N}$ are weight matrices and $b \in \mathbb{R}^K$ the bias vector, which are the same for each time step. An output for each time step can be obtained by applying some function to h_j. For example, we can apply Equation 2.5 to the hidden states of the recurrent layer to obtain an output y_j for each time step j, i.e. $y_j = g(h_j, W)$. The complete RNN then outputs a vector $Y \in \mathbb{R}^N$ with elements y_j for an input sequence X, i.e. $Y = f(X, W)$.

Equation 2.7 represents the update rule for a unidirectional RNN, i.e. an RNN which only propagates through time in one direction. However, in some applications it is beneficial to utilize information form both directions in time. This can be achieved by concatenating the hidden states of two RNN components, propagating though the input in opposite directions:

$$H_j = \begin{bmatrix} h_j \\ h'_j \end{bmatrix} = \begin{bmatrix} \phi(U h_{j-1} + V x_j + b) \\ \phi(U' h'_{j+1} + V' x_j + b') \end{bmatrix}. \tag{2.8}$$

h'_j is of the same dimensionality as h_j and is obtained using weight matrices V' and W' and bias vector b', which are of the same dimensionality as V, W and b respectively. The result, $H_j \in \mathbb{R}^{2K}$, can be used in the same way as before, e.g. $y_j = g(H_j, W)$.

The key to a well-performing network is not only to have an appropriate network architecture, but also to have a good training procedure. A network is trained by updating its trainable parameters W to optimize for a given cost or loss function $\mathcal{L}$. To do so, generally the derivative of $\mathcal{L}$ is taken with respect to W with a process called "backpropagation", which efficiently computes the gradient at each layer using the chain rule. The derivative can be used to update W in the direction of a lower loss value, for example using stochastic gradient descent (SGD) or more advanced optimization algorithms such as Adam (Kingma and Ba, 2014).

Given a data set, the network is generally trained on a "training" subset, evaluated on a "validatation" subset after each training iteration, and applied to an unseen "test" subset to evaluate its final performance

after training. Overfitting occurs when a network performs well on the training set, but poorly on the validation or test set. One could avoid overfitting by applying a penalty on the norm of the weights in the network, e.g. L2 regularization:

$$\mathcal{L} = \mathcal{L}' + \frac{\alpha}{2}\|W\|_2^2. \tag{2.9}$$

Here $\mathcal{L}'$ is a task-dependent loss term, $\|W\|_2^2$ is the sum over all weight values squared, and α is the parameter that is used to tune the weight penalty. For SGD, L2 regularization is also referred to as weight decay, but for adaptive gradient algorithms such as Adam, these two terms are not equivalent (Loshchilov and Hutter, 2017). The authors of this work therefore propose a modification to be used for adaptive gradient algorithms, which they refer to as decoupled weight decay.

Another problem may occur if the network has many layers. Since the gradient of the first layer in the network is computed using the chain rule, i.e. by multiplying the gradients of all subsequent layers, we may have that the gradients quickly explode or vanish. For a vanilla-RNN this is problematic, because the gradient is propagated through time, in which we treat each time step in the RNN as a separate layer. Suppose that the input X consists of 1000 data points. Even if the gradients of each individual layer lie around 0.9, then the gradients of the loss with respect to the weights of the first layer can get as low as $0.9^{1000} \approx 10^{-46}$. This means that earlier time steps will contribute very little to the weight update, and the RNN will not be able to learn long term dependencies.

For this reason, different RNN architectures have been proposed. Several proposed RNN architectures use gates to better control the flow of information. One of these is the long short-term memory (LSTM, Hochreiter and Schmidhuber (1997)) RNN. For LSTM, Equation 2.7 can be replaced by the following[3]:

$$i_j = \sigma(W_{ii}x_j + b_{ii} + W_{hi}h_{j-1} + b_{hi}) \tag{2.10}$$
$$f_j = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{j-1} + b_{hf}) \tag{2.11}$$
$$g_j = \tanh(W_{ig}x_j + b_{ig} + W_{hg}h_{j-1} + b_{hg}) \tag{2.12}$$
$$o_j = \sigma(W_{io}x_j + b_{io} + W_{ho}h_{j-1} + b_{ho}) \tag{2.13}$$
$$c_j = f_j \odot c_{j-1} + i_j \odot g_j \tag{2.14}$$
$$h_j = o_j \odot \tanh(c_j), \tag{2.15}$$

where $\odot$ is the element-wise product. The gates i_j, f_j, g_j, o_j are known as the input, forget, cell and output gates respectively, and c_j is known as the cell state.

Another, yet similar, architecture is the gated recurrent unit (GRU, Cho et al. (2014)) RNN. For GRU, Equation 2.7 can be replaced by the following[4]:

$$r_j = \sigma(W_{ir}x_j + b_{ir} + W_{hr}h_{j-1} + b_{hr}) \tag{2.16}$$
$$z_j = \sigma(W_{iz}x_j + b_{iz} + W_{hz}h_{j-1} + b_{hz}) \tag{2.17}$$
$$n_j = \tanh(W_{in}x_j + b_{in} + r_t \odot (W_{hn}h_{j-1} + b_{hn})) \tag{2.18}$$
$$h_j = (1 - z_j) \odot n_j + z_j \odot h_{j-1}, \tag{2.19}$$

where the gates r_j, z_j, n_j are known as reset, update and new gates respectively. GRU has fewer gates and thus fewer parameters than LSTM. In some cases, however, exploding gradient can still cause problems (Kanai et al., 2017).

2.5.2 Applications

RNNs have been applied to light curves to predict stellar parameters and the number of potential transit signals by Hinners et al. (2018). In the latter, however, the bidirectional LSTM (bi-LSTM) only achieved near-random results, which the authors attributed to the imbalance in the data. We note that it might also have been due to the sparse learning signal that is given to the RNN. Namely, only after having seen about 7000 input data points, the RNN outputs a single classification vector or value, for which a learning signal is given. This way, figuring out where transit signals are present in a given light curve can only be learned implicitly by the model, even though we have the possibility to provide this information as learning signal at every time step.

Another application of the RNN is that of detrending light curves (Morvan et al., 2020). An LSTM was trained to learn out-of-transit patterns, and interpolate these within the duration of a given transit signal. By doing so, the authors show that the background of a transit signal can be better subtracted, which allows for better characterization of the exoplanet. Results improved if centroid data was included.

Other applications of RNN mostly involve the classification of light curves as a whole, for example for the classification of variable stars. Jamal and Bloom (2020) compare different neural network architecture for this

[3] https://pytorch.org/docs/stable/generated/torch.nn.LSTM.html
[4] https://pytorch.org/docs/stable/generated/torch.nn.GRU.html

task, including dilated temporal convolutional networks (dTCNs), LSTMs, GRUs, temporal convolutional NNs (tCNNs) and encoder-decoder models. The LSTM and tCNN were found to be performing best, with the latter having the benefit of shorter computation times.

In the context of variable star classification, the problem of irregularly sampled time series is addressed by Naul et al. (2018). To deal with this problem, time intervals between measurements are used as additional input to the RNN. Becker et al. (2020) also experiment with time intervals between measurements as inputs to their model, and additionally use differences between flux values as inputs instead of their raw values. They used GRU over LSTM, because of its roughly similar performance, but lower computational requirements. Within machine learning literature, other methods have been proposed to deal with irregularly sampled time series. One of these methods is the ODE-RNN (Rubanova et al., 2019; Chen et al., 2018), which is an RNN based on ordinary differential equations, which allow time to be treated as continuous instead of discrete. Although this architecture seems beneficial for our purposes, it comes at the cost of altering of the training process as opposed to using a standard RNN, in which batched training does not come straightforward. Moreover, the benefits of the ODE-RNN mostly come forward if the sampled input data is sparse. In the task of transit detection, however, this is not the case as we generally have transit durations in the order of hours and an observation cadence in the order of minutes.

Unrelated to exoplanet science or light curves, several other works investigate the use of RNNs for the detection of signals in time series. For example, LSTMs have been used to detect anomalies in time series in an unsupervised learning approach (Malhotra et al., 2015). Their RNN is trained to predict subsequent steps of "normal" data, which is clear of anomalies, and a deviation from the predictions is considered as anomaly. The same approach can be used to detect collective anomalies (Bontemps et al., 2016), i.e. anomalies that cover multiple data points. Transit signals can be seen as collective anomalies in time series dominated by "normal" stellar activity. However, as noted by Cherdo et al. (2020), this "unsupervised" approach still requires knowing which light curves are clear of anomalies. We may never know for sure if a light curve is clear of transit signals, but in some cases we do know for sure whether a transit signal is present. Therefore, we expect a supervised learning approach to be better suited for our task.

Chapter 3

Methodology

The transit detection algorithm consists of two main components: the RNN and the method to determine the period P and epoch t_0 for candidate transit signals using the RNN outputs. In the first sections of this chapter, we describe the basis network, its training, and the methods we use to determine P and t_0. Sections 3.3 to 3.5 describe how the basis RNN is extended to allow for filling missing values in light curves, confidence estimation over the RNN outputs, and providing learned representations of data patterns and signals in addition to its standard outputs. The last section of this chapter describes which data are used for the experiments.

3.1 Network

The basis network architecture, training objective and procedure are described in the following sections. The main task of the network is to classify individual data points as signal or non-signal, where every data point within the duration of a transit is considered as part of the signal. Extensions, such as the prediction of flux values within data gaps, are described in separate sections, as these require slightly different network architectures and training. A visualization of the network is given in Figure **??**. All of the following was implemented using PyTorch (version 1.8.1).

3.1.1 Architecture

Since light curves may contain thousands of data points, we avoid the "vanilla"-RNN, which is known to suffer from vanishing gradients. Other recurrent cells considered were GRU and LSTM. We found GRU to produce better and more stable results than LSTM, which is why we used GRU in our basis RNN design. Both directions in time are relevant for the classification of a data point. For example, both ingress and egress are strong features of a transit signal. This means that at mid-transit, knowing information from both directions could make it easier to classify the data points, compared to only knowing the past. For this reason, we make use of a bidirectional RNN, i.e. bi-GRU.

In order to classify each time step as signal or non-signal, the output of the network should be a single value at each time step. To achieve this, we apply fully connected (FC) layers to the outputs of the recurrent layers, which project the hidden representation at each time step down to a single node. The result is passed through the sigmoid activation function, so the outputs lie between 0 and 1. Between fully connected layers, the ReLU activation function is applied to introduce non-linearities. For most experiments, we used a bi-GRU with a single recurrent layer consisting of 64 hidden nodes, and two fully connected layers, both consisting of 64 nodes. This network was obtained by informal hyperparameter tuning, and comparison against different architectures (see Section 4.2).

3.1.2 Training

In a supervised fashion, we train the network to classify data points as signal or non-signal. This requires knowing the ground truth of the data. In our case we use simulated data, so we have access to the ground truth. Alternatively, one could use known (non-)transit signals in real-world light curves as ground-truth, or inject simulated transit signals in real-world light curves.

Typical light curves, however, contain too many data points to be used for training directly. First of all, the training procedure requires storing the gradients at each time step which would quickly overflow memory. Second of all, the RNN cannot well learn dependencies over thousands of time steps, because even LSTM and GRU have forget and reset gates, so using full-length light curves for training would be unnecessarily inefficient. Therefore we assume that the data used for training consists of light curve segments, obtained by splitting the original light curve in segments of equal length N. N should be chosen larger than the typical transit duration

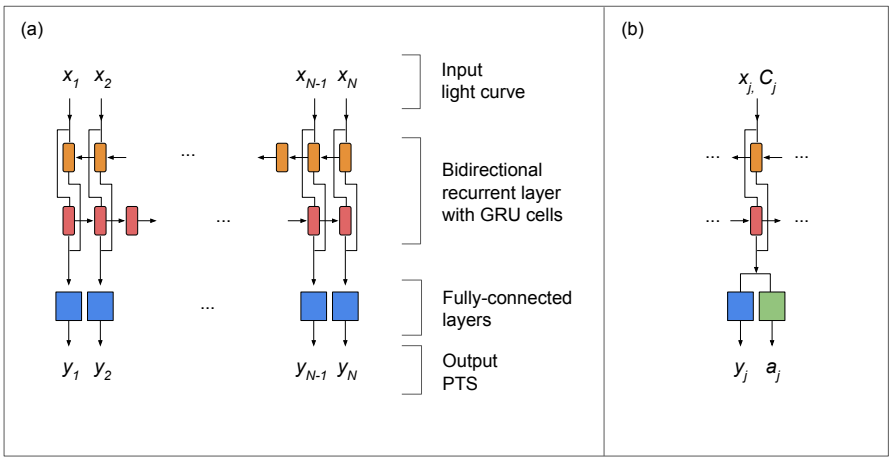
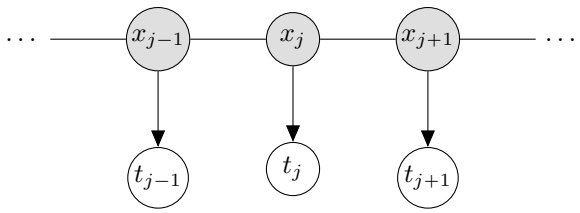


Figure 3.1: (a) The basis RNN which takes as input $x_1, \ldots, x_N$ and outputs $y_1, \ldots, y_N$. Colors indicate which blocks share the same weights and arrows indicate the flow of information. The outputs of both components of the bidirectional recurrent layer are concatenated before they are passed to the fully-connected (FC) layers. (b) The network is extended in several ways. First, we may use additional inputs C_j, e.g. centroid data, at each time step j. Second, we may add an additional network of FC-layers, to obtain an additional outputs a_j, e.g. confidence values, or representation vectors.

(∼6 hours) so sufficient background activity is included in the segments, but small enough to allow for efficient training. This work uses $N = 1500$ for the training of the network, which corresponds to 50 hours for a cadence of 2 minutes, unless stated otherwise.

The network is used to model $p(T|X)$, where T represents the targets $\{t_j\}_{j=1}^N$ for each data point $\{x_j\}_{j=1}^N$ in a given light curve segment X. We assume the network inputs to be preprocessed such that the flux values x_j can take any value, i.e. $x_j \in \mathbb{R}$. A target t_j is 1 if x_j is part of a transit signal, otherwise it is 0. We assume that the target t_j depends on all flux values through x_j, which is dependent on its neighbours $x_{j\pm1}$, which are dependent on their neighbours, and so on. This can be visualized by:



Since we observe X, the targets become separated in the graphical model. Therefore we model them as independent, given X, i.e.:

$$p(T|X) = p(t_1, \ldots, t_N|X) = \prod_{j=1}^{N} p(t_j|X) \tag{3.1}$$

In other words, we assume that knowing the a target t_k at time step $k \neq j$ in addition to the flux values X, would not make a difference for the probability of observing $t_j = 1$, compared to only knowing X. Therefore, the only information we feed to the network at each time step is x_j.

Our network outputs a value $y_j \in (0, 1)$ for each x_j, using the function $Y = f(X, W)$, where W represents the weights of the network. Y represents the collection of outputs $\{y_j\}_{j=1}^N$. We interpret y_j as the conditional probability $y_j = p(t_j = 1|X, W)$, such that $p(t_j = 0|X, W)$ is given by $1 - y_j$. The conditional distribution of targets at time step j is then given by:

$$p(t_j|X, W) = y_j^{t_j}(1 - y_j)^{1-t_j} \tag{3.2}$$

Given a set of independent observations $\{T_i\}_{i=1}^M$ for light curve segments $\{X_i\}_{i=1}^M$, we aim to maximize the likelihood $p(T|X, W)$ by updating the weights W of the network. Equivalently, we can maximize the log

likelihood, which is given by:

$$\log p(T|X,W) = \sum_{i=1}^{M} \log p(T_i|X_i,W) \tag{3.3}$$

$$= \sum_{i=1}^{M} \log \prod_{j=1}^{N} p(t_{ij}|X_i,W) \tag{3.4}$$

$$= \sum_{i=1}^{M} \sum_{j=1}^{N} \log[y_{ij}^{t_{ij}}(1-y_{ij})^{1-t_{ij}}] \tag{3.5}$$

$$= \sum_{i=1}^{M} \sum_{j=1}^{N} t_{ij} \log y_{ij} + (1-t_{ij}) \log(1-y_{ij}), \tag{3.6}$$

where t_{ij} represents the target at time step j in light curve i and $y_{ij} = y(X_i, W)_j$.

