University of Amsterdam

MSc Artificial Intelligence
Master Thesis

Detecting Exoplanet Transit Signals in Light Curves Using Recurrent Neural Networks

---

by
Yke Jan Rusticus
11306386

June 28, 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

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. It was only after the 90's that scientists became certain of their existence and nowadays more than 4000 examples of exoplanets are known[1]. Several approaches to exoplanet discovery exist, the most successful of which is the transit method. This method relies on the rare event that a planet moves in front of its host star with respect to our line of sight, a so-called "transit". Such events occasionally occur for Mercury and Venus. 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.

In the following sections, the problem of detecting transit signals is described in further detail, as well as the motivation to approach this problem from different perspectives. Limitations of common approaches are discussed, and we describe how the solution to the problem might benefit from artificial intelligence (AI). The contributions and outline of this work are given in the last sections of this chapter.

1.1 Problem

In short, the problem addressed in this thesis is that of detecting dips in stellar brightness that could indicate the presence of an transiting exoplanet. The signal may repeat itself due to the periodicity of the planet's orbit, but it may also only occur once if the observation span is shorter than the orbital period of the planet. These candidate transit signals are searched for in data we refer to as *light curves*, which describe the observed brightness, or flux, of stars over time. However, while the physics of transits is well-understood, the detection of their signatures in light curves is hindered by stellar and spacecraft induced noise. Furthermore, transit signal shapes vary, and the chance that any given light curve contains detectable transit signals is low, especially in the case of small, Earth-like planets. For a full description of the challenges and their origins, see Section 2.2.2.

In the problem of detecting transit signals in light curves, we aim to find an algorithm that is capable of specifying a reference time t_0 and the period P (if possible) of potential exoplanet candidates, given an input light curve. The reference time t_0, also referred to as epoch, is generally the mid-transit time of the first transit event in the light curve. These parameters are the minimum requirement to allow follow-up research to retrieve the signal that was detected.

1.2 Motivation

Within the course of a few decades, the status of exoplanets has changed from hypothetical to common. The discovery of exoplanets has greatly enriched our view on the universe, as with each new discovery we can better answer questions regarding, for example, the distribution of exoplanet compositions or planetary system dynamics and evolution. The search for a second Earth is still going, and with that the search for life. For these reasons, research is always pushed to find undiscovered planets.

Existing approaches to exoplanet discovery each come with their own benefits and drawbacks (see Section 2.1.2). This thesis focuses only on the transit method, which is motivated by several reasons. Firstly, the problem to solve is relatively straightforward, namely detecting dips in brightness over time. Secondly, plenty of data is available for this method, because the only thing to be measured is the brightness of stars over time, which can be done at a single wavelength interval for many stars simultaneously. Lastly, even though the method is straightforward, a transit signal carries a wealth of information about the planetary system at hand.

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

For example, the depth of the signal is related to both the size of the planet relative to its host star and several orbital parameters such as the inclination of its orbital plane. The periodicity of the signal directly tells us the orbital period of the planet, although even the duration of a single transit signal can already constrain the orbital period. More information about the transit method and what we can learn from it is given in Section 2.1.2.1.

Before we can determine all these parameters, however, the signal need to be detected. Several algorithms exist to search for transit signals in light curves, but as each algorithm can differ, so can their returned detections. One algorithm may miss a planet that another algorithm finds and vice versa. For this reason, we aim to develop a new transit detection algorithm which works substantially different from existing methods, that is capable of providing complementary detections.

1.3 Common approaches and limitations

An intuitive approach to search for dips in light curves is to compare at each point in time the flux within a small window with the background flux. A large enough difference could indicate a potential transit signal. In fact, this is similar to what some commonly used approaches do. The same result can be achieved by fitting a box function to the data, representing the transit signal. Better results could be obtained by fitting more realistic transit shapes to the data.

However, there are two main problems with these approaches. First of all, the transit shape is not known beforehand and can vary in duration, depth, and other features (see Section 2.2.2.5). Iterating over all possible combinations of these parameters can be costly. Another problem is that the background flux is not constant, instead it can be dominated by time-dependent noise. An important step in these algorithms is therefore the detrending step, in which is attempted to remove all unwanted time-dependent noise from the light curve prior to the search for transit signals. However, this operation comes with risks. Altering the input light curves may also harm the transit signals that we wish to find, because the signals are often intertwined with the noise.

AI comes with the potential of avoiding these problems, because background patterns and the shapes of transit signals can be learned beforehand. In the field of exoplanets, artificial intelligence is increasingly being applied, though only very limited in the task of transit detection.

1.4 Contributions

We report the first attempt in literature of using recurrent neural networks (RNNs) for the task of detecting transit signals in light curves. This particular network can handle arbitrary input sizes and can provide predictions at every time step of the light curve. The properties are ideal for detection as they allows us to locate potential signals in time for any given light curve. Furthermore, during the training phase, different transit shapes can be fed to the network. When searching for new transit signals, this prior knowledge is taken into account by the network, so there is no need to iterate over transit durations, depths and shapes anymore. Furthermore, the network can be trained to distinguish these signals from complex background patterns without requiring the input to be detrended.

In this thesis, we investigate whether an RNN-based approach to detecting transit signals is viable and competitive as compared to more classical approaches. Successes and shortcomings are identified using simulated data, and potential extensions to this approach are explored. These extensions include the use of confidence estimation over the RNN predictions for better monotransit detection and the use of hidden representations of transit signals to disambiguate multiple repeating signals in a single light curve.

[TODO: list findings and highlight potential for monotransit detection]

This work can be used as reference for astronomers to how AI can help solve problems in possibly previous unforeseen ways, and for AI researchers to how a well-established network design can be used in new ways for complex and important problems outside the scope of planet Earth.

1.5 Outline

Since the problems and challenges in exoplanet science differ substantially from more "typical" AI-problems encountered on Earth, a large portion of Chapter 2 is devoted to create a common ground. Exoplanets are described in further detail, as well as noteworthy discoveries, discovery methods and facilities. A detailed description of the challenges of detecting transit signals is given, and we discuss previously proposed approaches to the problem. Furthermore, we discuss work on AI applied to different problems in exoplanet science, and other related work. Chapter 3 describes our proposed method, as well the simulated data that was used in the experiments. In Chapter 4, the RNN-based detection algorithm is evaluated and compared to other methods,

such as the Box Least Squares (BLS) algorithm Kovács et al. (2002). The results are discussed in Chapter 5 and in Chapter 6 the main conclusions are given and the work is summarized.

Chapter 2

Background

[TODO: write short chapter summary and describe structure]

2.1 Exoplanets

Most people are familiar with the planets in our solar system, e.g. Venus, Mars, Jupiter. What these planets have in common is that they all orbit the same star, our Sun. Exoplanets may be similar to the planets we know well, but they orbit different stars, which makes them hard to find and difficult to study.

2.1.1 A brief history of exoplanet science

For a long time, Earth was one of the few planets known to human kind, as part of the only known planetary system. However, billions of stars exist within our galaxy alone, each potentially with their own orbiting exoplanets. Since the first detection of an exoplanet, many more exoplanets were discovered, and thus the list of known planetary systems was extended. [TODO: list noteworthy discoveries]

2.1.2 Discovery methods

[COMMENT: this section "Discovery methods" still needs a lot of work, as well as its subsections] Exoplanets are not as straightforward to observe as planets in our own solar system. For example for Mars or Jupiter, light reflected from the planet's surface can be observed with the unaided eye. To observe Neptune, a small telescope is sufficient. For exoplanets however, the distances between stars play a role. Distances between planets and their host star are small compared to the distances between stars. The distance between Earth and the Sun is 1 Astronomical Unit (1 AU $= 1.496 \times 10^8$ km) and the distance to the nearest star is about 10^5 AU. First of all, this means that any exoplanet will thus be much farther away than the planets in our solar system. Second, we cannot observe exoplanets in isolation from their environment as we can do with planets in our solar system. Nevertheless, several methods have successfully been used to discover exoplanets, a few of which are described in the following.