To define our loss function, which we aim to minimize, we take the negative of $\log p(T|X,W)$, averaged over the number of samples M and the number of data points per sample N, i.e. for a single light curve segment i we have the loss:

$$\mathcal{L}_{\text{BCE},i} = \frac{1}{N} \sum_{j=1}^{N} l_{ij} + (1-t_{ij}) \log(1-y_{ij}), \tag{3.7}$$

where we used $l_{ij} = t_{ij} \log(y_{ij})$, and the subscript BCE because this is known as the binary cross-entropy loss. Using this loss function for a batched training with M light curve segments per batch, we get:

$$\mathcal{L}_{\text{BCE}} = \frac{1}{M} \sum_{i=1}^{M} \mathcal{L}_{\text{BCE},i}. \tag{3.8}$$

In terms of individual data points, however, we are dealing with a highly unbalanced data set, because there are many more non-signal points than signal points. In order to deal with unbalances, we use a different definition of l_{ij}:

$$l_{ij} = p_t w_{ij} t_{ij} \log(y_{ij}). \tag{3.9}$$

[TODO: references of loss weighting] Here p_t is the positive weight, which is used to tune the weight of positive samples, i.e. the data points which are part of a transit signal. If $p_t > 1$, then the predictions over signal data points will be given extra weight in the loss function. For a given classification threshold, increasing p_t will increase the recall, but will also lower the precision. In addition to the weight p_t, which is the same for all positive samples, we explore the use of transit-specific weighting. It might be that the data set contains more shallow than deep transit signals, in which case we may choose to increase the weight of data points belonging to deep transit signals in the data set, and vice versa. This can be done by letting the weight w_{ij} depend on the depth δ_{ij} of the transit signal at time step j. For example, we can set $w_{ij} = \delta_{ij}/\sigma_i$, where σ_i is the estimated level of white noise in light curve i, so w_{ij} depends on the relative transit depth rather than absolute transit depth. In this case, predictions over data points belonging to a transit signal which is twice as deep as another signal in the same light curve, will get twice the weight in the loss function. In case both p_t and w_{ij} are used, a small adjustment to p_t needs to be made for maintaining consistent results between different choices for w_{ij}. The new weight for signal data points during training becomes $p_t' = p_t \cdot \sum_{ij} t_{ij} / \sum_{ij} w_{ij}$, where for $w_{ij} = t_{ij}$ we have $p_t' = p_t$.

Finally, we found that applying weight decay helped to prevent exploding gradients in the GRU. Simply clipping gradients above a certain value would sometimes fail when gradients jumped to infinity in a single training step, causing the training stop before they could be clipped. Weight decay is used as implemented in PyTorch, i.e. decoupled weight decay from Loshchilov and Hutter (2017). A weight decay with tuning parameter $\alpha = 5 \times 10^{-5}$ was sufficient in most cases.

3.2 Detection algorithm

In case the search for signals is directed towards monotransits, the network outputs for a given input light curve can directly be used for detection. For example, one can set a threshold on peaks in the PTS, which can be considered as candidate detections. In case we wish to search for repeating signals, the RNN outputs alone are not enough. In order to compare the performance of an RNN-based detection algorithm with conventional methods such as BLS, we also need to determine the period and epoch of the signal. Two different algorithms are tested to do so, which only make use of the RNN outputs, i.e. the PTS, for a given input light curve.

3.2.1 PTS-Peak

As proposed by Pearson et al. (2018), one could use the distances between peaks in the PTS to determine the period of a repeating signal. Here we implement this idea and refer to the algorithm as PTS-Peak. In order for this algorithm to work well, we need to take into account the presence of potential false detections in the PTS, or multiple detections of signals belonging to different planets. Therefore we adopt several rules and filtering steps to ensure consistency between matched peaks and resulting parameter estimates. The algorithm we used takes the following steps:

1. The PTS is smoothed using a one-dimensional Gaussian filter with a standard deviation of 9 data points, and peaks above a pre-set threshold are identified and indexed.

2. For each peak i, the width of the peak above the threshold is used as candidate transit duration τ_i and the central point of the peak as candidate mid-transit time $t_{\mathrm{mid},i}$. For each pair of peaks $\{i,\, j\}$, where $j > i$, the candidate period is determined by $P_{i,j} = t_{\mathrm{mid},j} - t_{\mathrm{mid},i}$.

3. For each pair of peaks $\{i,\, j\}$, a set is c formed consisting of the indices of peaks that may correspond to signals from the same planet. The peak k that has $t_{\mathrm{mid},k}$ closest to time $t_{\mathrm{exp}} = t_{\mathrm{mid},j} + P_{i,j}$ is added to the candidate signal set $c = \{i,\, j\}$, if $t_{\mathrm{mid},k}$ is within three hours of t_{exp}. We then get the candidate $c = \{i,\, j,\, k\}$, which is used to update the expected period, i.e. $P_{i,j,k} = (t_{\mathrm{mid},k} - t_{\mathrm{mid},i})/2$. Iteratively, the candidate set is expanded with the indices of peaks that lie within the expected range, until $t_{\mathrm{exp}} - 3\mathrm{h}$ is larger than the maximum time in the PTS. Candidate sets that have missing peaks at times where you would expect them are rejected and the rest is passed to the next step. Note that this would filter out detections if only a single peak is missing in the PTS. However, such signals can still be retrieved by evaluating the harmonics of a detected signal, as is described in step 6.

4. For each candidate c, the candidate duration τ_c is defined by the median duration of the individual peaks, the period P_c by the median distance between peaks and the $t_{0,c}$ by the median of the epoch expected from subtracting P_c from each individual $t_{\mathrm{mid},i}$. A candidate is rejected if τ_c is smaller than 15 minutes or if $P_c < P_{\mathrm{min}}$ for a pre-set minimum period P_{min}.

5. Each candidate is then given a score. For each individual event $n = \{0, \ldots, E-1\}$ in the candidate signal defined by $t_{0,c}$, τ_c and P_c, i.e. $t_{0,c} + nP_c - \frac{1}{2}\tau_c < t < t_{0,c} + nP_c + \frac{1}{2}\tau_c$, the maximum in the PTS is recorded as $y_{n,\mathrm{max}}$. The candidate score is then given by:

$$score = \frac{1}{\sqrt{E}} \sum_{n=0}^{E-1} y_{n,\mathrm{max}}, \tag{3.10}$$

   where the square root is used to prioritize detections with more individual events over detections with similar scores $y_{n,\mathrm{max}}$, but fewer events. Only the best scoring candidate is passed to the next step.

6. Finally, the harmonics of the candidate signal are evaluated to allow signals to be detected which have missing peaks in the PTS. For each harmonic $h = \{2, 3, \ldots\}$, the score is determined similarly to step 5, but with P_c/h and corresponding epoch $t_{0,h}$ instead of P_c and $t_{0,c}$. h is iteratively increased until $P_c/h < P_{\mathrm{min}}$. The best score and the corresponding parameters are passed to the next step.

7. To search for multiple planets in a single light curve, the events from the best scoring candidate are masked in the PTS and the process is repeated. The best score and corresponding parameters after each iteration are returned as potential detection. The score is used to set a detection threshold.

3.2.2 PTS-Fold

It could be that individual transit events are missed by the RNN, or that peaks in the PTS are too weak to be taken into account by PTS-Peak. To be less dependent on distinguishable peaks in the PTS, but more on overall response to transit signals, we define another algorithm which we refer to as PTS-Fold. This algorithm is similar to many existing detection algorithms (e.g. phase dispersion minimization or BLS) in that it folds the input time series over a set of trial periods. However, whereas other algorithms fold the raw light curve over trial periods, PTS-Fold only folds the PTS. Since each value the PTS indicates the extend to which a transit signal might be present at the corresponding time step, we can efficiently compute a candidate detection score for repeating signals by aggregating overlapping data points in the folded PTS. To clarify, the algorithm used takes the following steps:

1. The PTS is smoothed using a one-dimensional Gaussian filter with a standard deviation of 9 data points.

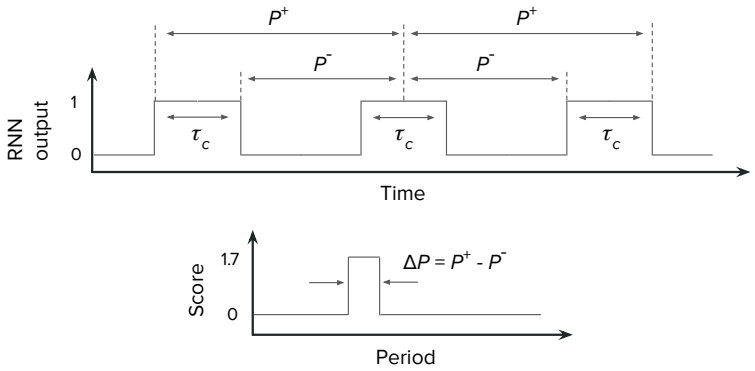


Figure 3.2: An illustration of how the width ΔP of a peak in the periodogram relates to the duration τ_c of a signal. The PTS shown in the top shows $E = 3$ potential events of a signal with a minimum and maximum period of P^- and P^+. In this case, the candidate score for each value between P^- and P^+ is the same in the periodogram shown below. For a number of events E in the PTS, we have $\tau_c = (E-1)\frac{1}{2}\Delta P$, which in this case translates to $\tau_c = P^+ - P^-$.

2. A set of trial periods is determined based on the time steps in the PTS. Our input is of 2-minute cadence, so the trial periods are exactly multiples of 2 minutes. The differences between subsequent trial periods increases roughly linearly with the value for the trial period to reduce computational requirements.

3. For each trial period, the PTS is folded such that at each point in the phase curve consists of a set of overlapping points.

4. For each fold, each point in the phase curve is given a score by aggregating the individual overlapping PTS values, i.e. for a single point with overlapping values $y_0, \ldots, y_{E-1}$:

$$score = \frac{1}{\sqrt{E}} \sum_{n=0}^{E-1} y_n. \tag{3.11}$$

   The maximum scoring point and corresponding value for epoch for each fold is passed to the next step.

5. The result, i.e. the maximum score for each trial period, defines the PTS-Peak periodogram and can directly be used to set a detection threshold. For example, a peak in the periodogram above a given threshold may indicate a detection with corresponding parameters for the period and the epoch.

6. To search for multiple planets in a single light curve, we mask the previously detected signal in the PTS and repeat the process. However, in order to mask detected signals, we need to have an estimate of the duration of the individual events. For efficiency, we avoided the determination of the duration during search. To obtain a rough estimation for the duration but maintain this efficiency, we therefore use the width ΔP of the peak in the periodogram at half of its maximum, and translate this to the candidate duration according to $\tau_c = (E-1)\frac{1}{2}\Delta P$, where E is the number of events belonging to the detected signal. The relation between the peak width and the duration of the signal is illustrated in Figure 3.2.

3.3 Handling data gaps

If we naively pass a light curve with missing values through the RNN, then varying time intervals will be ignored and false detections could be triggered around gaps. In this research, we evaluate different gap filling approaches to deal with this problem. Since the RNN offers the possibility to make predictions at every time step, we explore the ability of the RNN to predict missing flux values when applied to a light curve with gaps. In other words, we extend the RNN and train it to predict subsequent flux values, in parallel to classifying data points as signal or non-signal. To do so, we apply a second FC-network to the outputs of the recurrent layers, to produce an output $\hat{x}_{ij}$ for each data point x_{ij}. We use the same amount of layers and hidden nodes for this FC-network as for the one used for the binary classification of data points. The only difference is that the output is not passed through the sigmoid function, because the preprocessed flux can have any value in $\mathbb{R}$. We use an additional loss term so the generative network is trained to improve its predictions such that $\hat{x}_{ij}$ gets closer to x_{ij}:

$$\mathcal{L}_{\text{MSE},i} = \frac{1}{N} \sum_{j=0}^{N-1} (x_{ij} - \hat{x}_{ij})^2. \tag{3.12}$$

The loss over light curve segment i for the generative RNN then becomes:

$$\mathcal{L}_i = \mathcal{L}_{\text{BCE},i} + \lambda \mathcal{L}_{\text{MSE},i}. \tag{3.13}$$

When this network is applied to a light curve with missing data, we a replace missing value $x_{ij} = \text{NaN}$ with the predicted value $\hat{x}_{ij}$ and use it as input to the network instead.

3.4 Increasing interpretability with confidence estimation

Although the basis network outputs are conveniently bounded between 0 and 1, they do not equal true probabilities. For example, when two peaks in two PTS have the same height, we cannot say that they are equally likely to correspond to true signals. This is because the two input PTS might be slightly different than the data seen during training, which could cause the outputs of the RNN to be less reliable. In addition to the standard outputs, it would therefore be helpful to have an estimation of confidence over the predictions to know when the network is less or more certain.

To this end, we make use of a simple approach proposed by DeVries and Taylor (2018). We extend the network from Section 3.1.1 by applying additional FC-layers to the outputs of the recurrent cells, which output a scalar c_{ij} for each time step j in light curve i. This value, which is mapped between 0 and 1 by the sigmoid function, represents the confidence of the network over the prediction y_{ij}. Since c_{ij} is still the output of a neural network, it should not be interpreted as confidence in the statistical sense, but rather as indication of confidence relative to other predictions.

c_{ij} is used in the loss function by replacing y_{ij} with $y'_{ij} = c_{ij} y_{ij} + (1 - c_{ij} t_{ij})$ in Equation 3.7. This means that for low confidence, i.e. $c_{ij} \approx 0$, we get that the network prediction during training becomes the same as the target, $y'_{ij} \approx t_{ij}$. For high confidence, $c_{ij} \approx 1$, the network uses its own prediction in the loss function, $y'_{ij} \approx y_{ij}$. To prevent the network from always returning low confidence values, an additional term is added to the loss function:

$$\mathcal{L}_{\text{conf},i} = \frac{1}{N} \sum_{j=0}^{N-1} -\log(c_{ij}), \tag{3.14}$$

so the loss over a given light curve i for the confidence RNN becomes:

$$\mathcal{L}_i = \mathcal{L}_{\text{BCE},i} + \lambda \mathcal{L}_{\text{conf},i}, \tag{3.15}$$

where λ is used as a weighting parameter. In line with DeVries and Taylor (2018), we adopt a budget parameter β which is used during training to adjust λ. If $\mathcal{L}_{\text{conf}} > \beta$, then we increase λ, and if $\mathcal{L}_{\text{conf}} < \beta$, we decrease λ. Lastly, only for half the samples we use y'_{ij} instead of y_{ij}, in order to encourage the network more to learn meaningful decision boundaries.

3.5 Separating signals from different planets

Another way to increase the interpretability over the network outputs is to utilize the network's representations of each data point. This comes in helpful if there are only a few potential transit signals in a given light curve. Based on the PTS only, we cannot tell whether the individual signals appear different from each other, apart from their durations. The detection algorithm might thus match the wrong signals together, leading to an incorrect detection which may been prevented if signal shapes are taken into account.

To illustrate how one could use an RNN and still take into account different signal shapes, we extend the network to learn representations of signals, such that signals from different planets are represented differently. The network from Section 3.1.1 is extended with additional FC-layers which are applied to the outputs of the recurrent cells. These layers project the outputs of the recurrent layer at time step j down to D dimensions into a vector R_{ij} for in light curve segment i. For different candidate transit signals, say, A and B, covering time steps $\{k, \ldots, l\}$ and $\{m, \ldots, n\}$ respectively, the network outputs representations $\{R_{ik}, \ldots, R_{il}\}$ and $\{R_{im}, \ldots, R_{in}\}$. We can average both sets of representations to obtain R_{A} and R_{B}, which correspond to the aggregated representations of signal A and signal B respectively.

In the supervised learning approach, we know whether A and B are signals caused by the same planet or not. Assuming that both A and B are true detections, we call A and B a positive pair if they are caused by the same planet, and a negative pair if they are signals from different planets. Actually, the network outputs a representation for each time step, regardless of whether a signal was detected at that time step. Therefore, R_{A} and R_{B} may as well be the aggregated representations of any two disjunct parts of the input light curve. A and B are thus also called a positive pair if they both cover a non-signal part of the light curve, and they are also called a negative pair if only one of the two corresponds to a transit signal and the other does not.

The network is trained to represent positive pairs similarly, and negative pairs differently. This is achieved by adding a score of similarity between R_{A} and R_{B} to the loss function. The score of similarity used in this work is the cosine of the angle $\theta_{\mathrm{A,B}}$ between R_{A} and R_{B}: [TODO: add cosine similarity source]

$$\mathrm{sim}(\mathrm{A},\mathrm{B}) = \frac{R_{\mathrm{A}} \cdot R_{\mathrm{B}}}{\|R_{\mathrm{A}}\|\|R_{\mathrm{B}}\|} = \cos(\theta_{\mathrm{A,B}}). \tag{3.16}$$

For smaller angles between the representation vectors, A and B are considered to be more similar.