2.1.2.1 Transit method

[COMMENT: a lot still needs to be (re-)written for this section, and I will include figures to guide the intuition for the parameters] [TODO: describe the requirement for an edge-on system]. [TODO: describe periodicity and transit shapes in general]. The most successful method in terms of number of discovered exoplanets is the transit method. This method forms the basis of this thesis. As illustrated in Figure **??**, an exoplanet can move between the observer and the star it orbits. If the observer measures the brightness of that star over time, a dip is observed in the resulting *light curve*. The depth of this transit signal depends on the relative size of the planet compared to its host star. If the radius of the star is given by R_*, then the stellar disk has a surface area of πR_*^2. During transit, the exoplanet blocks the star light that is emitted from an area of πR_p^2 from the stellar surface, where R_p is the planet's radius. Therefore, during an ideal transit, a a fraction R_p^2/R_*^2 of the star's emitted light is blocked by the exoplanet.

2.1.2.2 Transit timing variations

[TODO: main points are that this method uses same data as transit method, and is used to find additional planets in systems with known exoplanets]

- perturbations in transit timing hint at presence of additional companion
- few discoveries using this method, but added here because it relates to transits

2.1.2.3 Radial velocity

[COMMENT: not finished, I will write more about the function this method has in confirming exoplanets, and in general how this method differs from the transit method in terms of data used, etc.] By observing the light of a star at different wavelengths, its radial velocity can be inferred from shifts in the resulting spectrum. A change in radial velocity over time can be explained by the mutual interaction between an orbiting body and the star, causing the star to to move back and forth slightly in our line of sight. This method can also constrain the masses of the objects within the system.

2.1.2.4 Direct imaging

[TODO: extend] This method aims to directly image a planet and resolve it from its host star. Although the idea is simple, in practice this method is challenged by the fact that the observed light from distant planetary systems is greatly dominated by the light emitted by the host star.

2.1.2.5 Microlensing

A microlensing event occurs when a compact body such as a star acts as a lens and magnifies a more distant object in the background. This is due to the fact that the path of a ray of light is bent when it passes a massive body. In case a foreground star passes a background star in our line of sight, this effect should be visible for some duration as a peak in their combined brightness. Based on the signature of this change in brightness, one could infer whether a planet is likely to orbit the foreground star.

2.1.3 Exoplanet-hunting missions

Since the discovery of the first exoplanets, the rate at which exoplanets are detected has taken an exponential boost. Where at first it was the radial velocity method, it is now the transit method that holds most discoveries. [TODO: removem the next part or make easier to read] At the time of writing, about one exoplanet is discovered every day over the past ten years on average by using the transit method. For all other methods combined, on average about one exoplanet is discovered every six days. The major turning point was when the search for transiting exoplanets was taken into space, and stars were monitored by the hundred thousands.

2.1.3.1 Ground-based surveys

[TODO: write]
- e.g. WASP, HAT, TRAPPIST, KELT

2.1.3.2 CoRoT

[TODO: write]
- first space-based mission designed for transit search
- 33 exoplanet discoveries

2.1.3.3 Kepler and K2

With over 2500 exoplanet discoveries, Kepler and its continued mission "K2" are responsible for the detection of more than half of the known exoplanets. The Kepler space telescope has observed half a million stars in a small section of the sky. For most targets the cadence, or observation interval, was 30 minutes. A small fraction was observed with a cadence of one minute. Kepler operated for nearly ten years. Since its retirement in 2019, Kepler data is still a source for exoplanet discovery, as hidden and previously overlooked transit signals may yet to be detected (Hedges et al., 2019).

2.1.3.4 TESS

Launched in 2018, the Transiting Exoplanet Survey Satellite (TESS) has taken over the work of Kepler. As opposed to Kepler, TESS observes the full sky during its mission in the search for transiting exoplanets. It observes more brighter and closer targets than Kepler. TESS observations are of 2-minute cadence, though some targets are observed with a cadence of 20 seconds. Every 27.4 days, or sector, TESS changes its field of view. About halfway through each sector TESS links data down to Earth, during which no observations are

made. At the time of writing, 125 exoplanet discoveries have resulted from TESS observations. With already over 200.000 stars observed, TESS is expected to discover thousands more.

2.1.3.5 PLATO

The amount of data in the field will only increase more rapidly, as future telescopes will get bigger and better. The PLAnetary Transits and Oscillations of stars (PLATO) mission will increase the numbers of several aspects. Planned for launch in 2026, this space-based telescope consisting of 34 [TODO: fix this number, I believe it was less] individual cameras will monitor up to a million stars. Its mission is to discover new transiting exoplanets and study their properties in greater depth. It will also perform asteroseismology to determine stellar parameters such as their mass. In combination with radial velocity measurements from the ground, this will allow for better characterization of the unknown worlds.

2.2 Process of discovering transiting exoplanets

Observing a transit event is one thing, finding the transit signal in a light curve is another. This task is simple if the existence of a planet is known beforehand, but this is not true in our case, so we need to search for signals. Once potential signals have been detected, the process continues, as the nature of each signal needs to be validated. The steps from raw light curves to the discovery of new exoplanets are described in the following subsection, after which follows a detailed description of the challenges that need to be overcome in the process.

2.2.1 Detection, identification and characterization

Apart from preprocessing the light curves, the detection of transit signals is the first step in the process of discovering transiting exoplanets. In literature, however, some ambiguity exists between the detection step and the step that follows detection. We follow the categories used by Jara-Maldonado et al. (2020) (as well as other works [TODO: cite these]) and emphasize the difference between detection and identification.

In the detection step, which is the focus of this thesis, transit signals are blindly searched for without prior knowledge of their timing, shape or existence in general. Epoch t_0, period P (if possible) and possibly other parameters of potential signals are determined by the detection step and are then passed to the next step.

Identification, or vetting, is the processes of filtering out false positives from the returned detections. Often this is a binary classification task, where the isolated transit signal is classified as planet candidate or false positive. The main difference with the detection step is that during identification, no new potential signals are being searched for. Admittedly, one could apply an identification model to every part of a given light curve to search for new signals, although in most cases this is not what the algorithm was designed for. Furthermore, this would likely results in an unnecessarily inefficient detection algorithm.

Once a potential signal has passed the identification step, this still does not mean we have found a new planet. According to Akeson et al. (2013), a candidate exoplanet must meet the following criteria to be included in the Exoplanet Archive: (1) it has a (minimum) mass estimate of no larger that 30 Jupiter masses, (2) its properties are described in the peer-reviewed literature, and (3) sufficient follow-up observations and validation have been undertaken to deem the possibility of a false positive as unlikely. For example, after the transit method is applied, one could attempt to confirm and characterize the system further using radial velocity measurements and more accurate modelling techniques.

2.2.2 Challenges

Although a transit signal is clearly defined, it is often a difficult task to detect or study them. This is mostly because the signals are usually weak and the background varies at greater amplitudes than the signal. These and other challenges are explained in detail in the following subsections.

2.2.2.1 Small chances and weak signals

For the transit method to work, a planetary system is required to be oriented edge-on (i.e. have an inclination $i \approx 90°$). If the system is seen face-on ($i \approx 0°$), any orbiting planet would not transit its host star with respect to our line of sight. However, no mechanism restricts planetary systems from being oriented randomly, so even if all stars would have orbiting planets, a large portion would go undetected. The chance of a planet transiting also depends on the distance from its host star. The farther away a planet is from its host star, the smaller the chance of that planet to transit the star. If one were to observe our Sun from far away, the chance that Earth would transit the Sun for a random orientation of our solar system is only 0.47%. Furthermore, even if it would transit, its 13-hour signal would only repeat once per year and leave a dip in the Sun's observed brightness of only about 80 ppm, or 0.0008%. Hence, the transit method is biased towards detecting short-period Jupiter

sized planets, as these cause transit signals that are more frequent and more profound. Nevertheless, the goal is also to detect Earth-like planets, which are small and have larger distances from their host star. For this to succeed, the background flux needs to be accurately modeled in order to disentangle it from the transit signals.

2.2.2.2 Stellar activity

Part of the reason why the background flux in a light curve is not constant is due to stellar activity. Stars, like our Sun, are not uniform spheres that emit a constant amount of photons at all times. First of all, starspots could be present. These visually dark regions on the stellar surface are relatively low in temperature, and thus decrease the flux. On the other hand, higher temperature regions can increase the flux. This alone would be no problem if the spots would always be present. However, starspots come and go with a lifetime depending on their size. Moreover, the star itself can be rotating, with the result that any spots that are present come in and out of sight as it rotates. This effect is called rotation modulation.