During training, we present the network with pairs of light curves with the same stellar background, and pre-selected regions A from the first light curve and B from the second. These could be true transit signals, but could also be background noise. The parameter $p_{\mathrm{A,B}}$ indicates whether A and B are a positive or a negative pair. The network is then trained to minimize the representation loss term given by:

$$\mathcal{L}_{\mathrm{repr},j} = (\mathrm{sim}(\mathrm{A},\mathrm{B}) \cdot (1 - 2p_{\mathrm{A,B}}) + 1)/2. \tag{3.17}$$

The factor including $p_{\mathrm{A,B}}$ within brackets is to ensure that a positive pair with a high similarity score results in a low loss, a negative pair with high similarity results in high loss, and vice versa. The addition of 1 and subsequent division by 2 is to ensure that this loss term always has a value between 0 and 1. The combined loss for light curve pair i is then:

$$\mathcal{L}_i = \mathcal{L}_{\mathrm{BCE},i} + \lambda \mathcal{L}_{\mathrm{repr},i}, \tag{3.18}$$

where λ is used to weigh the representation term. In this case $\mathcal{L}_{\mathrm{BCE},i}$ is the average of the BCE loss term over both light curve segments in the input pair.

3.6 Data simulations

For the development and evaluation of our RNN-based algorithm we used simulated data. This is because simulated data comes with an absolute ground-truth of the hidden signals, and for the development of the algorithm it showed useful to be in control over the parameters of the input light curves. In contrast, real-world data cannot be changed, and in some cases the ground-truth is ambiguous. Realizing that a single source of simulated data may limit our results, two sources of data were used. First, our Light Curve Simulator, or LCSim, was specifically designed for this work. Second, we made use of simulated light curves in the Lilith-4 data set, which was produced by the Lilith data simulator of the TESS pipeline. The following subsections describe the details of the simulations, the data sets used in this work, and some general preprocessing of the data.

3.6.1 LCSim

In line with TESS data, we adopt a 2-minute cadence for all simulations in this work. The simulator, made available here[1], may also be used to generate 30-minute cadence light curves, to obtain light curves that are closer to Kepler data.

Stellar variability is simulated using Gaussian processes (GPs). GPs have been used several times in literature to account for background activity in light curves, both for characterizing and simulating stellar activity (Barros et al., 2020; Zucker and Giryes, 2018). GPs are defined by a mean and a covariance function, or kernel. In our case, the mean function is always 1, so the behaviour is fully determined by the kernel, which should therefore hold all of the star's relevant properties. An instance of the function representing stellar variability can be obtained by sampling from the multi-dimensional Gaussian distribution that is defined by the mean and covariance function. However, if we use the same kernel for each sample, then each light curve has the same underlying stellar properties. Therefore, we construct a kernel for each light curve. To construct a kernel that is able to simulate quasi-periodic behaviour as observed in stars, we make use of the Python library `celerite` (Foreman-Mackey et al., 2017). [TODO: cite use cases of celerite] `celerite` has built-in kernels to simulate stellar granulation and rotation modulation. These kernels are specified by the standard deviations and time scales of the processes, among several other parameters which are given in Table 3.6.1. The two oscillation terms describing stellar rotation are partly defined by Q_0, dQ and f which respectively describe the quality factor of the secondary oscillation, the difference between quality factors of both oscillation modes, and the fractional amplitude of the secondary mode compared to the primary (see the documentation[2]). P_{rot} and σ_{rot} describe the primary period of the rotation and the standard deviation of the variability respectively. For the granulation term, we have P_{gran} and σ_{gran} to describe the period and standard deviation, and Q the quality factor of the oscillation. For granulation it is standard to choose $Q = 1/\sqrt{2}$ [TODO: cite].

[1] https://github.com/ykerus/transit-detection-rnn15

[2] https://celerite2.readthedocs.io/en/latest/api/python/

To simulate photon noise, we sample from a Gaussian distribution $\epsilon_{ij} \sim \mathcal{N}(0, \sigma_i)$ and add ϵ_{ij} to the corresponding flux at time step j in light curve i. The standard deviation σ_i (σ in Table 3.6.1) defines the level of time-independent noise in the light curve, and is specified for each light curve separately.

For the simulation of transit signals, we made use of the Python library `batman` (Kreidberg, 2015), which is based on the equations from Mandel and Agol (2002) which describe the physics of transits. We approximate the stellar limb darkening effect with the built-in quadratic function, parametrized by u_1 and u_2. The values for these parameters were sampled using intermediate parameters q_1 and q_2 such that they always produce physical transit signals, according to Kipping (2013a). We constrained the orbital period P of a planet and its semi-major axis a according to Kepler's Third Law (see Section 2.1). In order to do so, we also specify the stellar mass M and radius R so the result is compatible with the input requirements of `batman`. The orbital inclination i is assumed to be 90° for all planets we simulate, to avoid problems with non-existent transit signals, or transit depths that are more difficult to predict. Setting $i = 90°$ ensures that a simulated planet moves in front of the stellar disk during transit. The eccentricity *ecc* is sampled from a Beta distribution, following Kipping (2013b). The radius of the planet is specified by *ror*, which is fractional radius of the planet relative to its host star. Lastly, w is specified to indicate in which part the exoplanet is in its orbit when it transits its host star. If $ecc = 0$, then w has no effect, otherwise $w = 90°$ will correspond to the shortest possible transit duration, and $w = 270°$ to the longest.

In case a light curve is required to have transit signals from multiple different planets, the transit simulator is simply called iteratively with different parameters for the planets while maintaining all the parameters belonging to the star. The result might be that the two planets coincidentally have the same distance from their host star, which would be unrealistic, but does not pose a problem for the purpose of this thesis. Overlapping transit signals, on the other hand, could make the process of developing and evaluating our detection algorithm more difficult. If overlapping signals have been found, one needs to make sure which signal triggered a detection: it could be one of the two, or both. Since overlapping signals are far less common in real-world data than non-overlapping signals, we only simulate non-overlapping transit signals in this work to avoid confusion.

Several data sets used in this work are based on LCSim. LCSim-1500 is a set of light curve segments consisting of $N = 1500$ data points per light curve, which is used for training and evaluating the network. These light curves are supposed to imitate light curve segments that are obtained from splitting an original full-length light curve into parts. However, with LCSim we simulated the segments directly to reduce computational costs, so we do not have access to the "original" light curves. LCSim-500 is similar, only it has fewer ($N = 500$) data points per light curve. Both data sets contain 15000 training samples, 5000 validation samples and 5000 test samples. Half of the samples per split contains no transit signals, 35% contains a single transit signal and 15% two transit signals. For the evaluation of our method's ability to retrieve signals in full-length light curves, we use LCSim-Mono and LCSim-Single. Both data sets consist of 5000 light curves spanning 27.4 days (i.e. 19728 data points per light curve). 50% of the samples in LCSim-Mono contain a single transit signal, and 50% of the samples in LCSim-Single contain at least three transit signals from a single transiting planet. None of these data sets contain data gaps. However, to evaluate the effect of gaps and different gap-handling approaches, we use LCSim-1500-Gap, which is LCSim-1500 with injected gaps. Zero, one or two large gaps between 2 and 10 hours are injected with 50%, 35% and 15% probability respectively. In addition, a random selection of 2% of the data points is removed. For the training of the representation RNN, we generated pairs of white-noise dominated light curves with the same distribution of transit signals as for the other data sets. 50% of the samples was a positive pair, and 50% a negative pair, as explained in 3.5.

Each light curve is directly simulated as median normalized, i.e. centered around 1. Prior to applying the RNN, we always center the inputs around zero by subtracting 1, and standardize the entire data set. To do so, for each data point we subtract the mean and divide by the standard deviation over all data points in the training split. In Chapter 4, we experiment with additional preprocessing steps.

3.6.2 Lilith-4

The Lilith-4 data set comprises four sectors of simulated TESS data, and takes into account readout errors, spacecraft jitter, focus errors, diffuse light, cosmic rays, stellar variability, transiting exoplanets, eclipsing binary stars, and more (Smith et al., 2019). This simulated data set is therefore expected to be considerably more realistic than ours, which is why we include it in this research. Furthermore, it allows for the evaluation of certain preprocessing steps in combination with our algorithm, that would be necessary in the case of using real-world data. Lilith-4 also includes information about the pointing of the telescope, the centroid data, which can be used by our algorithm to potentially benefit from.

We prepared two data sets based on Lilith-4. As explained in Section 2.3.5, we excluded light curves with (B)EB signals. The first data set, Lilith-1500, is similar to LCSim-1500. Lilith-1500 is obtained from median normalizing the light curves that only span a single sector in Lilith-4, and subsequently splitting them into equally sized segments of $N = 1500$ data points. Missing data points are replaced with NaN values, and slight deviations from a 2-minute cadence are corrected for. The segments were chosen to be partially overlapping,

	Parameter	Sampling distribution or relation	Units
Photon noise	σ	$\exp(\text{Uniform}(\log(0.0005), \log(0.003)))$	-
Stellar rotation	Q_0	$\text{LogNormal}(0, 2)$	-
	$\mathrm{d}Q$	$\text{LogNormal}(0, 2)$	-
	f	$\text{Uniform}(0.1, 1)$	-
	P_{rot}	$\lvert\text{Normal}(5, 2)\rvert + 1$	days
	σ_{rot}	$\exp(\text{Uniform}(\log(0.0003), \log(0.005)))$	-
Stellar granulation	ν	$\text{LogNormal}(4.5, 1) \cdot 10^6$	Hz
	P_{gran}	$1/\nu/86400$	days
	σ_{gran}	$\text{LogNormal}(\log(2 \times 10^{-6} \cdot \nu^{-0.61}), 0.1)$	-
	Q	$1/\sqrt{2}$	-
Star	M	$\lvert\text{Normal}(0.9, 0.25)\rvert + 0.1$	$\mathrm{M}_\odot$
	R	$\lvert\text{Normal}(M^{1/3} - 0.1, 0.2)\rvert + 0.1$	$\mathrm{R}_\odot$
	q_1, q_2	$\text{Uniform}(0, 1)$	-
	u_1	$2q_2\sqrt{q_1}$	-
	u_2	$(1 - 2q_2)\sqrt{q_1}$	-
Exoplanet	P	$\exp(\text{Uniform}(\log(P_{min}), \log(P_{max})))$	days
	a	$((M \cdot \mathrm{M}_\odot) \cdot 86400P \cdot G/(2\pi)^2)^{1/3}/(R \cdot \mathrm{R}_\odot)$	R
	ror	$\exp(\text{Uniform}(\log(0.02), \log(0.15)))$	-
	ecc	$\text{Beta}(0.867, 3.03)$	-
	i	90	deg
	w	$\text{Uniform}(0, 360)$	deg

Table 3.1: Sampling distributions for the parameters used in LCSim (see main text for their descriptions). Inspiration for these distributions came from online tutorials, e.g. from `exoplanet` (Foreman-Mackey et al., 2021) for Q_0, $\mathrm{d}Q$ and f; literature, e.g. Martins et al. (2020) for stellar rotation, Kallinger et al. (2014) for granulation, Kipping (2013a) and Kipping (2013b) for efficient sampling of ecc, u_1 and u_2; and parameter values from known transiting planets and real-world light curves from TESS (Ricker et al., 2014) for remaining planetary and stellar parameters and photon noise. $\mathrm{M}_\odot = 1.989 \times 10^{30}$ kg and $\mathrm{R}_\odot = 6.963 \times 10^8$ m are the solar mass and radius, G is the gravitational constant, and P_{min} and P_{max} a pre-defined minimum and maximum orbital period of the planet. Note that the relations between parameters are in some cases not realistic. For example, M and R are only sampled to get a reasonable and slightly random relation between P and a, but themselves have no other function in the simulation. However, for the purpose of our research this is no problem as we do not make explicit use of these parameters in our detection algorithm, and the transit signals are still defined by a complex interplay between the parameters specified above.

so we have more data to train the network. Segments are rejected if they contain transit signals which are less than half visible, overlapping, less than 1 or more than 13 hours in duration, or have a depth δ relative to σ (i.e. δ/σ) of less than 0.25 or more than 10. For Lilith light curves, σ is not given, so we estimate it by flattening the input with a sliding median filter with a window size of 30 minutes, and subsequently taking the standard deviation. From a total of 68236 segments, 61% was used for training, 22% for validation and 17% for testing. Segments from the same light curve were kept in the same split, to avoid overlap between training and test data. To evaluate the ability of our algorithm to detect planets in Lilith light curves, we prepared Lilith-Multi. This data set consists of 6511 light curves which were excluded from Lilith-1500, 2670 of which contained at least three transit signals from at least one planet. We used light curves of each sector separately, so we did not stitch light curves from targets which had observations over multiple sectors. Only planets with at least three transit signals in the same light curve with relative transit depths $0.25 < \delta/\sigma < 10$ were used in the evaluation. Regarding preprocessing, we take the same steps as for LCSim data. Additionally, we filtered out flagged low-quality data points, using the flags explained by Tenenbaum and Jenkins (2018).

Chapter 4

Experiments and results

The following experiments were designed to help in answering our research questions. The first section of this chapter is concerned with preprocessing of the data that is fed to the RNN, where we evaluate different scaling and gap handling approaches and their effect on the performance of the RNN in the given task. Often in classification tasks, the accuracy is used as a measure of the model's performance. However, since we are dealing with an imbalanced data set, the accuracy might give a wrong impression. This is because classifying each data point as non-signal would already result in high accuracy. Therefore we primarily look at the precision and recall, which are given by:

$$\text{precision} = \frac{tp}{tp + fp} \tag{4.1}$$

$$\text{recall} = \frac{tp}{tp + fn}, \tag{4.2}$$

where tp is the number of true positive classifications, fp the number of false positives and fn the number of false negatives. However, accuracy, precision, recall and related metrics such as F1-score are dependent on a classification threshold, which defines when a data point is classified as signal ($y_{ij} \geqslant$ threshold) or non-signal ($y_{ij} <$ threshold), For this reason, we use area under the precision-recall (PR) curve as a measure that is independent of the threshold. The area under the PR-curve is also known as average precision (AP).

Subsequently in Section 4.2, we compare different models and training schemes. This is to motivate our architecture choice for the RNN used in the detection algorithm, and to assess whether RNNs can compete with the more commonly used CNN for this task. In the sections that follow, i.e. 4.3, 4.4 and 4.5 we compare our RNN-based transit detection algorithms with box-fitting algorithms in real-world problems. These include, respectively, the retrieval of monotransits, the retrieval of the repeating transits of a single transiting planet, and the retrieval of potentially multiple planets with repeating transit signals in a light curve. Finally, we take a closer look at a few success and failure cases in Section 4.6.

4.1 Preprocessing and performance

In the following, we assume all "raw" light curves to be median normalized, and each light curve segment to be obtained from splitting this median normalized light curve into parts. The first preprocessing step we take is centering the light curve segments by subtracting 1. Additionally, we investigate the effect of scaling the light curves by their estimated noise level σ, so the network performance becomes less dependent on this noise and instead more on the transit depth relative to this noise. We will refer to this as sigma scaling.

Still, the range of flux values might greatly vary between two separate light curve segments, as illustrated in Figure 4.1. This is understood by realizing that the original light curve may contain large-scale fluctuations due to stellar rotation modulation. The network might interpret these range differences as indicators of the presence of a transit signal, in particular if there is little data available for a certain range of flux values. This is not what we want, because a transit event can occur at any given time, independent from the activity on the stellar surface. Applying median normalization again to the segments is not a solution, because then we prevent the network to learn to deal with input ranges it would normally encounter in full-length light curves. An alternative which avoids this problem, is to take the derivative of the input light curve and feed it to the network instead of the absolute values.

Figure 4.2 shows the effect of these different preprocessing steps on the AP of the RNN in the task of classifying individual data points as signal or non-signal. In this figure, we can observe a small difference between using no scaling and using sigma scaling, in favor of no scaling. We found that using derivatives

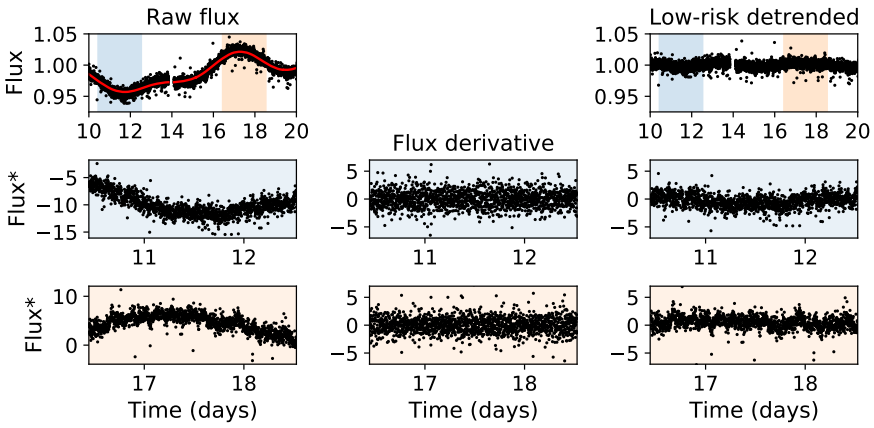


Figure 4.1: (Left) Two segments are taken from a "raw" median normalized light curve, which have considerably different flux ranges after preprocessing. (Middle) Taking the derivative solves the problem of different input ranges, while conserving all relevant information. (Right) With loss of information, one could carefully remove large-scale trends in the light curve, i.e. the red trend line in the top-left panel, to achieve a similar result.