To complicate things further, a star's surface is dominated by cell-like structures called granules, which are caused by the energy transport below the stellar surface. As hot and therefore brighter plasma rises, cooler and dimmer plasma descents, which results in a constantly changing surface with a fluctuating brightness. These effects can closely mimic transit signals in a light curve, and thus often lead to false positive detections. Other stellar activity includes for example, stellar oscillations and sudden outbursts of energy, or flares.

In summary, a light curve may exhibit quasi-periodic patterns that are caused by complex mechanisms at local and global scales on and below the stellar surface. [TODO: compare amplitudes of fluctuation with transits, see Barros et al. (2020)].

2.2.2.3 Instrumental and photon noise

No instrument is perfect, neither is the setup for observations. For ground-based observatories, weather could play a role and measurements might be affected by stray light from nearby cities. In this thesis however, we assume all data to come from space-based observatories. These are not affected by the obstacles mentioned above, but still have imperfections. For example, temperature changes in the spacecraft can alter the pointing of the telescope, resulting in jumps or drifts in the observed flux from stars. As the telescope typically observes thousands of targets at the same time, this affects all the thousands of corresponding light curves, though potentially in slightly different ways. Instead of stray light from cities, stray light from the moon may still negatively influence the quality of measurements.

Poor quality data is often filtered out, leaving gaps in the resulting light curves. This could pose a problem for certain data processing algorithms. Larger gaps could result from when the spacecraft transmits data back to Earth. During this process, the telescope might not gather new data, causing the light curves to have gaps of several hours to a few days.

Another source of noise, although not due to instrument imperfections, is photon noise. This type of noise originates from the arrival of different amounts of photons at the detector between different observations. The number of arrivals per observation can be modeled as a Poisson distribution, which for a large number of photons approximates a Gaussian distribution. The photon noise can thus often be seen as the time independent white noise in the data.

2.2.2.4 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, the corresponding transit signals will be larger than for exoplanets, but if the stars' orbital plane is slanted and they have grazing transits, the signals can become more similar to exoplanet transits.

More similar to exoplanet transit signals can be due to an EB system lurking in the background of an observed star. These background EBs (BEBs) can be much farther away than the star under consideration, but contribute to the star's light curve. The BEB transit signals will thus also be smaller 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. This can well be done in the identification step. Similarly, the detection step could return a signal that is in fact caused by a large exocomet or asteroid. Restricting our detection algorithm too much on what kind of signals it is allowed to pick up, may exclude various interesting signals like these. In many cases, however, it would be of great astrophysical value to find and study those signals.

2.2.2.5 Transit signal shapes

The basic shape of a transit signal is determined by a set of parameters. These parameters include, among others: the orbital period of the planet, its radius relative to its host star, the distance from its host star, the inclination of its orbital plane and the eccentricity of its orbit. Each of these affects the transit shape differently. For example, the inclination determines if the signal will be U-shaped or V-shaped, and the eccentricity has an effect on the duration of a transit. It is not the task, however, of the transit detection algorithm to determine all these parameters. Nonetheless, one needs to ensure that the algorithm is robust against differences in shapes between transit signals. Some other sources of differences in transit shapes are discussed below (excluding rarer phenomena such as gravity darkening effects).

Stellar limb darkening

Although most patterns caused by background stellar activity are simply overlayed with the transit signal, some phenomena caused by the star only have an effect in combination with an exoplanet transit. One of these phenomena is stellar limb darkening, which only shows its effect on a light curve when a planet moves in front of the star. The deeper into a star's interior, the hotter it gets. As an exoplanet passes in front of a star, it first passes its edge, or limb. From our perspective, the limb region of a star is less deep than the central part of the stellar disk. For this reason, the stellar limb appears dimmer than the central part. As the exoplanet proceeds, it will thus block an increasing fraction of the emitted light, until it passes the center of stellar disk, after which the fraction of blocked starlight decreases again. This effect will cause the transit depth to not be constant, but instead be slightly changing over time with a maximum depth at mid-transit time. The limb darkening effect is also dependent on the wavelength at which is observed.

Occulted starspots

Although starspots by themselves can leave their imprint on a light curve, and thus on the background of a transit signal, in some cases the combination of starspots and transits lead to interesting changes in transit signal shapes. In case a starspot is occulted by the transiting exoplanet, the fraction of blocked starlight is relatively low, so the transit signal will have a positive bump for some duration, meaning the transit depth at that point is smaller than one would expect. In this thesis we do not consider occulted starspots in our data or evaluation. The proposed method could, however, still be applied in the case of transits in combination with occulted starspots, as distorted transit signals could be learned by the RNN prior to the search.

2.2.2.6 Data volume

Provided the data is of good quality, more data is often preferred. Especially machine learning algorithms are known to work well in the large-data regime. However, it could be challenging to work with the large amounts of data the exoplanet-hunting missions produce. For example, assume we have 35 sectors ($\sim$2.5 years) of TESS data, each incorporating the light curves of 20.000 stars. Assuming a 2-minute cadence, each light curve consists of about 20.000 measurements. If we only consider the flux values, we already have about 14 billion data points, or 112 GB worth of data. One could even choose to include additional information such as measurement errors, or information about the pointing of the telescope at each time step, which further increase the data volume.

2.3 Transit signal detection methods

[COMMENT: from here onwards up to Methodology, the text was written a while ago. I am satisfied with the first few sections, but from section 2.5 (RNNs) I need to change/add a lot still.]

[TODO: write better section introduction] When Kepler or TESS data are released, both the data and the initial analyses by their data processing pipelines are released. These analyses include the detection of potential transit signals. In the following, we describe how the pipeline detections come about and discuss several other approaches that have been proposed in literature to detect transit signals.

2.3.1 Least squares: BLS and TLS

The Box Least Squares (BLS) algorithm, proposed by Kovács et al. (2002), is commonly adopted for transit signal detection [TODO: cite examples]. The transit signal is approximated by a two-level signal with a maximum of, say 1, and a minimum of $1 - \delta$ in units of normalized brightness, or flux. This box signal is parametrized by its depth δ, duration, period and reference time, or epoch. The search is performed by trying many values for each of these parameters, and finding minima in the squared error between the fit and the data. In order for this to work well, the light curve is expected be detrended prior to the search. Subsequently, a detection can be

defined using a threshold on the signal-to-noise ratio (SNR) of the fit, or other metrics. The SNR of a transit event would intuitively be defined as SNR $= \delta/\sigma_w \cdot \sqrt{n_t}$, with n_t the number of measurements that belong to the transit signal and σ_w the estimated white noise in the detrended light curve.

However, according to Pont et al. (2006) the time dependent "red" noise in light curves also affects transit detection. This red noise includes, for example short scale stellar activity such as granulation, which could mimic transit signals. They redefine the SNR of a transit signal by including both the white noise and the time dependent noise on the time-scale of the transit duration, and provide an algorithm to estimate these values. Their results show that a detection threshold for BLS can be set better if red noise is accounted for.

In later years, variants to the BLS algorithm were proposed. For example, Carter and Agol (2013) relax the assumption of strictly periodic signals, and use a quasi-periodic transit function that is matched with the data.

More recently, the Transit Least Squares (TLS) algorithm was propsed (Hippke and Heller, 2019), which is similar to BLS in that it fits a function to the data by trying many parameter settings. 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.

Both BLS and TLS require the light curve to be detrended prior to the search, which can be seen as a downside to the simplicity of both algorithms. Their performance therefore greatly depends on the way the input light curve is detrended. 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 which is used as default by the light curve processing library Lightkurve, and more. The authors show how detrending could significantly alter or reduce the transit signals, for some methods more than other. Furthermore, for window-based filters, it was found that a window size of three times the transit duration works best, but that is generally not known beforehand.

2.3.2 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 each light curve was 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.

On the other hand, 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.

2.3.3 TESS and Kepler pipeline

Although the pipelines of TESS and Kepler cover the processing of raw images to systematics corrected light curves, we focus only on the method that is used to search for transit events. 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 different sectors of data together and filling gaps. Subsequently, a time-varying whitening filter is applied, so the light curve is transformed to the time-frequency domain. This pre-whitening filter is supposed to clear the light curve of time dependent, or colored, irrelevant 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 transit pulse train that is used as a match filter is whitened by the same filter. After single events have been determined, a grid of periods, durations and epochs is searched through to find a least squares fit to the signal. A reprentative limb darkening model is used to better match the trial signal with the data. [TODO: describe differences TCE and TOI and KOI]

2.3.4 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 the transit signal this is in general not the case as its duration is short relative to its period. For ultra-short-period (< 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. Sanchis-Ojeda et al. (2014) use a Fourier-based method to detect transits from ultra-short-period 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 by using 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 shoothed 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.

2.3.5 "Intelligent" algorithms

The human eye can function as an excellent tool for pattern recognition and can sometimes exceed algorithms in complex ways. For this reason, a citizen science project was launched, called Planet Hunters. A participant, or user, is presented with a light curve from Kepler and is asked to flag any part of the light curve that resembles a transit signal. After six months since the launch, millions of classifications were made and later Fischer et al. (2012) reported the detection of first two planet candidates that were flagged by participating volunteers. We can view this process as a detection algorithm: a pipeline corrected light curve is presented to the user, say detector, which has prior knowledge about the shape of the signal that we are interested in. 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 last 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 artificial intelligence for the task of transit detection, as proposed by Pearson et al. (2018). In this work, a one-dimensional convolutional neural network (1D-CNN) was trained to classify light curve segments either as containing a transit signal or not. On its own, this approach could be categorized as identification method, as their model can only be applied to small inputs of fixed size (180 data points), and provides a single binary output for a given input. To use the 1D-CNN for detection, two approaches are proposed. In the first approach, the model is applied to the whole light curve by using a sliding window, to obtain a prediction at each point in time which is referred to as the probability time series (PTS). Subsequently, 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 a given period, after which the sliding window approach is applied to the resulting phase curve. A detection in the phase curve directly gives an estimate of the periodicity, but this approach requires a brute-force search over period values to find the one for which the signals overlap in the phase curve.

Extending on this work, Chintarungruangchai and Jiang (2019) present a similar method. However, in their approach the light curve is folded, such that the result is not one, but two dimensional. This is accomplished by stacking each part of the folded light curve that would normally be interleaved. Subsequently, they train a 2D-CNN to classify these "images" as signal or non-signal. The search for a signal still requires trying different values for the period over which the light curve is folded, but this methods should be more robust to small errors between the trial and the true period.

Zucker and Giryes (2018) test the feasibility of using 1D-CNNs for transit signal detection in comparison with the BLS algorithm. They use simulated data for which Gaussian processes are used to model stellar variability. For a given input light curve, the tested methods both output a single binary value (signal, non-signal), which was used for the analysis of their performance. However, no comparison was made between BLS and the 1D-CNN in their ability to explicitly detect a transit signal from a given planet. In other words, the question remains whether a CNN-based algorithm could compete with the BLS algorithm if the task was to output the period P and epoch t_0 of a detected signal. These values allow one to know exactly which signal was detected and when its repetitions can be expected in the future.

2.4 Identification methods

While the focus of this thesis lies on detection, a brief overview of some of the existing work on identification is given both for completeness, and to give the reader a better sense what machine learning methods have been applied in this field.

The identification of candidate signals is in general a binary classification problem, i.e. signal or non-signal. For this reason, the 1D-CNN of Pearson et al. (2018) can be be considered an identification model. In fact, Pearson et al. (2018) provide a comparison between several models for the task of identification (even though it is referred to as "detection"). These models include a support vector machine (SVM), multi-layer perceptron (MLP), a box function as matched filter, an MLP that takes the wavelet decomposition of the light curve as input, and their 1D-CNN. This work inspired Shallue and Vanderburg (2018) to develop *Astronet*, a 1D-CNN optimized for transit identification. Several papers followed that describe applications of or variations to *Astronet*. For example, with and without alterations to the model, Dattilo et al. (2019) applied *Astronet* to K2 data, Osborn et al. (2020) and Yu et al. (2019) applied the model to TESS data Ansdell et al. (2018) added scientific domain knowledge to the model such as centroid data and stellar parameters. Koning et al. (2019) proposed methods to reduce the network complexity.

Instead of supervised learning, Armstrong et al. (2016) made use of self-organizing-maps (SOMs), which are trained to cluster the data in an unsupervised manner. Other methods which do not rely on neural networks, but on decision trees and random forests, are *Autovetter* and *Robovetter* (Catanzarite, 2015; Coughlin, 2017). Thompson et al. (2015) identify transit signals using a method based on k-nearest neighbours.

Several more works exist that compare different models, e.g. Schanche et al. (2019) compare for example an SVM, random forest classifier, k-nearest neighbours and logistic regression. Lastly, Jara-Maldonado et al. (2020) compare a wide range of identification models. In addition, they extend the models to take as input the wavelet decomposition, or multi-resolution analysis (MRA), coefficitions instead of the raw light curve, and motivate how MRA can improve transit identification.

2.5 Recurrent Neural Network (RNN)

[TODO: write short section introduction; include more RNN-related work (e.g. signal detection in ECG data)]

2.5.1 Application to exoplanets

In a task that resembles detection, Hinners et al. (2018) compared a bi-LSTM with a feature engineering approach. Among other tasks, the task was to predict the number of transits present in a given light curve. Their bi-LSTM model produced near-random results. Other tasks included the prediction of parameters of the star given its light curve. They address the problem of the RNN-based model could be due to the fact that their data set was imbalanced. However, we note that it might 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 receives one learning signal over its single output value. This way, figuring out where transit signals are present in a given light curve can only be implicitly learned by the model, while we can also explicitly provide this information as learning signal at every time step. This idea has motivated us to develop the RNN-based transit detection algorithm presented in this work, which is further described in Chapter 3.

In another work utilizing LSTMs, Morvan et al. (2020) show the successful application of LSTMs for detrending. The model was trained to learn out-of-transit patterns, and interpolate these within the duration of a given transit signal. By doing so, the background signal could be better subtracted from the signal, and thus the transiting planet better characterized. Results improved if centroid data was included.

2.5.2 Application to variable stars

Remaining literature mostly focuses on light curve classification, in general for variable stars. For example, Jamal and Bloom (2020) compare different neural network architecture for this 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.

Naul et al. (2018) address the problem of irregularly sampled data in light curves by including time intervals between measurement as input to an RNN encoder. The RNN is used for variable star classification. Becker et al. (2020) also experiments with time intervals between measurements as input to the model, and instead of using the raw flux they also use the differences between subsequent flux values as input. To reduce computational requirements, they adopted a sliding window approach over the input light curve, of which each window is used

seperately to train the RNN. While most works employ a bidirectional RNN, in this work a unidirectional RNN was used to allow for its application in an online fashion.

2.5.3 Anomaly detection

Transit signals can be seen as anomalies in time series of "normal" stellar behaviour. LSTMs have been used to detect anomalies in time series in an unsupervised learning approach by Malhotra et al. (2015). In this case, the LSTM is trained to predict the normal data and the errors on the predictions are used to threshold the detection of an anomalous event. If the LSTM is trained to predict multiple steps into the future, the same approach can be used to detect collective anomalies (Bontemps et al., 2016), for example transit signals which generally cover multiple data points. However, as Cherdo et al. (2020) noted, this 'unsupervised' approach to using LSTMs for anomaly detection still requires training on normal data, i.e. we need to know beforehand which data is clear of anomalous events. In our case, we might never be fully sure to whether a light curve is clear of transit signals, e.g. small and previously unseen signals might still be hidden in the data. Training the model to treat these light curves as "normal" data, might seriously harm the ability of the model to detect the smallest transit signals. For our purposes, we therefore turn to the supervised approach, as we can utilize both the transit signals from known exoplanets as well as realistically simulated signals as inputs to our model.

2.5.4 Irregularly-sampled time series

Although attempts have been made to handle irregularly sampled time series using RNNs for variable star classificiation, developments within AI can provide different perspectives on the problem. Not related to variable stars but related to handling irregular time series using RNNs, is the ODE-RNN Rubanova et al. (2019). This model design, inspired by Chen et al. (2018), treats time as continuous, whereas the standard RNN treats time as discrete steps. The ODE-RNN can handle arbitrary time gaps between observations, which could be useful for our purposes. However, this comes at the cost of a required 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 are most profound if the sampled data is sparse. With transit durations in the order of hours and a observation cadence in the order of minutes, this is generally not the case. For this reason, we rely on the standard RNN architectures in this work, and leave the application of ODE-RNNs for exoplanet science for future work.