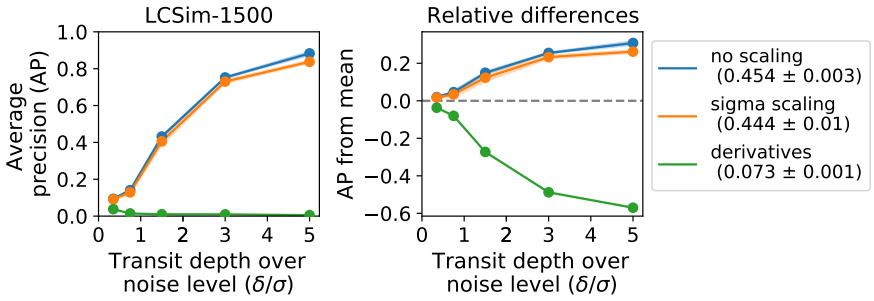


Figure 4.2: The average precision of the RNN evaluated at different transit depth bins for different basic preprocessing steps. At larger depths, the AP is expected to be higher. The right figure shows the deviations from the mean of all curves. Filled regions, which are narrow in this figure, show one standard deviation over the results of three differently initialized networks applied to the LCSim-1500 test split. Values between brackets in the legend show the overall AP and standard deviations over three runs.

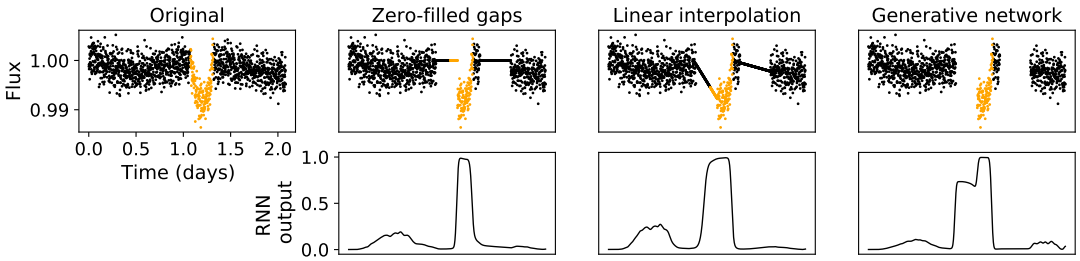


Figure 4.3: An example from the LCSim-1500-Gap validation split, with different approaches to dealing with gaps and the corresponding PTS of the RNN, i.e. the RNN's outputs at every time step. The signal data points are indicated by orange. Zero-filling in this example translates to one-filling, because the light curve segments are centered around zero before they are fed to the RNN.

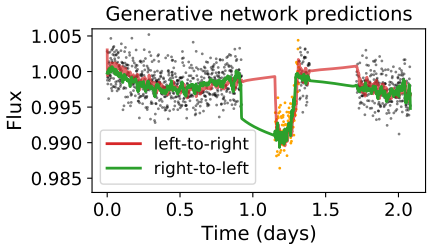


Figure 4.4: The predicted flux values visualized with red and green from the bidirectional generative RNN corresponding to the right-most figure in Figure 4.3. The network seems to follow an expected trend in both directions, however, the gaps are not filled in a satisfactory manner.

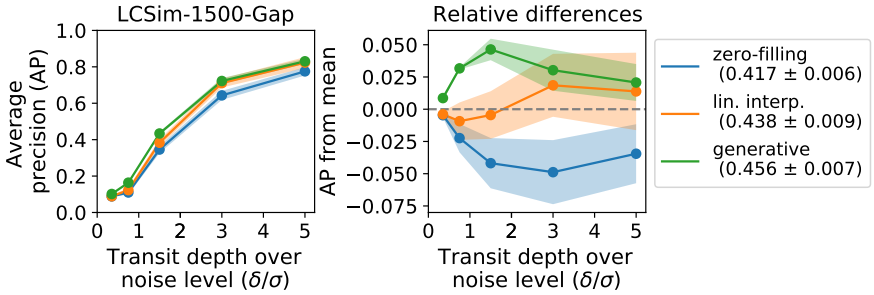


Figure 4.5: The average precision for varying transit depths, with filled regions indicating the standard deviation of three runs on the test split of LCSim-1500-Gap with differently initialized networks. The comparison includes different methods to deal with data gaps: filling gaps with zeros, linear interpolation, or using predictions from the generative RNN.

instead of absolutes prevents the network from learning[1].

To deal with gaps, we compare two gap-filling approaches and the use of the generative RNN. The gap-filling approaches are simple as we do not aim to reconstruct the original light curve, but rather produce an input that the RNN can handle well. Figure 4.3 visualizes how each approach works, and Figure 4.4 visualizes the predictions of the generative RNN applied to the same example. Since we use a bidirectional RNN, we get two streams of predictions. We can see from the figures that this creates some unrealistic artefacts in the predicted curve, which is likely the cause for the bumped peak in the PTS of the generative RNN. Therefore unexpectedly, we found that the generative network slightly outperformed the other two approaches when applied to a larger data set, especially for the shallowest transit signals. The results are plotted in Figure 4.5. The second best approach tested was linear interpolation, and the worst was zero-filling. In terms of efficiency, linear interpolation is preferred over the generative RNN. This is because during training, the generative RNN needs to evaluate at each time step whether the data point is missing, and if so, replace it with its own prediction before it proceeds to the next time step. This, in combination with the fact that we could not use pre-implemented RNN architectures in PyTorch for this purpose, resulted in an increase of training time of more than five times compared to the standard RNN applied to linearly interpolated data.

In Figure 4.6, we evaluate a subset of the same preprocessing steps applied to the Lilith-1500 data set. In the shallower transit range, it appears that sigma scaling results in worse performance than no scaling. However, for deeper transits the difference between the scaling approaches becomes larger, in favor of sigma scaling. From the figure it also seems that zero-filling is the better approach to deal with data gaps, in contrast to our earlier results.

Lastly, we evaluate additional preprocessing steps applied to Lilith light curves. First of all, to address the problem illustrated in Figure 4.3, we detrend the original light curves (before segmenting) with a filter that only removes the largest trends. We refer to this as low-risk detrending (LRD). For this, a Savitzky-Golay filter was used with a polynomial order of 2 and a window length of 2881, i.e. ∼4 days (see Hippke et al. (2019) for details on the implementation of the Savitzky-Golay filter). We also test the effect of detrending the original light curve with a filter that would more commonly be used in combination with other search algorithms. We refer to this as high-risk detrending (HRD). For this, a time-windowed sliding median filter was used with a window length of 12 hours. Another preprocessing step is that of removing outliers. In the comparison, we

[1]In an earlier iteration of this work when less complex data was used (e.g. with less stellar variability), the derivative inputs worked well. Only with increased complexity the model started to fail.

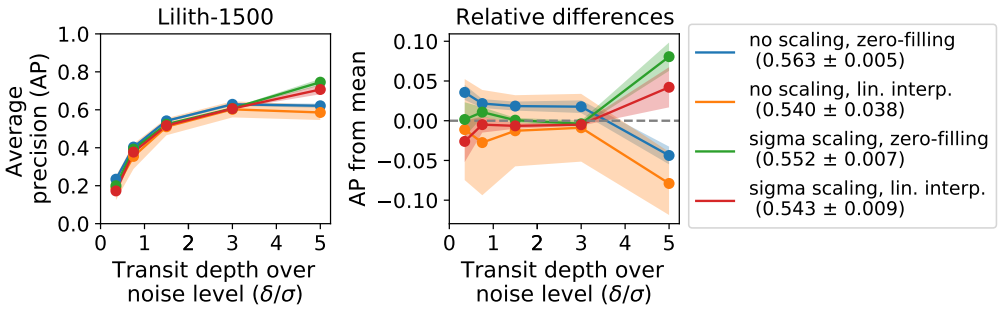


Figure 4.6: The effect of different basic preprocessing steps on the performance of the RNN applied to Lilith light curve segments. Filled regions indicate the standard deviation over three runs using differently initialized models that were applied to the test split after training. Values between brackets show the overall AP.

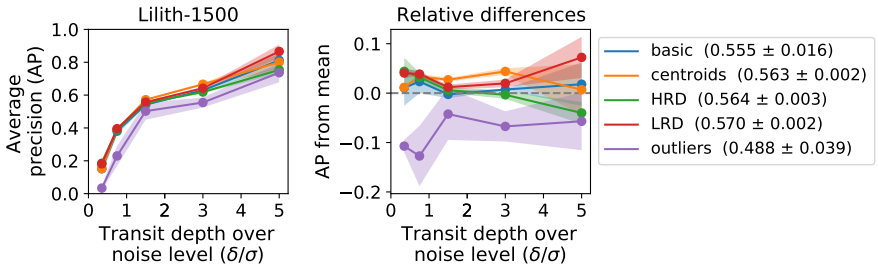


Figure 4.7: [TODO: outliers] The effect of high- and low-risk detrending, centroid inputs and outlier removal on the average precision of the RNN, in comparison with only using basic preprocessing, i.e. sigma scaling and linearly interpolating data gaps. The results are obtained on the test set of Lilith-1500. Standard deviations over three runs are indicated by the filled regions.

test the effect of removing outliers above 6σ and below 12σ from the median of a sliding 30-minute window, to avoid clipping out transit signals. Finally, since Lilith provides centroid data, we investigate the effect of using the centroids as additional input to the network at each time step. The results shown in Figure 4.7 show that outlier removal resulted in worse performance. This is possibly due to the fact that removing outliers creates missing data which need to be imputed. The results also show that LRD and centroid inputs are beneficial to the network, although there is some overlap between different runs. In the mid-range of transit depths, LRD and centroid inputs seem to be consistently better than other methods. Furthermore, without detrending the network is still able to perform comparably to using HRD and LRD, which underlines our claim that the RNN does not require its inputs to be cleared of short scale time dependent noise in order to do its task.

In each of the following experiments, we make use of sigma scaling and linear interpolation in case the data contains gaps.

4.2 Model comparison

Although the task of our network is to classify individual data points, we can make a small adjustment to the RNN so we can compare its performance with the more commonly used CNN in the task for which the CNN is generally applied. The CNN proposed by Pearson et al. (2018) was evaluated on its accuracy of classifying entire light curve segments as signal or non-signal. To obtain a single classification of the RNN for an input light curve segment, we have two options. Either we take the maximum value of the corresponding PTS as prediction and make no changes to the network or training, or we train the network to output its classification over the entire segment only at the last time step. The latter approach, we refer to as "naive" because we only provide a sparse learning signal to the network, even though we have the option to provide a learning signal at every time step.

Table 4.2 shows that the RNN (bi-GRU-1, i.e. single-layer bidirectional GRU) outperforms the CNN that was used in the comparison. It must be noted however, that the performance of the CNN depends on many hyperparameters (e.g. number of layers and channels, kernel sizes, pooling sizes, strides), and the one used here differs from the one used by Pearson et al. (2018) because it led to better results in our case. Each network architecture was obtained by changing the hyperparameters in small steps until no large improvements were

Model	Segment accuracy ($N = 500$)	Segment accuracy ($N = 1500$)	Data point accuracy ($N = 1500$)
MLP	0.6846 ± 0.0007	0.60 ± 0.01	
CNN	0.692 ± 0.008	0.61 ± 0.03	
GRU-1 (Naive)	0.64 ± 0.10	0.507 ± 0.006	
GRU-1	0.68 ± 0.02	0.64 ± 0.01	0.935 ± 0.001
bi-GRU-1	**0.719** ± 0.003	**0.681** ± 0.004	**0.9399** ± 0.0003
bi-GRU-2	0.712 ± 0.005	0.66 ± 0.03	0.938 ± 0.003
bi-LSTM-1	0.708 ± 0.005	0.59 ± 0.04	0.931 ± 0.003

Table 4.1: The comparison of different models applied to LCSim-500 and LCSim-1500, which consist of light curve segments spanning respectively 16.7 hours and 50 hours. The center columns show the test accuracy of classifying entire segments as signal or non-signal, similar to Pearson et al. (2018). The last column shows the accuracy of classifying individual data points as signal or non-signal, which is high by default because of the large data imbalance. The classification threshold in each case was set at 0.5, meaning that a segment or data point is classified as signal if the network output is larger than 0.5. Each RNN, except the naive RNN, uses the maximum of the PTS for the classification of the entire light curve segment. The values given are the means and standard deviations over three independent runs of each model.

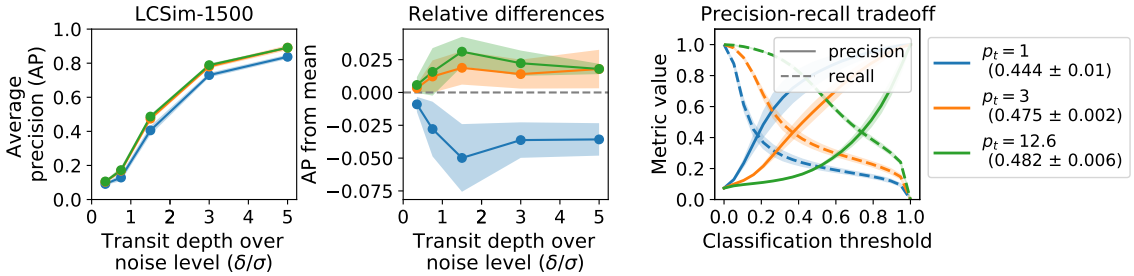


Figure 4.8: The effect of different positive weights on the AP evaluated on the test split of LCSim-1500, and the corresponding shifts in precision and recall versus the classification threshold. To clarify, a threshold of 0.25 would mean that data points are classified as signal if the corresponding RNN outputs are above 0.25. Filled regions indicate the standard deviation over the results of three independent runs. Values between brackets show the overall AP per method.

observed on the validation accuracy. To have a more consistent baseline than CNN, we also include the MLP in the comparison, which depends on fewer hyperparameters. Other models, such as two-layer GRU or LSTM are also included in the comparison as a means to motivate our choice for the bi-GRU-1. Each architectures used ReLU activation functions and a learning rate between 0.005 and 0.008. The MLP had three layers with 128, 64 and 64 nodes respectively and a weight decay of 5×10^{-3}; the CNN had three convolutional layers with 4, 12 and 1 channels and corresponding kernel sizes of 7, 7 and 3 with strides of 1. Batch nornalization and max pooling was applied between layers. The last layer of the CNN is an FC-layer with 48 nodes. Both the CNN and the naive RNN had a weight decay of 1×10^{-4}. For all RNN variants, we used 64 nodes in the recurrent cell, which is followed by two FC-layers of both 64 nodes. The results show that the preferred model (i.e. bi-GRU-1) copes well with longer input sequences, whereas the CNN is affected more strongly by the increase of data points. This is reassuring, as our network should be able to cope with long input sequences which contain more distracting background patterns than the smaller inputs the CNN is generally applied to. The results also show that bidirectionality is important and that simple architectures, e.g. a single recurrent layer instead of multiple, are preferred over more complex ones.

To further explore the potential of the RNN in the task of transit detection at the level of individual data points, we evaluate different weighting schemes to deal with the imbalances in the data for this task. First, we vary the positive weight p_t in Figure 4.8. The results suggest that setting a positive weight $p_t > 1$, is beneficial in terms of the tradeoff that can be made between precision and recall. The weight $p_t = 12.6$ would effectively fully balance the data, because there are about 12.6 times more non-signal data points in LCSim-1500 than there are signal data points. However, as is clear from the figure, a weight this high would also require us to set a high classification threshold to obtain reasonable precision (e.g. ¿0.5). Therefore, we prefer a lower weight, e.g. $p_t = 3$, because it still increases the AP compared to $p_t = 1$, and it allows for setting a more intuitive classification threshold compared to $p_t = 12.6$.