2.6 Other relevant work in AI

[TODO: write section introduction, and extend on the subsections; make the flow more natural, and include more related work]

2.6.1 Uncertainty in neural networks

Although the topic of uncertainty can be a thesis subject on its own, we briefly touch on the subject as uncertainties could play a role in the task of transit detection. For example, if multiple candidate transit signals are detected and detector's response is similar for all, we might wish to accept only the signals for which the detection uncertainty is below a certain threshold. For neural networks, though in some cases their output has some probabilistic meaning, in general one must be cautious with interpreting their outputs as probability. If unseen out-of-distribution data is presented to the network, which differs substantially from the training data, its outputs might not make sense anymore. [TODO: explain: why would out of dist data occur?] Instead, it would be convenient if the network outputs an additional value that indicates the confidence of a certain classification.

Gal and Ghahramani (2016) describe how dropout with repeated application of the network can be used to obtain uncertainty estimates over predictions. Although this approach is appealingly simple, a few problems remain. First of all, this approach might not solve the problem, because for each instance of the model the data would remain out-of-distribution, and might trigger an erroneous detection in each case with low uncertainty as a result. Furthermore, as RNNs are already less efficient than for example CNNs, the need for repeated application might harm their scalability considerably. Obtaining reliable uncertainty estimates for RNNs is not a straightforward task. Work on uncertainty estimation for RNNs is still being published (Alaa and Van Der Schaar, 2020; Hwang et al., 2020; Wang et al., 2020). For our purposes, we use the method proposed by DeVries and Taylor (2018) and explore it for our task. Their approach is to adjust the loss function that is used for training, such that the network learns for which inputs it is more certain of its outputs and vice versa. The details and integration of this method in combination with our RNN is described in Chapter 3.

Chapter 3

Methodology

Since this work makes use of simulated light curves, the following sections describe in detail how the data was obtained that was used. Subsequently, the network design and training procedure are described as well as the extensions that were explored. Baselines and specific parameter choices which are dependent on the experiments, are described in the corresponding sections in Chapter 4.

3.1 Simulated data

Data from missions such as Kepler and TESS are publicly available. Light curves from these missions are, however, difficult to work with from scratch. They contain data gaps, outliers, measurements at slightly varying time intervals. More importantly, they could contain undetected transit signals, or noisy patterns falsely identified as transit candidates. Using only these light curves for the development and validation of our method is therefore undesirable. For the purpose of this thesis, we therefore use simulated data, as these provide us with a ground-truth, and full control over parameters.

Two sources of simulated data are used, the first of which was developed specifically for this thesis. For clarity we refer to our own simulated data and the simulator that was used to produce these as "LCSim" (Light Curve Simulator) in the following. The second source is the openly available Lilith-4 data set[1], produced by the Lilith data simulator of the TESS pipeline.

3.1.1 Generating artificial light curves

In order to generate realistic data, one needs to consider the components and challenges described in Section 2.2.2. For the development and evaluation of new algorithms, it is convenient if certain features can easily be included or excluded. The LCSim simulator is built so that features such as stellar variability can be turned on and off, or even tuned. In the following, all components are described that make up the LCSim light curves, as well as the distributions used for sampling stellar and planetary parameters. Note that the aim of this part of the work was not to design the most realistic simulator, but rather to be able to generate data that is realistic enough for the analysis and comparison of new signal detection algorithms. The code is made publicly available[2], in part for transparency, but mainly to encourage the development of new algorithms for transit detection using LCSim light curves. [TODO: structure code and write proper readme file]

3.1.1.1 Cadence

In line with TESS data, we adopt a 2-minute cadence for all simulations in this work. This means that the sampling interval of flux values, or observations, is 2 minutes. For other work, one may choose a cadence of 30 minutes, in order to obtain light curves that are closer to Kepler data. For more realistic time intervals, one could copy the time arrays of real-world light curves and pass them through LCSim, which will overlay each time step with simulated flux values.

3.1.1.2 Stellar variability

[COMMENT: this section needs revision, as this was written (and done) a long time ago]
[TODO: improve overview and celerite intuition]

[1] https://archive.stsci.edu/missions-and-data/tess/data-products/lilith-4
[2] https://github.com/ykerus/transit-detection-rnn

The physics of stars is still an active region of science. Stars are complex (see 2.2.2), so they are not straightforward to model. Gaussian processes (GPs) are used several times in literature to account for background patterns in light curves, both in the context of characterization and simulation of the stellar activity (Barros et al., 2020; Zucker and Giryes, 2018). Given their ability to simulate quasi-periodic and stochastic behaviour also observed in stars, we use GPs to account for the stellar variability in our simulated data.

Similar to a Gaussian distribution, which is defined by a mean and a standard deviation or covariance matrix, a GP is defined by a mean function and a covariance function, or kernel. In our case, we define the mean function to be 1 at each time step, so the behaviour of the GP is fully determined by its kernel. We can get an instance of the function representing stellar variability by taking a sample from the multidimensional Gaussian distribution defined by our mean and covariance function. Sampling multiple times from the same distribution will give us unique samples. However if we use the same kernel for each of the samples, then they will still have the same properties in terms of variability, e.g. amplitudes and timescales. To generate truly unique samples, we define a kernel for each light curve separately. This can be a costly process, as for each light curve consisting of 20000 data points, a naive approach would require the construction of a kernel with 20000×20000 entries, which would be around 3 GB to store in memory.

Instead, we make use of "celerite" Foreman-Mackey et al. (2017) to construct kernels and sample from GPs. celerite allows for efficient implementation of GPs and is optimized for one-dimensional astronomical data. The kernel we use for each light curve consists of two terms: one for simulating the short-term variability due to granulation, and one for long-term variability due to rotation modulation. The former is a kernel representing a stochastically-driven damped harmonic oscillator (SHO). The SHO term has corresponding parameters ρ for the undamped period of the oscillator, τ for the damping timescale of the process and σ_{SHO} for the standard deviation of the process. [TODO: give formulas and meaning of other parameters such as Q, which are implicitly defined by the above three parameters]. A value $Q = 1/\sqrt{2}$ is commonly adopted to describe the process of granulation (e.g. Barros et al. (2020)), which we also use in this work. We can then write: [TODO: formula rewriting], which leaves us with ρ and σ_{SHO} to specify. A mixture of two SHO terms can be used to account for stellar rotation, which we use as second term of our kernel. This term is defined by five parameters: σ_{rot} indicating the standard deviation of the process; the period of rotation; Q_0 which represents the quality factor for the secondary oscillation; dQ for the difference between quality factors of the two modes; and f indicating the fractional amplitude difference between the two modes. We will loosely refer to σ_{rot} as the amplitude of stellar rotation.

3.1.1.3 Photon noise

To simulate photon noise, we simply sample from a Gaussian distribution $\epsilon_i \sim \mathcal{N}(0, \sigma)$ and add ϵ_i to the corresponding flux at time step i. The standard deviation σ defines the level of time independent noise in the light curve, and is specified for each light curve separately.

3.1.1.4 Transit signals

The most important ingredient of our data is the transit signal. Fortunately, several tools exist to simulate signals from transiting exoplanets. One of these is "batman" (Kreidberg, 2015), which is based on the equations formulated by Mandel and Agol (2002). Given a set of planetary and stellar parameters, batman computes the transit light curve, which is equal to the fraction of observed starlight computed at each time step during in- and out-of-transit. This light curve is a idealized light curve, assuming no noise and no stellar activity. We therefore only need to multiply this with our noisy background to get the desired light curve with transit signals.

The parameters used for the simulation of transit light curves are: epoch t_0, or the transit time of the first transit in the light curve; the orbital period P of the planet; the planet's radius relative to its host star (radius-over-radius or *ror*, e.g. for Earth, $ror = 0.01$); orbital parameters, such as semi-major axis a, inclination i and eccentricity *ecc*; and limb darkening parameters of the host-star. We approximate the limb darkening of stars with a quadratic function, parametrized by two coeffients u_1 and u_2 [TODO: give function]. We constrained the values of P and a using Kepler's Third Law:

$$P^2 = \frac{4\pi^2}{GM} a^3. \tag{3.1}$$

In this formula, G is the gravitational constant ($= 6.67408 \cdot 10^{-11} \text{m}^3\text{kg}^{-1}\text{s}^{-2}$), and M is the star's mass. We thus get the extra parameter M that we need to specify. Furthermore, a is expressed in terms of stellar radii R, which in turn needs to be specified in order to compute Equation 3.1.

The orbital inclination is assumed to be 90° for all planets we simulate, meaning that each planet moves across the center of the stellar surface in our line of sight while it orbits. In reality, the inclination can take any value between 0° and 180°. However, as described in 2.2.2, only a few values result in a transit signal, which also makes the evaluation of detections more difficult. To avoid these problems, we therefore simply set $i = 90°$.

3.1.1.5 Multiple planets

A single star can have multiple orbiting planets. LCSim allows for the simulation of arbitrarily many orbiting planets, albeit ignores whether a certain configuration of planets is realistic or not. In case a light curve is required to have transit signals from two different planets, the transit simulator is simply called twice 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 is unrealistic but does not pose a problem for the aims 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.

3.1.1.6 Parameter sampling

[COMMENT: this is still very vague, and still needs to be written fully] For each light curve, we need to specify 8 parameters to define the background variability. For each transiting planet we need to specify 3 parameters, with an additional 4 for the first planet to define necessary relations with the host star. The distributions used to sample each parameter are given in Table [TODO: table]. These distributions are based, partly on previous literature (see table footnotes), and partly on a subset of TESS light curves for which parameters such as stellar amplitude, white noise, were estimated and parameters such as radius were given. We took inspiration from the set of TOIs (TESS Objects of Interest) and confirmed exoplanets to define parameter ranges for the transiting exoplanets.

3.1.2 Lilith-4

[COMMENT: I will also need to revise this part still] [TODO: explain somewhere how the lilith data is separated into a NN train/valid/test dataset and a transit detection evaluation se.] In addition to the LCSim light curves, we use light curves generated by the Lilith simulator of TESS' pipeline to evaluate and compare our method to other detection algorithms. This data brings us close to real-world data, but still comes with a ground-truth. 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 (Osborn et al., 2020). We use this data for several reasons. First, it functions as a test for both our algorithm and the LCSim simulator. Unexpected results would lead us to believe that either our algorithm is too dependent on the data that is used, or that LCSim data is too unrealistic. Second, it allows the evaluation of 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 telescope within the so-called centroid data, which can be used by our algorithm to potentially benefit from.

3.2 Network

The basis network architecture, training objective and procedure are described in the following sections. The main function of the network is to classify individual data points as signal or non-signal, where every data point that falls 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 a separate section, as these require slightly different network architectures and training. All of the following was implemented using PyTorch (version 1.8.1).

[TODO: describe that for only flux inputs, RNN assumes uniform time intervals]

3.2.1 Architecture

The RNN forms the basis of the detection algorithm. Though several options for the recurrent part of the network exist, each has the properties of allowing arbitrary input sizes and providing an output for each time step of the input series. As light curves may contain many data points, we avoid the "vanilla" RNN, which is known to suffer from vanishing gradients. For this work, the two recurrent cells considered were LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit). [TODO: formulas]

As both directions in time are relevant for the classification of a data point, we make use of a bidirectional RNN (bi-RNN). For example, knowing that a data point is both preceded by a signal that could indicate a transit ingress and followed by one that could indicate a transit egress, would make the classification of that data point easier as compared to only knowing the past. The hidden representations of the bi-RNN components at each time step are concatenated and passed to the next layer.

In order to classify each time step as signal or non-signal, the output of the network should be a single number at each time step. To achieve this, we apply fully connected layers to the output of the RNN, which project the hidden representation at each time step down to a single node. The result is passed through the sigmoid activation function. The resulting value, ranging from 0 to 1, should not be confused with probability, but does indicate the extent to which the network classified the data point as non-signal or signal respectively. Between fully connected layers, the ReLU activation function is applied to introduce non-linearities.

3.2.2 Training

A supervised learning approach is adopted, which relies on labeled training data. This requires knowing which data points are part of a transit signal and which are not. In our case the simulator provides a ground-truth. In the case of using real-world data, one might label known exoplanet transits as signal and everything else as non-signal, although one may never be sure of the non-existence of transit signals. Another option is to use simulated data for training and real-world data for inference. Lastly, one could inject simulated transit signals in real-world data, relying on the fact that transit signals are rare and ignoring a few real transit signals would likely not harm the performance of the network considerably.

3.2.2.1 Objective

For each data point i in a light curve, the target class t_i is 0 (non-signal) or 1 (signal). The network outputs a value y_i between 0 and 1 at each time step. For a light curve j of length N, the loss is given by the mean binary cross entropy loss over all time steps:

$$\mathcal{L}_j = \mathcal{L}_{\text{BCE},j} = \frac{1}{N} \sum_{i=0}^{N-1} w_i \cdot p'_t \cdot t_i \log y_i + (1 - t_i) \log(1 - y_i), \tag{3.2}$$

where p'_t and w_i are weighting factors. Our task deals with imbalanced data sets, as the number of transit data points is always less than the non-transit data points. Therefore, in some cases it is preferred to increase the weight of transit data points for higher recall. Sometime, it is even necessary to encourage the network to start learning in general, instead of classifying all data points as non-transit. To do so, a parameter p_t can be used to apply more ($p_t > 1$) weight to transit data points. Increasing p_t will increase the recall, but will also lower the precision. An alternative form of weighting is to use transit-specific weighting. Some transit signals may be shallower than others, and thus should get a lower weight during training. Since we assume the transits in the training data are known, we can include their relative depths in the loss function through w_i. In case both p_t and w_i are used, a small adjustment to p_t needs to made for maintaining consistent results. This is to ensure that for the same value of p_t in multiple runs of training, we may expect approximately the same values for precision and recall, regardless of the values for w_i. The new weight for transit data points during training becomes $p'_t = p_t \cdot \sum_i t_i / \sum_i w_i$, where for $w_i = t_i$ we have $p'_t = p_t$.

The network is trained using mini-batches consisting of B input light curves, so the combined loss per batch becomes:

$$\mathcal{L} = \frac{1}{B} \sum_{j=0}^{B-1} \mathcal{L}_j = \frac{1}{BN} \sum_{j=0}^{B-1} \sum_{i=0}^{N-1} w_i \cdot p'_t \cdot t_i \log y_i + (1 - t_i) \log(1 - y_i), \tag{3.3}$$

where we assume that each light curve in the batch is of the same length N. [TODO: fix indexing, now t_i is the same for each light curve j. Instead use T_{ij} or similar]. [TODO: include weight decay penalty]

We used the Adam optimizer to minimize this loss during training (Kingma and Ba, 2014).

3.2.2.2 Light curve segments

Typical light curves, e.g. from TESS, are too large to be used for training the RNN. Not only does the RNN iterate through the light curve recursively, the training also requires storing the gradients over each step through the light curve. In addition, the RNN cannot well learn dependencies over scales spanning thousands of data points. For these reasons, using full-length light curves would make training unnecessarily inefficient.

For training the RNN, we therefore use light curve segments, which can be obtained by splitting the original light curve in pieces, or by directly simulating smaller sized light curves. The size of these segments should be larger than the typical transit duration (~6 hours), but small enough to allow for efficient training. In this work we adopt segment sizes of $N = 1500$ spanning 50 hours in time, unless stated otherwise.

The RNN is forced for this reason, to only learn local relations, e.g. features belonging to individual transits. For detecting individual transit events, periodicities in the signals are ignored (unless the period is extremely small). If the RNN subsequently applied to a full-length light curve, the detection results of individual transit events may be further inspected for periodicities.

3.2.3 Extensions

As this is the first work to use RNNs for transit detection, we opted to use a fairly basic network architecture as basis of the detection algorithm. Nevertheless, the true power of neural networks is often discovered through creative design choices. In the following, several extensions to the basis architecture are described that are explored in this work for our task.

3.2.3.1 Generative network

Missing data points in light curves could pose a problem to our RNN-based detection algorithm. Naively passing the available flux values through the network, would result in the network ignoring the varying time intervals between measurements. A way to counter this problem is to feed time differences between measurements in addition to the available flux values. However, this extra stream of input data could increase the risk of overfitting. Other simple approaches are to replace all missing values with zeros, or to fill gaps by linear interpolation.