Lastly, we evaluate the use of the transit-specific weighting parameter w_{ij}. In the simple case we have

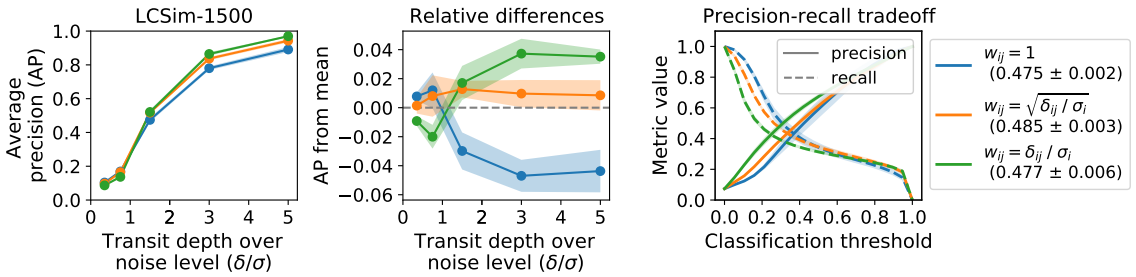


Figure 4.9: The effect of different transit-specific weights on the AP evaluated on the test split of LCSim-1500. No large shifts in precision or recall over the classification threshold are observed.

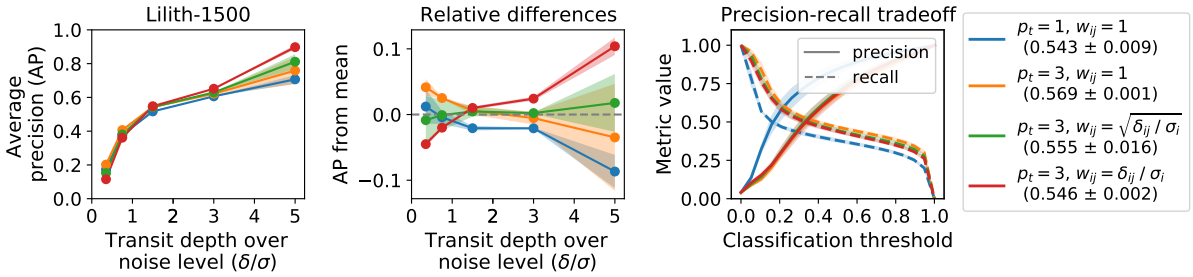


Figure 4.10: The effect of different weighting schemes on the AP evaluated on the test split of Lilith-1500. The results are similar to those obtained using LCSim-1500.

$w_{ij} = 1$, meaning that all signal data points are weighted the same, regardless of their corresponding transit depths. If the data set contains more shallow than deep transit signals, which is true in our case, then using the same weight for all transits might cause the network to become biased toward only detecting shallow signals, even though we should expect the network to perform well also in the simpler case of deep transit signals. Therefore, we evaluate setting $w_{ij} = \delta_{ij}/\sigma_i$, which sets a weight on transit samples that grows linearly with the corresponding transit depth relative to the white noise. Additionally, we set $w_{ij} = \sqrt{\delta_{ij}/\sigma_i}$, which has a similar, but less strong effect. Figure 4.9 shows that w_{ij} can indeed be used to tune the focus of the network towards specific transit samples. In our case, $w_{ij} = \sqrt{\delta_{ij}/\sigma_i}$ provides a good balance in classifying transit signals of varying depths. Similar results are observed for the different weighting schemes applied to Lilith data in Figure 4.10.

4.3 Monotransit retrieval

With the knowledge gained on preprocessing, network architectures and training schemes, we now apply the RNN to real-world transit detection. The RNN trained with weighting parameters $p_t = 3$ and $w_{ij} = \sqrt{\delta_{ij}/\sigma_i}$, is applied to full-length light curves and tested on its ability of retrieving single transit events in LCSim-Mono. A detection is defined by a peak in the PTS for a given input light curve above a given threshold, which visualized in Figure 4.11. In this figure where we also illustrate the use of the confidence outputs produced by the network we refer to as Conf-RNN. It can be seen from the figure that the confidence outputs are as expected, e.g. the network is seems to be less confident at the edges of a transit signal, compared to points around the mid-transit time.

As baseline we use the monotransit detection algorithm used by Foreman-Mackey et al. (2016), which is based on a box function that is fitted to the data at different points in time. The only adjustment we made was that instead of searching for transits of a single duration, we combine the detection results from different trial durations ranging from 1 to 13 hours in steps of 1 hour. The output of the algorithm is a time series of SNR values of the best-fitting box at each time step, which we will refer to as SNR-TS. We set a detection threshold on the SNR above the noise floor in the SNR-TS, which is computed using a sliding median filter. Before applying this method, we detrend the input light curves with a sliding median filter with varying window sizes, i.e. 6, 12, and 24 hours. Since the algorithm is similar to BLS, we refer to it as Mono-BLS-{6h, 12h, 24h}, where the extension (e.g. "6h") refers to the window length of the detrending filter.

Figure 4.12 shows the PR-curves of the different detection algorithms. In this experiment, the RNN outperformed Mono-BLS, which can also be seen in Figure 4.13 where we take a closer look at the performance of both

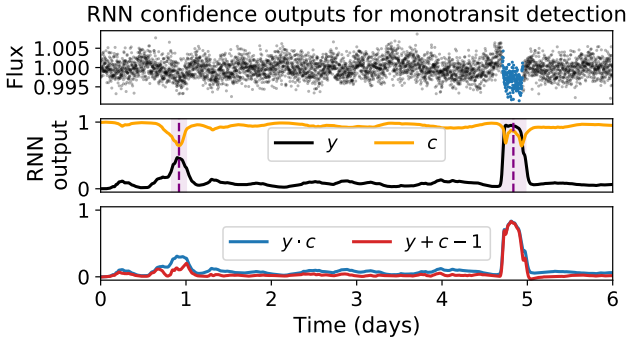


Figure 4.11: An example of how the RNN outputs could be used for monotransit detection. The standard outputs y define the PTS for the given input, which can directly be used the set a threshold on candidate detections (e.g. purple shows candidate detections for $y > 0.25$). In case the confidence RNN is used we also get c, which we use in two ways in combination with y to define an alternative PTS with the aim of reducing peaks for which the RNN is less certain. These curves, as shown in the bottom panel, can similarly be used to set detection thresholds.

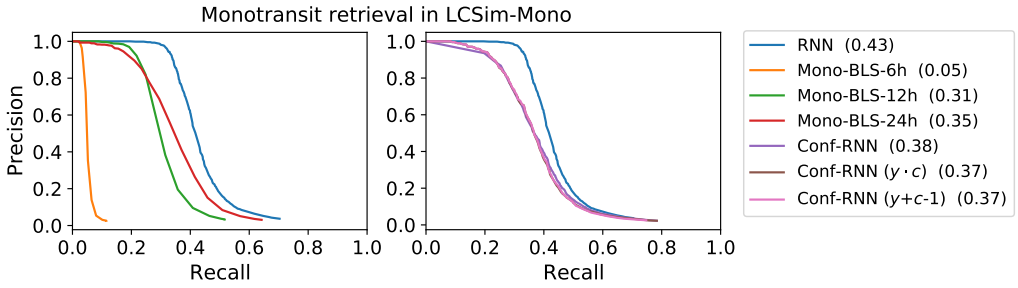


Figure 4.12: Precision-recall (PR) curves of different methods in the task of monotransit detection, divided over two figures to avoid clutter. The area under the curve, or average precision (AP), is given between brackets for each method. The RNN is the single-layer bi-GRU which only outputs the PTS for a given input light curve. Conf-RNN is a network which outputs confidence values in addition to the standard PTS, which performed worse than the standard RNN, even if the confidence values are not used.

methods for specific transit samples. Furthermore, the Conf-RNN performed similar to Mono-BLS and thus worse than the standard RNN, even the Conf-RNN that did not make use of its confidence outputs. In other words, the supposedly more intuitive network came at the cost of worse performance. Table 4.3 presents the exact number of planets retrieved by the best performing methods at a precision of 0.5, including the number of planets found by one method that were not found by the other.

In terms of computational efficiency, the RNN greatly outperformed (our implementation of) Mono-BLS. For the search for monotransits in 5000 light curves, Mono-BLS took about 10 hours on a CPU, while the RNN only took a few minutes on a GPU. If no GPU is available, the RNN would have taken about 2 hours for this task. For Mono-BLS, the computation time could probably be reduced to a few hours by using less trial durations, using a lower time resolution for the search, or using a faster programming language for the implementation. However, since the RNN can generally easily be run on a GPU, if available, with a library such as PyTorch, and pre-implemented algorithms such as (Mono-)BLS do not naturally come with GPU compatibility, the RNN remains the preferred choice in terms of efficiency.

4.4 Single planet retrieval

For the evaluation of the ability of the RNN-based algorithms to retrieve single planets with multiple transit signals, we apply the RNN from previous sections to light curves in the LCSim-Single data set. Subsequently, we use PTS-Fold and PTS-Peak to determine the period and epoch of a maximum of three candidate signals per light curve. Both algorithm provide for each candidate a corresponding detection score, as described in Section 3.2, which is used to set a detection threshold. A detection is counted as correct if the estimated period is correct with a 1% error margin, and the estimated epoch is within the duration of the first transit in the light curve.

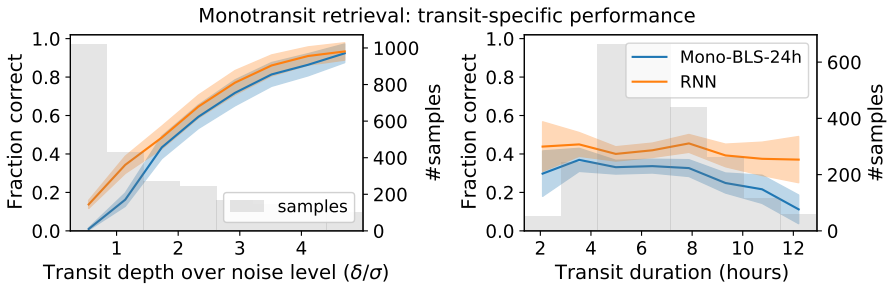


Figure 4.13: A close in on the ability of the presented methods to retrieve specific transit samples, e.g. if the fraction correct is 0.5, then half of the planets were detected in the corresponding parameter bin. Underlying data distributions are shown in gray which also indicate the bins used in the evaluation. The detection threshold for both methods was set closest to a corresponding detection precision of 0.5. Filled regions shows the Wilson score interval (Wilson, 1927) which approximates the 95% confidence interval of the performance within each bin.

	RNN		Mono-BLS-24h	
	1042	(1.00, 0.42)	818	(1.00, 0.33)
not RNN	-		74	(0.09, 0.03)
not Mono-BLS-24h	298	(0.29, 0.12)	-	

Table 4.2: The absolute and relative number of correct detections in the task of retrieving monotransits, where the detection thresholds are set closest to a corresponding detection precision of 0.5. An element, say with labels "A" in the corresponding column and "not B" in the row, corresponds to the number of detected planets by A that are not detected by B. Values between brackets respectively show the fraction with respect to the total number of detections of A, and the fraction with respect to the total amount of planets in the data, i.e. 2500.

As baseline we use the standard BLS algorithm, as implemented in the `astropy` package[2]. We set a detection threshold on the SDE of each candidate in the BLS periodogram, and recursively apply the method to search for a maximum of three potential signals per light curve, each time masking the previous signal that was found. The algorithm is applied to detrended light curves using sliding median filters of varying lengths, i.e. 6, 12, and 24 hours.

Figure 4.14 shows that a 12-hour window works best for detrending in combination with BLS for this task, which is in line with the results from Hippke et al. (2019). Moreover, the results show that the BLS algorithm outperformed the RNN-based detection algorithms in this experiment. Among the RNN-based algorithms tested, PTS-Fold performed best, which was as expected because it does not rely on distinct peaks in the PTS as opposed to PTS-Peak. The performances of the best algorithms are further inspected in Figure 4.15. This figure shows, in line with our monotransit experiment, that towards larger periods, i.e. fewer transit signals, the performance of BLS drops faster than that of the RNN-based algorithm. This suggests that if there is periodicity in the signal, BLS remains dominant over the RNN. However, if the periodicity is lacking, the RNN is the preferred method. Table 4.4 presents the number of planets retrieved by the methods, including the number of planets that were detected by one algorithm and not by the other.

The RNN-based algorithm required a longer computation time in this case than in the monotransit case. To obtain the PTS of each of the 5000 light curves still required only a few minutes on a GPU, but the determination of the periodicity of potential signals added more to the necessary computation time. PTS-Peak required only 5 minutes on a CPU. PTS-Fold on the other hand, took about 1.5 hours. Still, PTS-Fold was faster than BLS which took over 3 hours of computation time. The computation times are in line with the relative performances of each method in this task. Nevertheless, it is noteworthy that PTS-Peak retrieved 20 planets that BLS was not able to find at a precision of 0.5, while using less than 35 times its computation time.

4.5 Multiplanet retrieval

Similar to the single planet case, we compare the RNN-based detection algorithms with BLS in the task of retrieving planets with multiple transit signals. In this case we use Lilith data, which in some cases contain transit signals from different planets within the same light curve. Again we restrict both methods to only output

[2] https://docs.astropy.org/en/stable/timeseries/bls.html

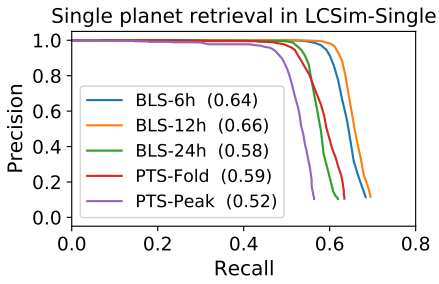


Figure 4.14: Precision-recall curves for different methods in the task of detecting repeating transit signals of a single planet. The average precision is given between brackets.

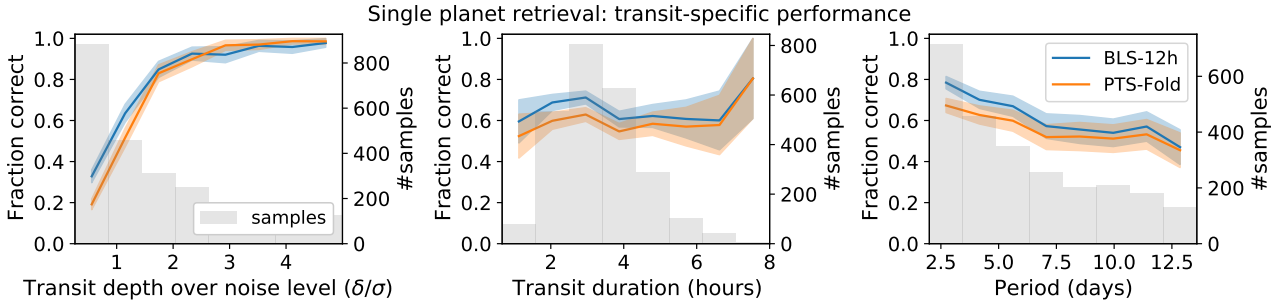


Figure 4.15: The ability of the presented methods to retrieve planets within the given parameter ranges. A detection threshold was set closest to a corresponding detection precision of 0.5. PTS-Peak is not shown because it behaves similarly but worse than PTS-Fold and would therefore only clutter the image. Filled regions show the Wilson score interval (Wilson, 1927), which approximates the 95% confidence interval of the performance per bin.

three candidate detections per light curve, to save on computation time.

In this experiment we also include the TESS pipeline detections for the given data set. However, since a threshold has already been set on the pipeline detections, we only have limited data, and the PR-curve for the pipeline therefore cannot be drawn fully. For the pipeline's PR-curve shown in Figure 4.16, we set a threshold on the SNR of the detections that are provided in the DV files [TODO: describe DV files and pipeline detections in Lilith-4 simuations section]. It can be seen from this figure, that the pipeline's PR-curve steadily decreases, suggesting that a high AP could be obtained if we had access to all the detections below the pre-set threshold. In contrast, the RNN-based algorithms drop down steeply. Nevertheless, the RNN-based methods seem to be able to maintain high precision for the detection of about 10% of all the planets. BLS on the other hand, starts off with relatively low precision, but is able to retrieve far more of the total amount planets. In terms of average precision, it therefore outperformed the other methods. Figure 4.17 shows the detection results over the distribution of transit samples for different parameters. We can see that each method behaves similarly, but at different levels of performance. Table 4.5 presents the number of planets retrieved by each method, including the number of planets that were detected by one method and not by the other. Table 4.5 is provided to give a closer view on the ability of PTS-Fold and BLS-12h to retrieve multiple planets in a single light curve. From this table we can see that PTS-Fold is able to detect multiple planets in the same light curve, but there is room for improvement as compared to BLS-12h.

4.6 Success and failure cases

The experiments as presented in previous sections provide insight into the general performance of our RNN-based algorithms. In this section we look at individual cases, in order to identify when exactly the algorithm fails, and when it succeeds. Only a few cases are presented here, which are chosen to represent common mistakes. We note that since this is only a selection, it does not cover all success and failure cases. Furthermore, since we are dealing with individual cases we only describe our observations based on these examples, without making hard conclusions of the performance overall.