Alternatively, one could let the RNN fill the gaps by itself, as a side-task to detecting transit signals. Say, an input light curve contains flux values x_i. For some values of i, $x_i = \text{NaN}$, indicating that these values are missing. In addition to the output y_i, which is used classify each data point as signal or non-signal, we let the network output an additional value $\hat{x}_i$. This value represents the predicted flux value at time step i, based on all values at time steps $j \neq i$ [TODO: fix as this is not entirely true. Each component of the bi-RNN predicts the next value at i, based on time steps $j < i$ and only the hiddens of both directions are contatenated]. Each time a NaN value is encountered, $\hat{x}_i$ is used as input to the RNN instead of x_i. $\hat{x}_i$ is obtained by applying a separate network of fully connected layers to the outputs of the recurrent cells. In this case the sigmoid function is not applied to the final result, as the flux values are free to take any value (they may even be negative after preprocessing).

The architecture is thus extended with a few fully connected layers to obtain predictions for missing values. In order for the RNN to learn to predict these value better, we extend our loss function with a term computing the mean squared error (MSE) between the true values x_i and the predicted $\hat{x}_i$:

$$\mathcal{L}_{\text{MSE},j} = \frac{1}{N} \sum_{i=0}^{N-1} (x_i - \hat{x}_i)^2, \tag{3.4}$$

which we evaluate for all i for which $x_i \neq \text{NaN}$. The combined loss for an input light curve j then becomes

$$\mathcal{L}_j = \mathcal{L}_{\text{BCE},j} + \lambda \mathcal{L}_{\text{MSE},j}, \tag{3.5}$$

where λ is used to weigh the MSE term.

3.2.3.2 Representation network

Another direction to explore is the way potential transit signals are represented within the network. It is beneficial for the network to represent transit signals different from non-transit background pattern, as this would help in the classification of data points. In the case of multiple transit signals however, the network is not encouraged to distinguish between differences in transit shapes, since all data points belonging to these signals are only labeled as signal.

However, a single light curve might contain transit signals from multiple different exoplanets. In fact, with many parameters at play that define the transit signal shape (see Section 2.2.2), different transiting exoplanets will generally have differently shaped transit signals. Inspecting the detected transit events in such a light curve for periodicities might in this case lead to ambiguities, as each detection is treated the same.

In an attempt to resolve such ambiguities, we could encourage the network to represent differently shaped transits differently using contrastive learning. To achieve this, the basis network is extended with fully connected layers which are applied to the outputs of the recurrent cells. These so-called projection layers project the representations of each time step down to D dimensions into a vector R_i for time step i. [TODO: check and cite contrastive learning paper]. For different detected candidate transit signals, say, A and B, covering time steps $\{i_a, \ldots, j_a\}$ and $\{i_b, \ldots, j_b\}$ respectively, the network outputs representations $\{R_{i_a}, \ldots, R_{j_a}\}$ and $\{R_{i_b}, \ldots, R_{j_b}\}$. 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:

$$\mathrm{sim}(\mathrm{A},\mathrm{B}) = \frac{R_\mathrm{A} \cdot R_\mathrm{B}}{\|R_\mathrm{A}\|\|R_\mathrm{B}\|} = \cos(\theta_\mathrm{A,B}). \tag{3.6}$$

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

During training, we present the network with pairs of light curves with the same stellar background, and pre-select a region A from the first light curve and a region 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

$$\mathcal{L}_{\mathrm{repr},j} = (\mathrm{sim}(\mathrm{A},\mathrm{B}) \cdot (1 - 2 \cdot p_\mathrm{A,B}) + 1)/2. \tag{3.7}$$

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 a light curve pair j is then

$$\mathcal{L}_j = \mathcal{L}_{\mathrm{BCE},j} + \lambda \mathcal{L}_{\mathrm{repr},j}, \tag{3.8}$$

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

3.2.3.3 Confidence network

Lastly, we explore a method for estimating confidence over the network predictions. As mentioned before, the output y_i of the RNN at time step i, although conveniently bounded by 0 and 1, should not be interpreted as a probability. After all, it could still occur that the the network is presented with data that was rarely seen during training, if at all. In that case the network might output a large value for y_i, leaving us to believe it is highly confident of its prediction. Although it is a difficult task to obtain statistically meaningful confidence estimations over network outputs, there are ways in which we can get an indication of confidence of certain predictions in relation to others.

To this end, we make use of a simple approach proposed by DeVries and Taylor (2018). Only two things need to be changed in the basic setup of training. First, similar to the extensions described in sections 3.2.3.1 and 3.2.3.2, we apply a separate network of fully connected layers to the outputs of the recurrent layer. The fully connected layers project the outputs of the recurrent layer down to a single node, which is than passed through the sigmoid activation function. The result, which we call c_i, is the confidence parameter given for the prediction y_i at time step i, and is bounded by 0 and 1. The second change that needs to be made is in the BCE term of the loss function. The BCE loss for an input light curve j becomes

$$\mathcal{L}'_{\mathrm{BCE},j} = \frac{1}{N} \sum_{i=0}^{N-1} w_i \cdot p'_t \cdot t_i \log y'_i + (1 - t_i) \log(1 - y'_i), \tag{3.9}$$

where the adjusted prediction y'_i is given by $y'_i = c_i y_i + (1 - c_i t_i)$. Intuitively, if the confidence c_i over the prediction y_i is high then $y'_i \approx y_i$, and if c_i is low then $y'_i \approx t_i$, where t_i is the target. The network can thus "ask for hints" if, for example, the input data is ambiguous.

To prevent the network from always returning low confidence values, an additional term

$$\mathcal{L}_{c,j} = \frac{1}{N} \sum_{i=0}^{N-1} -\log(c_i), \tag{3.10}$$

is included in the loss, which penalizes low values for c_i. The combined loss becomes

$$\mathcal{L}_j = \mathcal{L}'_{\mathrm{BCE},j} + \lambda \mathcal{L}_{c,j} \tag{3.11}$$

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 $\mathcal{L}_{c,j} > \beta$, then we increase λ, and if $\mathcal{L}_{c,j} < \beta$, then we decrease λ. Lastly, only 50% of the times we use $\mathcal{L}'_{\mathrm{BCE},j}$ and the other times we use $\mathcal{L}_{\mathrm{BCE},j}$ from Equation 3.2, in order to encourage the network to still learn meaningful decision boundaries.

3.3 Detection algorithm

[TODO: highlight that the RNN alone is ready to be used for monotransit detection and is very efficient]. The RNN can handle arbitrary input sizes, but will only learn local features of a light curve during training, i.e. which data point belongs to a transit signal and which does not. Parameters such as orbital period P of a planet, are ignored by the network. If the network is applied to a full-length light curve after training, it might run into multiple transit events from the same planet. In the task of detecting potential exoplanets, it is important to determine the periodicity of the signal if possible, as this in combination with the epoch t_0 allows follow-up research to predict when the next transit events will occur. Furthermore, restricting our search to transit signals that repeat a certain number of times during the span of observations would help in reducing the number of false positive detections.

Following Pearson et al. (2018), we refer to all the network outputs y_i over time steps i in a light curve as the probability time series (PTS), while keeping in mind that the name "probability" here is not in place correctly. The task of our transit detection algorithm is to determine the epoch and period (t_0, P) of candidate exoplanets, given the PTS produced by the RNN. Since the PTS only contains the response of the RNN to potential transit signals, each peak in this series could indicate a transit event. To find periodicities, we adopt two different approaches, which are described in the following subsections.

3.3.1 Evaluating peak distances in RNN outputs

[TODO: refer to this as PTS-Peak] As proposed by Pearson et al. (2018), one approach to obtain periodicities of potential signals is to evaluate the distances between peaks in the PTS. For this, we use a Gaussian-smoothed PTS, so that peaks are not accidentally counted twice due to noise. We define a peak as a series of values above a certain peak threshold. After having identified the peaks in the PTS, the central time of each event is determined by dividing the peak's area under the curve in two, and taking the time step at the border between the two halves. One could also simply use the time step at the center of the peak, but this was found to produce less accurate results.