In Figure 4.18, we see the RNN and Mono-BLS applied to a light curve from the LCSim-Mono data set. In this case, the RNN was able to detect the planet, while for Mono-BLS the SNR did not exceed the detection

	PTS-Fold		PTS-Peak		BLS-12h	
	1480	(1.00, 0.59)	1333	(1.00, 0.53)	1655	(1.00, 0.66)
not PTS-Fold	-		7	(0.01, 0.00)	224	(0.14, 0.09)
not PTS-Peak	154	(0.10, 0.06)	-		342	(0.21, 0.14)
not BLS-12h	49	(0.03, 0.02)	20	(0.02, 0.01)	-	

Table 4.3: The absolute and relative number of correct detections in the task of retrieving planets with multiple transit signals, where the detection thresholds are set closest to a corresponding detection precision of 0.5. The data contained a total of 2500 planets.

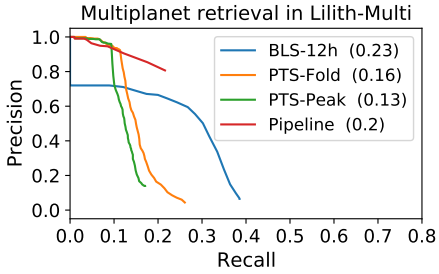


Figure 4.16: Precision-recall (PR) curves of different methods in the task of detecting repeating transit signals in light curves of Lilith-Multi [TODO: change lilith-4 to lilith-multi]. The PR-curve for the pipeline could not be drawn fully due to limited available data.

threshold. This is likely due to the fact that the box-fitting algorithm picked up more noise than the RNN. For Mono-BLS, it seems that each fluctuation in brightness triggers some response to a transit event, while the RNN is relatively robust against this noise. We expect this to be the reason that a better threshold could be set with the RNN, in terms of the tradeoff between precision and recall, compared to the Mono-BLS algorithm.

When periodicity is involved, we found BLS to outperform our RNN-based algorithms in Section 4.4. Especially in the small period regime (e.g. see Figure 4.19), we found the RNN-based algorithms struggle with the fact that it relies on the detection of individual events, only after which the period of the signal is determined. BLS on the other hand, uses the periodicity to its advantage by folding the input light curve, effectively increasing the SNR of the signal. Still, in cases where the input light curve is very noisy, as shown in Figure 4.20, we found the RNN to be able to distinguish signals from the background, allowing PTS-Fold to detect the planet while BLS failed to do so. PTS-Fold and PTS-Peak are compared in Figure 4.21, where we see how the lack of strong peaks in the PTS can lead to PTS-Peak failing, while PTS-Fold still succeeds.

For Lilith light curves, shown in figures 4.22 and 4.23, several things were observed. Firstly, the PTS seems to be less noisy compared to the PTS obtained with LCSim data. For this reason, the PTS-Fold periodogram is also cleaner. Although this seems like a good thing, it also means that a transit signal is either detected, or not at all. This is likely the reason for the high precision at low recall of the RNN-based detection algorithms applied to Lilith data. We expect that a bit of noise in the PTS is beneficial to the algorithm, as it allows more planets to be detected at lower precision.

Secondly, we found that our algorithms are prone to match the signals from two different planets together,

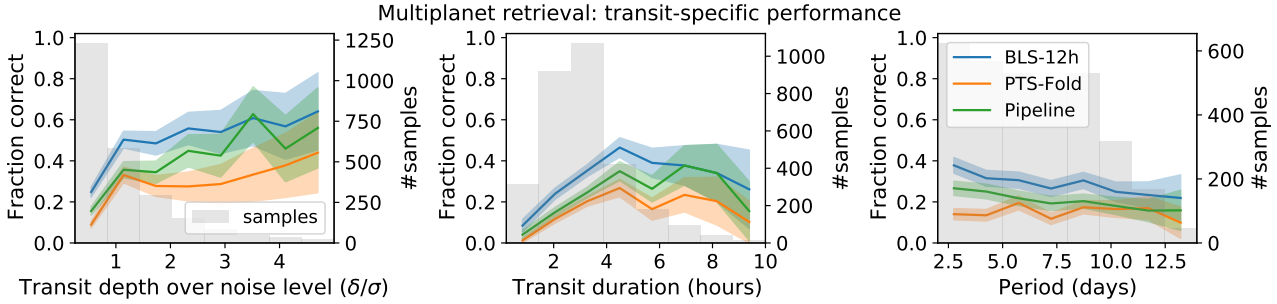


Figure 4.17: The ability of each method to retrieve planets in the given parameter ranges. A detection threshold is set closest to a corresponding detection precision of 0.5. Filled regions show the Wilson score interval Wilson (1927), which approximates the 95% confidence interval of the performance per bin.

	PTS-Fold		BLS-12h		Pipeline	
	493	(1.00, 0.15)	993	(1.00, 0.30)	710	(1.00, 0.22)
not PTS-Fold	-		522	(0.53, 0.16)	304	(0.43, 0.09)
not BLS-12h	22	(0.04, 0.01)	-		46	(0.06, 0.01)
not Pipeline	87	(0.18, 0.03)	329	(0.33, 0.10)	-	

Table 4.4: The absolute and relative number of correct detections in the task of retrieving planets with multiple transit signals, where the detection thresholds are set closest to a corresponding detection precision of 0.5. For the pipeline, the level of precision is in fact higher and the numbers presented here are relatively small, because we had no access to its lower confidence detections. The data contained a total of 3285 planets.

# planets (# light curves)	Method	Detected 1	2	3	4
1 (2130)	PTS-Fold	371			
	BLS-12h	666			
2 (474)	PTS-Fold	80	9		
	BLS-12h	120	65		
3 (57)	PTS-Fold	13	5	0	
	BLS-12h	8	20	6	
4 (9)	PTS-Fold	1	0	0	0
	BLS-12h	4	2	1	0

Table 4.5: The number of correctly detected planets per light curve with a given number of planets. For example, of the 57 light curves that contained 2 planets, PTS-Fold was able to retrieve 2 planets from 5 light curves, and only 1 planet from 13 light curves. The detection threshold was set closest to a corresponding detection precision of 0.5.

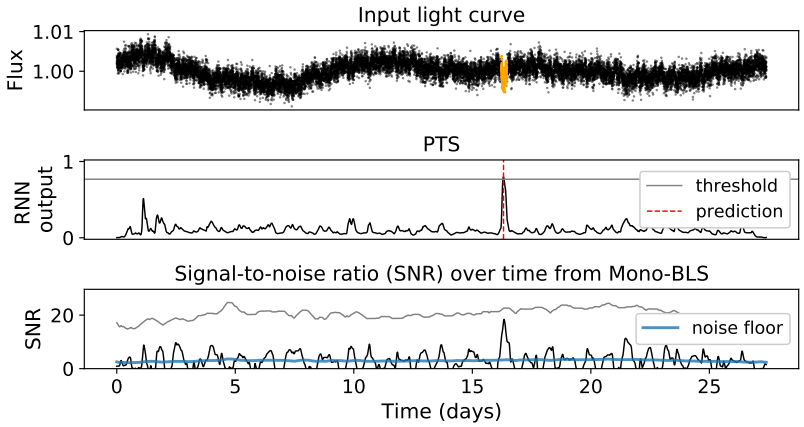


Figure 4.18: An example of where the RNN succeeds and Mono-BLS fails in the task of monotransit retrieval. A detection threshold is set closest to a corresponding detection precision of 0.5 over the entire LCSim-Mono data set. The transit signal in the top figure is visualized in orange and a detection above the threshold is indicated with a dashed red line.

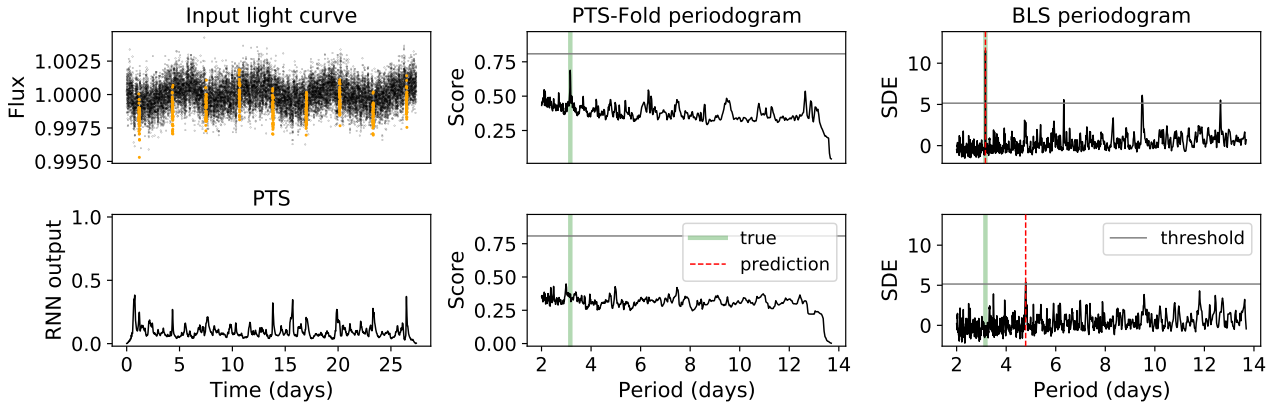


Figure 4.19: An example of where BLS succeeds and PTS-Fold fails in the task of retrieving repeating transit signals from a single planet. A detection threshold is set closest to a corresponding detection precision of 0.5 over the entire LCSim-Single data set. Transit signals are visualized in orange and a detection above the threshold is indicated by a dashed red line. Two steps of each method are visualized from top to bottom, where the top shows the initial periodogram, and the bottom shows the periodogram after the strongest signal has been removed.

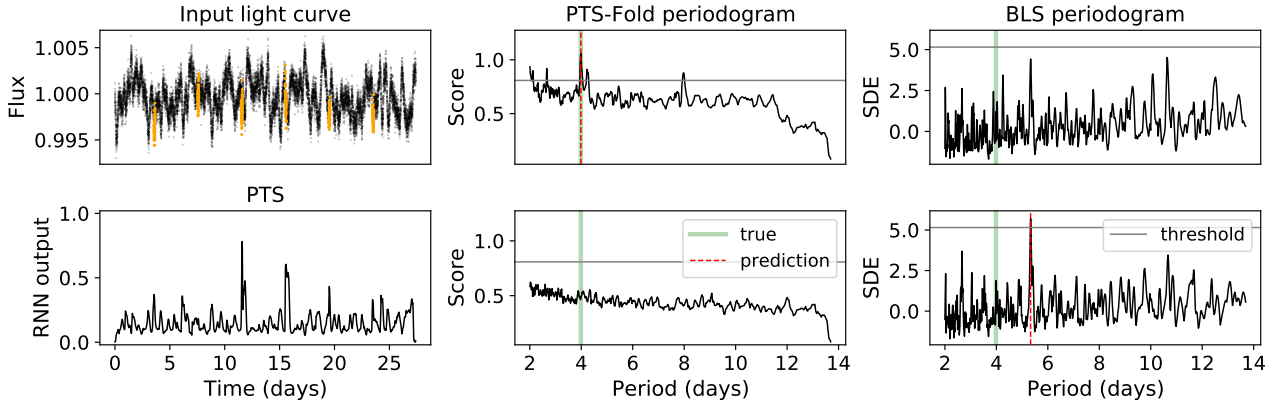


Figure 4.20: An example of where PTS-Fold succeeds and BLS fails in the task of retrieving repeating transit signals from a single planet. A detection threshold is set closest to a corresponding detection precision of 0.5 over the entire LCSim-Single data set.

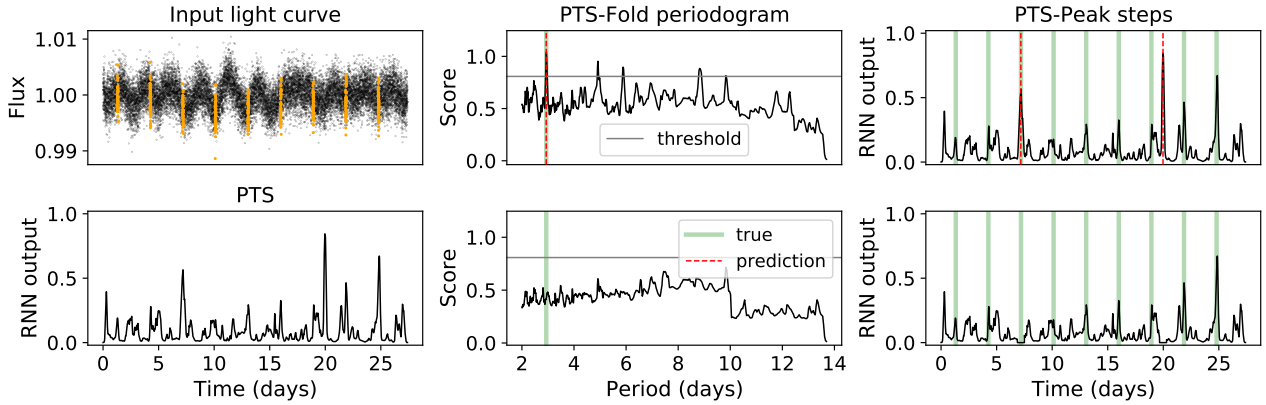


Figure 4.21: An example of where PTS-Fold succeeds and PTS-Peak fails in the task of retrieving repeating transit signals from a single planet. A detection threshold is set closest to a corresponding detection precision of 0.5 over the entire LCSim-Single data set. Two steps of both methods are shown from top to bottom. For PTS-Peak the top figure shows the original PTS, with green colors indicating the location of true transit signals, and red dashed lines indicating the predicted transit signals. The lower figure shows the PTS where the detected signal is removed.

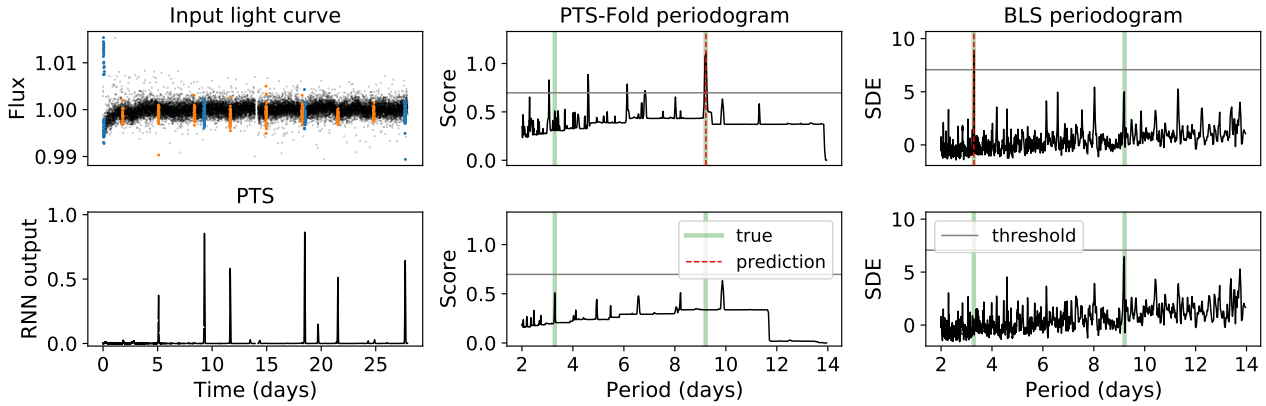


Figure 4.22: An example of where BLS and PTS-Fold both detect a planet that the other method was not able to detect. The input is a Lilith light curve with transit signals from two planets, indicated by blue and orange in the top left figure. A detection threshold is set closest to a corresponding detection precision of 0.5 over the entire Lilith-Multi data set.

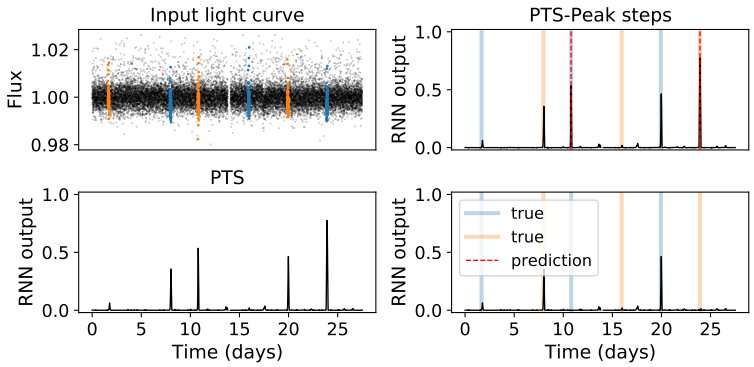


Figure 4.23: An example of where PTS-Peak fails to match the peaks corresponding to the same signal for a given Lilith light curve. The colors indicate which transit signal belongs to which planet. The dashed line indicates the peaks that are extracted to define a detection, which in this case include peaks from two different planets.

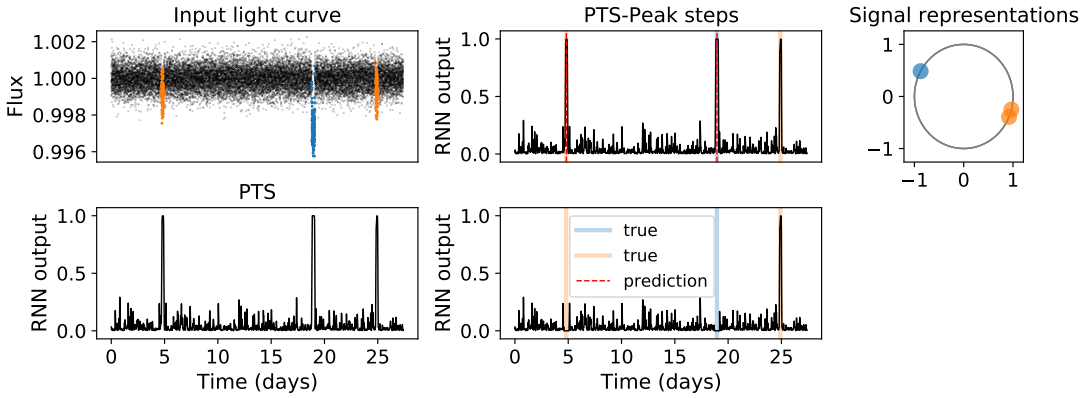


Figure 4.24: An example of how PTS-Peak may falsely match signals together from different planets, leading to incorrect detections. This problem could be prevented by using learned representations of transit signals, shown in the upper right. The dots represent the aggregated representations over all time steps withing a given peak in the PTS. The colors correspond to the colors used in the upper left figure. The light curve in this example consists exclusivey of white-noise.

which could lead to false detections. This phenomenon is made clear in Figure 4.23, where the wrong peaks in the PTS are combined by PTS-Peak to define a detection. This behaviour can be understood by the fact that peaks in the PTS carry little information about the shape of the detected signal in the light curve. Only the duration of a signal can be directly encoded in the PTS. For other features, we resort to the representations learned by the representation RNN from Section **??**. Figure 4.24 shows that, even though the signals in the light curve are visually different, the peaks in the PTS are similar enough for PTS-Peak to match the wrong signals together. This could, however, be prevented if the signal representations are taken into account, which are separated in representation space. The representations are in this case two-dimensional, and the angle between two representations indicates how different they are. Figure 4.25 shows another example of this failure case, where the two signals are now more similar. The representations are therefore also closer to each other, but still can be separated in representation space.

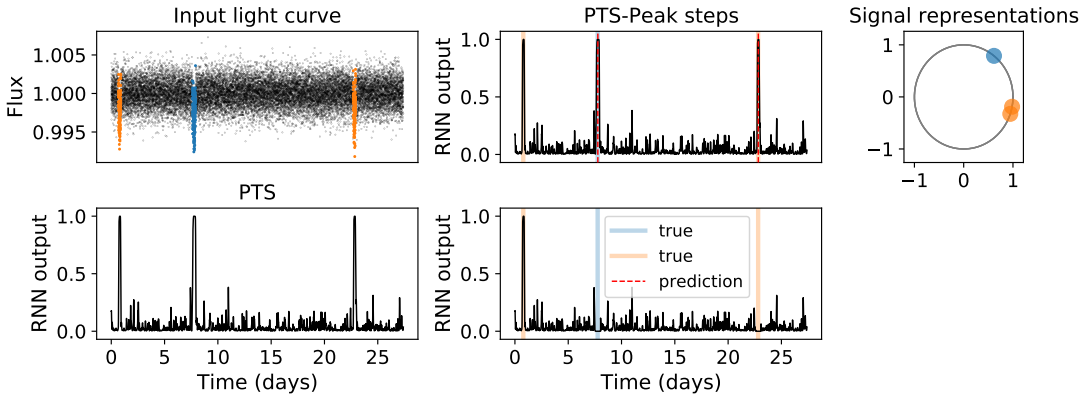


Figure 4.25: Another example of how the use of learned signal representations could have prevented PTS-Peak from matching the wrong peaks together in the PTS.

Chapter 5

Discussion

The main questions in thesis were whether and how RNNs could be used for the detection of transit signals, and whether they provide a competitive and efficient alternative to existing approaches. In addition, we attempted to investigate how the outputs of the RNN could be made more interpretable for this task, such that it would benefit the transit detection algorithm. In the following, we discuss these questions with regard to the results presented in Chapter 4. The subsequent sections describe the limitations of this work and directions for future work.

5.1 Answers to research questions

Although this work is based on simulated data, the results achieved on both LCSim and Lilith data show that RNNs have promising capability of performing well in the task of transit detection. The question of how RNNs should be applied for this task was investigated in several experiments. Different preprocessing steps were evaluated, involving for example the way data gaps are handled and the way input data is scaled. Although RNNs come with the possibility of predicting missing values in a light curve, we found that a simple gap-filling approach such as linear interpolation can lead to similar results, while requiring less computation time. Furthermore, it was shown that zero-filling gaps in Lilith data can result in better performance of the RNN, compared to using linear interpolation. Possibly, zero-filling allows the network to better distinguish imputed data points from real ones, such that it takes into account the fact that they were missing from the original light curve. In literature concerning transit detection, the effect of missing data on the performance of neural networks was only touched upon by Pearson et al. (2018), however no methods were evaluated to deal with this problem. In the task of light curve classification, Naul et al. (2018) and Becker et al. (2020) provide the differences between sampling times as input to an RNN to deal with data gaps. This approach has yet to be evaluated in the task of transit detection.

Regarding the scaling of light curves prior to applying the RNN, we expected that scaling light curves by their estimated white noise level would be beneficial to the network. Namely, we would argue that by removing the variance in white noise, the transit signal would stand out more. However, the results showed that this sort of scaling, which we referred to as sigma scaling, was only beneficial for the deepest transit signals in Lilith data. For LCSim data and shallower transits in Lilith data, either no scaling led to better results than sigma scaling, or no clear distinguishment could be made.

In line with Pearson et al. (2018), it was shown that detrending is not necessary for the algorithm to work, as opposed to for example the BLS algorithm Kovács et al. (2002). Nevertheless, a positive effect was observed when low-risk detrending was applied to remove only large-scale stellar variability, with the goal of making the input ranges to the network more consistent. Centroid data also seemed to help the network in distinguishing background patterns from data points belonging to transit signals. [TODO: cite centroid inputs]

In addition to preprocessing, the network architecture and training also take part in our question of how the RNN should be applied for our task. It was shown how using different weighting parameters in the network's loss function can help finetune the network on the unbalanced training data which we are dealing with. For example, it was found that by adding a small positive weight to all data points belonging to transit signals, one can optimize the trade-off between precision and recall at any given classification threshold. We expect that this weight helps to increase the noise floor in the PTS for a given light curve, which we found to be higher for LCSim data than for Lilith data in several cases. Since the noise in the PTS may still contain relevant information about the presence of potential signals, a larger positive weight for Lilith data may have resulted in higher recall for the RNN-based detection algorithms. A transit-specific weight was also adopted, with the desired effect of balancing the network's focus of learning to detect transit signals of varying depths. Nevertheless, slight changes to these parameters may lead to better results in different data sets, and we therefore re comment to tune them

to the data at hand.

Regarding the network architecture, we opted for a relatively simple design. In the first place, this was to illustrate that this task does not require a highly complex network architecture. In addition, a simple architecture would allow for easier reproduction of the work. During the process of comparing architectures, in fact we found that simpler architectures led to better results than more complex ones. For example, GRU produced better and more stable results than LSTM, while having less parameters. Furthermore, a network with multiple recurrent layers performed slightly worse than the simpler network with only one layer. Including bidirectionality in the RNN on the other hand, led to a great improvement in the results. The confidence RNN was only slightly more complex than the basis network, but also led to worse results.

The question of whether RNNs provide an efficient and competitive alternative to existing approaches was investigated in three real-world problems. In terms of efficiency, neural networks in general were expected to be appealing compared to the common brute-force approach that is taken to search for transits. The computation times observed in the experiments for Mono-BLS and BLS compared to our RNN-based approach were in line with this expectation. Among different types of NNs, RNN is known to be less efficient than the CNN, due to its recurrent structure. However, since CNNs are required to be applied multiple times to the same light curve in the task of transit detection Pearson et al. (2018), this difference becomes smaller. A comparison of efficiency between the CNN and RNN has not been made in this work, because many different design choices of this way of using the CNN could lead to different results. For typical TESS light curves of 27.4 days, our RNN was able to compute the PTS in under one second for over 50 light curves at once. This is reassuring, because the data sets that are currently produced by TESS and in the future by PLATO require efficient algorithms. The most time was taken by the method which determines the periodicity of candidate signals using the PTS. In this task, PTS-Fold led to more correct detections than PTS-Peak, but PTS-Peak outperformed PTS-Fold in terms of efficiency by at least one order of magnitude. This is because PTS-Fold tends more towards a blind search, while PTS-Peak is guided by the detection of individual events in the PTS. If the search is directed towards monotransits, however, we do not require PTS-Fold or PTS-Peak, and the PTS can directly be used to set detection thresholds. For this reason, the search for monotransits in large data sets can be conducted in minutes using an RNN.

Not only in terms of efficiency, but also in terms of detection performance, the RNN showed most potential in the case of monotransits. We found it to be more robust against transit-resembling noise patterns than a box-fitting approach. This is likely because it was specifically trained on individual transit signals, and not on aggregated or folded light curves. It therefore does not rely on periodicities of signals the way BLS does. BLS uses periodicities to its advantage by aggregating multiple transits in the same light curve, effectively increasing the SNR of the signal. We expect this to be the reason that BLS was still dominant in the case of repeating signals. Nevertheless, we found multiple examples of where BLS failed to detect a planet, while the RNN was able to detect it. Some of these cases showed to be highly noisy light curves with noise patterns at time scales similar to the transit signal. In these cases, detrending may be the limiting factor for BLS to detect the planets. Since the RNN does not require detrending, it may still distinguish the signals from the distracting background.

Lastly, two approaches were evaluated to increase the interpretability of the RNN. Although true uncertainties are difficult to obtain, we attempted to let the RNN output an indication of its confidence over given outputs. Visually, the results were as we would expect: in noisy regions of a light curve the confidence was often slightly reduced, even though the standard outputs remained the same; in the case of a transit signal, the confidence would be low at ingress and egress, indicating that the network is not certain when exactly the transit starts and ends. However, in some cases the confidence outputs c were approximately the same as y mirrored, i.e. $y = 1 - c$. We expect this to be the reason why we did not observe differences in the performance if the confidence outputs were or were not used in the task of monotransit detection.

Another attempt of increasing the interpretability of the RNN outputs, involved the use of learned representations of signals. We only provided an illustration of how these representations may be used to resolve ambiguities, but the few examples that were evaluated showed intuitive results. The standard outputs of the RNN provide only limited information. Any increase in the PTS may indicate the presence of a transit signal, but sometimes it is not clear what triggers the response of the RNN. Moreover, the connection between two peaks in the PTS can only be based on their duration. Learned representations could open the door to understanding better what caused the RNN to output a certain value, by comparing its representations at a different moments in time. In addition, as was shown here, they can be used to separate the detection of signals from different planets. However, more research needs to be conducted to evaluate the effectiveness of this approach.

5.2 Limitations and suggestions

The main limitation of this work is that we used simulated data. Simulated data was used to aid in the development and evaluation of our method, because simulations allow for full control over parameters and come with a ground-truth. However, we cannot say based on our experiments whether the results will hold when

real-world data is used. We expect that real TESS data is similar to Lilith data, and therefore do not expect to observe large differences when the RNN would be applied to real-world light curves. We did observe differences between the results obtained using LCSim data and Lilith data. One of the differences was the overall lower recall of both BLS and the RNN, when applied to Lilith light curves instead of LCSim light curves. We believe that the main reasons for this were the fact that Lilith data contained relatively more shallow transit signals than LCSim data, and for the RNN that the absence of noise in the PTS for Lilith light curves prevented planets to be detected at low detection thresholds.

Related to the fact that we used simulated data, is our choice to leave out signals from eclipsing binary (EB) systems in this work. When applying the detection method to real-world data, this choice cannot be made, because we do not know beforehand whether an EB signal is present in a given light curve. Our current approach however, can still be applied to light curves containing EB signals. In case the EB signal resembles a transit signal, the RNN might treat it as transit signal and outputs classifies it as such. Other algorithms would likely also make mistakes if the EB signal resembles a transit signal. In case the EB signal is stronger than the typical transit signal, the RNN might treat it as noise, because it has never seen such signals during training. However, for the same reason, its outputs might also become unreliable in these cases. Alternatively, one could include known EB signals during training, and treat them as separate class of signal. The network could then be trained for the three-class classification problem of separating data points belonging to transit signals, EB signals or noise, with corresponding transit-PTS, EB-PTS and noise-PTS.

In this research, we separated two components of our detection algorithm: the RNN and the method used to extract candidate detections from the RNN's predictions. When referring to the detection performance of the RNN, in most cases we referred to the trade-off that can be made between precision and recall in the task of classifying individual data points as signal or non-signal. However, this measure of performance is not the same as the performance of our method to detect transit events, because transit events cover multiple data points. We tuned our RNN based on the performance measure over individual data points, and the subsequent algorithm based on the actual task of providing the correct P and t_0 of transit signals. The overall algorithm might be improved if it is tuned from front to end, based on the final outputs for P and t_0. Such an approach would, however, result in longer computation times since in the worst case, the number of parameter settings to be evaluated is increased exponentially.

Another limitation of this work is our choice of baseline. We opted for box-fitting algorithms, as they are intuitive and relatively efficient compared to other least squares methods. However, the more realistic transit model used in Transit Least Squares (TLS, Hippke and Heller (2019)) was found by the authors to match the transit signals in Kepler data in over 99% of the cases better than a box model. TLS requires priors on the stellar density to perform comparably with BLS in terms of efficiency. In the case for LCSim, we did not implement a physical relation between stellar density and limb darkening parameters of the simulated star. Without priors, TLS would have taken up to three times longer than BLS to run in our experiments, which is what motivated us to use BLS over TLS.

The use of least squares methods compared to the use of a neural network brings us to the problem of interpretability. Although they may be computationally expensive, least squares fitting algorithms with physics motivated transit models are arguably more interpretable than the hidden dynamics of neural networks. First of all, in our attempts of increasing the interpretability of our network, we only looked at the outputs of the network. In other words, the black box remains a black box, simply with more outputs, e.g. confidence values, or representation vectors. Similarly, one could let the RNN output the predicted depth of a signal, which, in combination with estimated duration allows for the approximate SNR to be computed of candidate detections. However, what causes a certain value or vector to be outputted by the model remains hidden from the user. Second of all, which values should be used for setting a detection threshold in the RNN-based algorithm remains a question. In this work, we aimed to come up with a simple detection criterion using the values in the PTS. In our approach we summed the values of the PTS corresponding to the same phase for a given period, while scaling down by the square root of the number of values in the sum. This was a rather arbitrary choice, but showed to be sufficient for the purpose of this work. Third of all, the output of our algorithm in its current form does not provide uncertainties over the predicted values for P and t_0. For the identification step, it would be useful to have access to these, as deviations from the expected time of signals could more easily be explained by imprecise detection results. We expect the PTS alone to be sufficient to allow for some form of uncertainty estimation in the provided parameters, but further research should establish how this can be done best.

5.3 Future directions

A clear direction for the future of the RNN-based detection method is to be applied to real-world data. Especially in the task of monotransit detection, we expect the RNN to lead to interesting new results. In our work, a general search for transits was evaluated, i.e. transits of varying depths and varying shapes. However, the training of the network can also be tuned for the search for specific kind of transit signals. For example, the training data

can be injected with transits from exoplanets with orbiting "exomoons". In this approach, the network can be trained to only detect these special transits, while ignoring the "normal" transit signals. Similarly, one could inject the training data with transits only from disintegrating rocky exoplanets (DREs), or transits signals from other particularly interesting objects.

To address several of the limitations described in the previous section, future research could also aim at improving the detection method. For example, it would be highly favorable if the network is able to work with periodicities of potential signals, such that network directly outputs P and t_0 and can be trained end-to-end. A considerable portion of this project was devoted to finding a solution for this, without success. Other directions include the estimation of more parameters in the output of the network, e.g. the depth or duration of the signal, and estimations of their uncertainties. Further research could also evaluate the effect of EB signals on the performance of the RNN. For example, either by considering EB signals as noise, or by including them in the problem as an additional class of signal.

Lastly, we found the problem of transit detection to not always be defined clearly or consistently in literature. Therefore we believe it would be worth constructing a benchmark data set with a clearly defined task, specifically for the task of transit signal detection. Not only would this make the comparison between different papers and methods easier, it may also motivate more AI researchers to tackle this problem from different perspectives.

Chapter 6

Conclusions

In a series of experiments using simulated data, we evaluated how recurrent neural networks (RNNs) could be applied for the task of transit detection in light curves, and how they compare to existing approaches.

To answer the question of how RNNs could be applied, we evaluated different data preprocessing steps, network architectures and training schemes. Our results suggested that simple gap-filling approaches such as linear interpolation or zero-filling can effectively and efficiently be used to deal with missing data. However, more research is needed in order to establish what method to deal with missing data works best in this task. Detrending or scaling light curves, apart from median scaling, was shown not to be necessary. However, detrending did lead to better results, especially if short-scale patterns were kept in tact. We therefore recommend to use the RNN in combination with a high-pass filter applied to the light curve, which only removes the largest trends. Throughout this research, we found the most stable and best performing network to be a bidirectional GRU RNN with a single recurrent layer followed by two fully connected layers. This network improved over the more complex LSTM, and multi-layered RNN architectures. The network was trained to classify individual data points in light curve segments as signal or non-signal. During training, we found that a small positive weight (>1) applied to all data points belonging to transit signals led to a higher average precision (AP) and allowed for setting more intuitive classification thresholds. Lastly, it was shown that a transit-specific weight can be adopted to balance the focus of the network during training, which also led to an improved AP in some cases.

Two RNN-based algorithms were compared to box-fitting algorithms in the task of detecting transit signals in full-length light curves. These algorithms were referred to as PTS-Fold and PTS-Peak. Although PTS-Peak was at least an order of magnitude more efficient than PTS-Fold, PTS-Fold consistently detected more planets than PTS-Peak. However, the BLS algorithm from Kovács et al. (2002) still outperformed both algorithms in terms of the number of the detected planets. In the case of monotransits, i.e. transit signals which only occur once in a light curve, we found the RNN to outperform the box-fitting algorithm from Foreman-Mackey et al. (2016). In that case, the RNN outputs can directly be used for detection, and light curves can be processed very efficiently (>50 light curves with $\sim$ 20000 data points per second). Overall, we found that the fewer the transits, the more potential the RNN has of improving over box-fitting algorithms.

However, the main result of our research is not that there was a slight winner in each of the experiments. In fact, we found both the box-fitting algorithms and the RNN-based algorithms to detect exoplanets that the other methods were not able to detect. From the 2500 monotransits, the RNN detected 298 planets which the box-fitting algorithm was not able to detect at 0.5 precision, which in turn detected 74 planets which the RNN was not able to detect. In the task of retrieving repeating transit signals from 2500 planets, the RNN-based algorithm detected 49 planets which BLS was not able to detect, which in turn detected 224 planets which the RNN-based algorithm was not able to detect. In other words, the methods performed complementary. For exoplanet science, this is an important result, as the RNN might open the door to detecting exoplanets in real-world data previously missed by commonly used detection methods. Future will tell, as this method has yet to be applied to real-world light curves. Even though these may only include a handful of undiscovered exoplanets, with every new detection our view of the universe, and our place therein, becomes more complete.

References

Megan Ansdell, Yani Ioannou, Hugh P Osborn, Michele Sasdelli, Jeffrey C Smith, Douglas Caldwell, Jon M Jenkins, Chedy Räissi, Daniel Angerhausen, et al. Scientific domain knowledge improves exoplanet transit classification with deep learning. *The Astrophysical journal letters*, 869(1):L7, 2018.

SCC Barros, O Demangeon, RF Díaz, J Cabrera, NC Santos, JP Faria, and F Pereira. Improving transit characterisation with gaussian process modelling of stellar variability. *Astronomy & Astrophysics*, 634:A75, 2020.

Ignacio Becker, Karim Pichara, Márcio Catelan, Pavlos Protopapas, Carlos Aguirre, and Fatemeh Nikzat. Scalable end-to-end recurrent neural network for variable star classification. *Monthly Notices of the Royal Astronomical Society*, 493(2):2981–2995, 2020.

Loic Bontemps, James McDermott, Nhien-An Le-Khac, et al. Collective anomaly detection based on long short-term memory recurrent neural networks. In *International Conference on Future Data and Security Engineering*, pages 141–152. Springer, 2016.

William J Borucki and Audrey L Summers. The photometric method of detecting other planetary systems. *Icarus*, 58(1):121–134, 1984.

S Carpano, Suzanne Aigrain, and F Favata. Detecting planetary transits in the presence of stellar variability-optimal filtering and the use of colour information. *Astronomy & Astrophysics*, 401(2):743–753, 2003.

Joshua A Carter and Eric Agol. The quasiperiodic automated transit search algorithm. *The Astrophysical Journal*, 765(2):132, 2013.

Joseph H Catanzarite. Autovetter planet candidate catalog for q1-q17 data release 24. *Astronomy & Astrophysics*, 2015.

Joheen Chakraborty, Adam Wheeler, and David Kipping. Hundreds of new periodic signals detected in the first year of tess with the weirddetector. *Monthly Notices of the Royal Astronomical Society*, 499(3):4011–4023, 2020.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366*, 2018.

Yann Cherdo, Paul de Kerret, and Renaud Pawlak. Training lstm for unsupervised anomaly detection without a priori knowledge. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4297–4301. IEEE, 2020.

Pattana Chintarungruangchai and Guey Jiang. Detecting Exoplanet Transits through Machine-learning Techniques with Convolutional Neural Networks. *Publications of the Astronomical Society of the Pacific*, 131 (1000):064502, 2019.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Jeffrey L Coughlin. Planet detection metrics: Robovetter completeness and effectiveness for data release 25. *Kepler Science Document KSCI-19114-002*, page 22, 2017.

Anne Dattilo, Andrew Vanderburg, Christopher J Shallue, Andrew W Mayo, Perry Berlind, Allyson Bieryla, Michael L Calkins, Gilbert A Esquerdo, Mark E Everett, Steve B Howell, et al. Identifying exoplanets with deep learning. ii. two new super-earths uncovered by a neural network in k2 data. *The Astronomical Journal*, 157(5):169, 2019.

Terrance DeVries and Graham W Taylor. Learning confidence for out-of-distribution detection in neural networks. *arXiv preprint arXiv:1802.04865*, 2018.

Debra A Fischer, Megan E Schwamb, Kevin Schawinski, Chris Lintott, John Brewer, Matt Giguere, Stuart Lynn, Michael Parrish, Thibault Sartori, Robert Simpson, et al. Planet hunters: the first two planet candidates identified by the public using the kepler public archive data. *Monthly Notices of the Royal Astronomical Society*, 419(4):2900–2911, 2012.

Daniel Foreman-Mackey, Benjamin T Montet, David W Hogg, Timothy D Morton, Dun Wang, and Bernhard Schölkopf. A systematic search for transiting planets in the K2 data. *The Astrophysical Journal*, 806(2):215, 2015.

Daniel Foreman-Mackey, Timothy D Morton, David W Hogg, Eric Agol, and Bernhard Schölkopf. The population of long-period transiting exoplanets. *The Astronomical Journal*, 152(6):206, 2016.

Daniel Foreman-Mackey, Eric Agol, Sivaram Ambikasaran, and Ruth Angus. Fast and scalable gaussian process modeling with applications to astronomical time series. *The Astronomical Journal*, 154(6):220, 2017.

Daniel Foreman-Mackey, Rodrigo Luger, Eric Agol, Thomas Barclay, Luke G. Bouma, Timothy D. Brandt, Ian Czekala, Trevor J. David, Jiayin Dong, Emily A. Gilbert, Tyler A. Gordon, Christina Hedges, Daniel R. Hey, Brett M. Morris, Adrian M. Price-Whelan, and Arjun B. Savel. exoplanet: Gradient-based probabilistic inference for exoplanet data & other astronomical time series. *arXiv e-prints*, art. arXiv:2105.01994, May 2021.

Trisha A Hinners, Kevin Tat, and Rachel Thorp. Machine learning techniques for stellar light curve classification. *The Astronomical Journal*, 156(1):7, 2018.

Michael Hippke and René Heller. Optimized transit detection algorithm to search for periodic transits of small planets. *Astronomy & Astrophysics*, 623:A39, 2019.

Michael Hippke, Trevor J David, Gijs D Mulders, and René Heller. Wōtan: Comprehensive time-series detrending in python. *The Astronomical Journal*, 158(4):143, 2019.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Sara Jamal and Joshua S Bloom. On neural architectures for astronomical time-series classification with application to variable stars. *The Astrophysical Journal Supplement Series*, 250(2):30, 2020.

Jon M Jenkins. The impact of solar-like variability on the detectability of transiting terrestrial planets. *The Astrophysical Journal*, 575(1):493, 2002.

Jon M Jenkins, Joseph D Twicken, Sean McCauliff, Jennifer Campbell, Dwight Sanderfer, David Lung, Masoud Mansouri-Samani, Forrest Girouard, Peter Tenenbaum, Todd Klaus, et al. The TESS science processing operations center. In *Software and Cyberinfrastructure for Astronomy IV*, volume 9913, page 99133E. International Society for Optics and Photonics, 2016.

Jon M Jenkins, Peter Tenenbaum, Shawn Seader, Christopher J Burke, Sean D McCauliff, Jeffrey C Smith, Joseph D Twicken, and Hema Chandrasekaran. Kepler Data Processing Handbook: Transiting Planet Search. *ksci*, page 9, 2017.

T Kallinger, Joris De Ridder, S Hekker, S Mathur, B Mosser, Michael Gruberbauer, RA García, C Karoff, and J Ballot. The connection between stellar granulation and oscillation as seen by the kepler mission. *Astronomy & Astrophysics*, 570:A41, 2014.

Sekitoshi Kanai, Yasuhiro Fujiwara, and Sotetsu Iwamura. Preventing gradient explosions in gated recurrent units. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 435–444, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

David M Kipping. Efficient, uninformative sampling of limb darkening coefficients for two-parameter laws. *Monthly Notices of the Royal Astronomical Society*, 435(3):2152–2160, 2013a.

David M Kipping. Parametrizing the exoplanet eccentricity distribution with the beta distribution. *Monthly Notices of the Royal Astronomical Society: Letters*, 434(1):L51–L55, 2013b.

Sebastiaan Koning, Caspar Greeven, and Eric Postma. Reducing artificial neural network complexity: A case study on exoplanet detection. *arXiv preprint arXiv:1902.10385*, 2019.

Geza Kovács, Shay Zucker, and Tsevi Mazeh. A box-fitting algorithm in the search for periodic transits. *Astronomy & Astrophysics*, 391(1):369–377, 2002.

Géza Kovács, Joel D Hartman, and Gáspár Á Bakos. Periodic transit and variability search with simultaneous systematics filtering: Is it worth it? *Astronomy & Astrophysics*, 585:A57, 2016.

Laura Kreidberg. batman: Basic transit model calculation in python. *Publications of the Astronomical Society of the Pacific*, 127(957):1161, 2015.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. Long short term memory networks for anomaly detection in time series. In *Proceedings*, volume 89, pages 89–94. Presses universitaires de Louvain, 2015.

Kaisey Mandel and Eric Agol. Analytic light curves for planetary transit searches. *The Astrophysical Journal Letters*, 580(2):L171, 2002.

BL Canto Martins, Roseane L Gomes, Yuri S Messias, Suzierly R de Lira, Izan C Leão, Leonardo A Almeida, Márcio A Teixeira, Maria L das Chagas, Jenny P Bravo, A Bewketu Belete, et al. A search for rotation periods in 1000 tess objects of interest. *The Astrophysical Journal Supplement Series*, 250(1):20, 2020.

Mario Morvan, Nikolaos Nikolaou, Angelos Tsiaras, and Ingo P Waldmann. Detrending exoplanetary transit light curves with long short-term memory networks. *The Astronomical Journal*, 159(3):109, 2020.

Brett Naul, Joshua S Bloom, Fernando Pérez, and Stéfan van der Walt. A recurrent neural network for classification of unevenly sampled variable stars. *Nature Astronomy*, 2(2):151–155, 2018.

Hugh P Osborn, Megan Ansdell, Yani Ioannou, Michele Sasdelli, Daniel Angerhausen, D Caldwell, Jon M Jenkins, Chedy Räissi, and Jeffrey C Smith. Rapid classification of TESS planet candidates with convolutional neural networks. *Astronomy & Astrophysics*, 633:A53, 2020.

Aviad Panahi and Shay Zucker. Sparse box-fitting least squares. *Publications of the Astronomical Society of the Pacific*, 133(1020):024502, 2021.

Kyle A Pearson, Leon Palafox, and Caitlin A Griffith. Searching for exoplanets using artificial intelligence. *Monthly Notices of the Royal Astronomical Society*, 474(1):478–491, 2018.

Peter Plavchan, M Jura, J Davy Kirkpatrick, Roc M Cutri, and SC Gallagher. Near-infrared variability in the 2mass calibration fields: A search for planetary transit candidates. *The Astrophysical Journal Supplement Series*, 175(1):191, 2008.

Frédéric Pont, Shay Zucker, and Didier Queloz. The effect of red noise on planetary transit detection. *Monthly Notices of the Royal Astronomical Society*, 373(1):231–242, 2006.

George R Ricker, Joshua N Winn, Roland Vanderspek, David W Latham, Gáspár Á Bakos, Jacob L Bean, Zachory K Berta-Thompson, Timothy M Brown, Lars Buchhave, Nathaniel R Butler, et al. Transiting exoplanet survey satellite. *Journal of Astronomical Telescopes, Instruments, and Systems*, 1(1):014003, 2014.

Kai Rodenbeck, René Heller, Michael Hippke, and Laurent Gizon. Revisiting the exomoon candidate signal around kepler-1625 b. *Astronomy & Astrophysics*, 617:A49, 2018.

Yulia Rubanova, Ricky TQ Chen, and David Duvenaud. Latent odes for irregularly-sampled time series. *arXiv preprint arXiv:1907.03907*, 2019.

YJ Rusticus, BH Foing, AM Heras, V Foing, CJ Hönes, and JM Terpstra. Towards deep learning for transiting exoplanet search using simulated tess data. In *Lunar and Planetary Science Conference*, number 2548, page 2080, 2021.

Roberto Sanchis-Ojeda, Saul Rappaport, Joshua N Winn, Michael C Kotson, Alan Levine, and Ileyk El Mellah. A study of the shortest-period planets found with kepler. *The Astrophysical Journal*, 787(1):47, 2014.

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

Jeffrey C Smith, Peter Tenenbaum, Jon M Jenkins, Douglas A Caldwell, Robert L Morris, Mark Rose, Eric Ting, and Joseph Twicken. A four-sector simulated data set for the transiting exoplanet survey satellite. *Research Notes of the AAS*, 3(7):111, 2019.

Peter Tenenbaum and J Jenkins. Tess science data products description document. Technical report, EXP-TESS-ARC-ICD-0014 Rev D https://archive. stsci. edu/missions/tess/doc . . . , 2018.

Adam Wheeler and David Kipping. The weird detector: flagging periodic, coherent signals of arbitrary shape in time-series photometry. *Monthly Notices of the Royal Astronomical Society*, 485(4):5498–5510, 2019.

Edwin B Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927.

Liang Yu, Andrew Vanderburg, Chelsea Huang, Christopher J Shallue, Ian JM Crossfield, B Scott Gaudi, Tansu Daylan, Anne Dattilo, David J Armstrong, George R Ricker, et al. Identifying exoplanets with deep learning. iii. automated triage and vetting of tess candidates. *The Astronomical Journal*, 158(1):25, 2019.

Shay Zucker and Raja Giryes. Shallow Transits—Deep Learning. I. Feasibility Study of Deep Learning to Detect Periodic Transits of Exoplanets. *The Astronomical Journal*, 155(4):147, 2018.

Appendix A

A.1 Conference and workshop contributions

This work has been communicated to the scientific community through presentations and abstracts at several international conferences and workshops. Early on in the project, the abstract titled "Towards Deep Learning for Transiting Exoplanet Search Using Simulated TESS Data"[1] (Rusticus et al., 2021) was submitted to and accepted by the 52nd Lunar and Planetary Science Conference (LPSC). At this conference, both a 3-minute live lightning talk and pre-recorded presentation of 15 minutes were given. At the annual meeting of 2021 of the European Astronomical Society (EAS), the abstract titled "A Transit Detection Algorithm Based on Recurrent Neural Networks"[2] was accepted. This resulted in a live 15-minute presentation in the session for "machine learning and visualisation in data intensive era". Lastly, at several EuroMoonMars (EMM) workshops, presentations of 5-10 minutes were given for ESA scientists, students and collaborators.

[1] https://www.hou.usra.edu/meetings/lpsc2021/pdf/2080.pdf
[2] https://eas.kuoni-congress.info/2021/programme/abstract/1918