Subsequently, each combination of peaks is inspected for consistent timing. For example, a combination of five peaks could be evaluated and with an average distance of 4 days in the PTS. If the individual distances between subsequent peaks differ for some peaks more than a few hours, the combination of peaks is considered to be inconsistent and therefore rejected. If a combination is not rejected, it is added to a list of candidates.

The list of candidates is then iterated through in order to find the best candidate. For each candidate signal, the estimated transit duration is determined by the median duration of each peak, the estimated period by the median distance between each peak, and t_0 by the time of the first peak. For each candidate, the score is defined by sum over all of the peak's maxima, weighted down by the number of peaks that were found. This weighting down is to avoid the algorithm to have a bias towards detecting mainly small period candidates over longer period candidates. If n is the number of peaks under consideration and s the sum over all the peak's maxima, then we define the candidate's score to be given by $s/\sqrt{n}$.

The highest scoring candidate is further analysed, mainly to assess whether single events were missed by the algorithm or the RNN. This last step is added because a few peaks in the PTS might have just not made it above the peak threshold, resulting in the rejection of nearly complete combinations of peaks. The process goes as follows: we iterate over harmonics $h = 1, 2, \ldots$, for as long as $P/h > P_{\text{min}}$, where P is the estimated period of the candidate planet and P_{min} the minimum period to search for. For each h we effectively evaluate a different number n_h of potential transit events, and s now becomes the sum s_h over each maximum value within the estimated transit duration of the candidate signal for the given period P/h. The score for harmonic h then becomes $s_h/\sqrt{n_h}$. If $s_h/\sqrt{n_h} > s/\sqrt{n}$, then the candidate's estimated period is adjusted to P/h and the epoch t_0 is adjusted accordingly.

[TODO: describe how multiple planets in a single light curve are searched for more or less simultaneously] [TODO: describe how matching of signal representations is included in the algorithm, which makes use of the representation RNN]

3.3.2 Folding RNN outputs over trial periods

[TODO: refer to this as PTS-Fold] Although the algorithm described in Section 3.3.1 has its benefits, it also has its downsides. The benefit of the approach is that we let the RNN guide us where to search for periodicities. This makes it a very efficient method. However, if the RNN fails to detect one or two transit events in a light curve, or has a response that is not large enough to be picked up by this algorithm, then the algorithm might fail. For this reason, we adopt a more classical approach in addition to the algorithm described in the previous section.

The approach we take in this algorithm is that of folding the PTS over many trial periods, each time defining a candidate score at each point in the resulting phase curve. The trial periods are chosen such that the data

points in the folded curve exactly coincide. In other words, the trial periods are exactly multiples of two minutes in case the input light curve is also of two minute cadence. This allows us to directly sum the PTS values of each set of points coinciding in the phase curve. This sum has the same meaning as s in the previous section, and the number n of transit events under consideration is in this case given by the number of data points that we take the sum of. For each trial period, we thus get a set of different scores $s/\sqrt{n}$, corresponding to different epochs t_0. For each trial period we store the maximum score and the corresponding epoch t_0.

We then get a spectrum of scores against different trial periods. This spectrum is evaluated for peaks, where the highest peak in the spectrum corresponds to the highest scoring planet candidate with corresponding period P and epoch t_0.

[TODO: describe how multiple planets in a single light curve are searched for iteratively]

Chapter 4

Experiments and results

In order to better understand and evaluate the individual components of our algorithm, the first sections of this chapter are devoted to take a closer look at each one of them. We compare the RNN with the more commonly used CNN in a binary classification task for input light curve segments, and illustrate how the RNN naturally extends to full-length light curves. In the same section, Section 4.1, we also motivate the network architecture that was used throughout the rest of the experiments. Since preprocessing can be of influence on the performance of the network, in Section 4.2 we compare the effect of several different preprocessing steps on training of the network. For example, several approaches to dealing with gaps are evaluated, as well as the use of a generative network which is trained to fill gaps by itself. Section 4.3 illustrates the workings of both detection algorithms described in Section 3.3, where success and failure cases are identified. In addition to these algorithms, we illustrate how the network's representations of each time step can be used to potentially resolve ambiguities in the matching of individual signals when searching for periodicities.

In the last three sections, we evaluate our method as it would be used in real-world applications. First, we show that the RNN has a large potential in the task of monotransits detection. Subsequently, we compare our RNN-based detection algorithm with the BLS algorithm in the task of detecting single planets in LCSim light curves. Lastly, in the task of detecting arbitrarily many planets in more realistic light curves from Lilith-4, we compare the detections from the TESS pipeline, BLS and our method.

4.1 Detection or identification: RNN as hybrid

Without changes to the model or the training process, the RNN can be used for both the identification task and the detection task. This is illustrated in the following subsections. First, the RNN is compared against an MLP and CNN in the task of classifying light curve segments as signal or non-signal. Then, we show how the RNN trained for this task can also be used for locating signals in time for arbitrarily sized input light curves. Different network architectures are tested, which serves as a motivation for our choices of the network architecture used in subsequent experiments.

4.1.1 Fixed input size, single output

As described in Section 3.2, the RNN is trained to classify each data point of a given light curve as signal or non-signal. Recall that we refer to these outputs as the probability time series (PTS). However, in the task of classifying an entire input light curve segment as signal or non-signal, we only need a single output. Two options to obtain a single RNN output for a given input light curve segment are: (i) use the maximum of the PTS for the binary classification of the whole segment; or (ii) train an RNN specifically for this task, while only using the output at the last time step for the binary classification of the segment. We refer to (ii) as the *naive* approach, since this approach does not utilize the full learning potential of the RNN as it only provides a sparse learning signal.

Two data sets were generated using LCSim, both consisting of 15000 light curve segments for training, 5000 for validation and 5000 for testing. For each split, 50% of the segments contained at least one transit signal and the other 50% contained no signals. One data set contained light curve segments consisting of $N = 500$ data points, and for the other data set $N = 1500$. In other words, the time spans of segments in each data set were 16.7 and 50 hours respectively. Several models were trained using the training sets, their architectures were determined by inspecting their performances on the validation sets, and only here analyse the performances on the test sets. For each model, an informal search over hyperparameters was performed to determine the final architecture, the process of which is further described in the following.

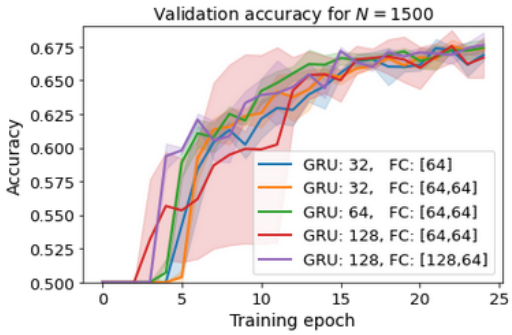

Figure 4.1: [TODO: caption]

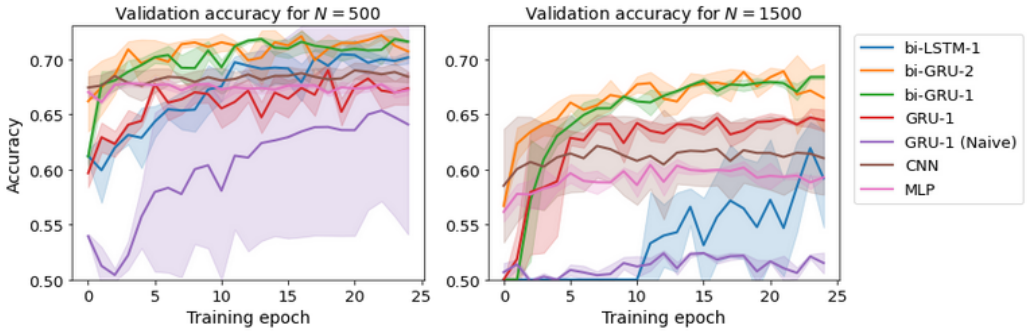

Figure 4.2: [TODO: caption]

The MLP is a simple network consisting of several fully-connected (FC) layers. We experimented with different numbers of layers and nodes per layer, and decided on a network consisting of three hidden layers with 128, 64 and 64 nodes respectively. Several variations to these values were tested, but resulted in similar performance. However, the weight decay parameter seemed to be of importance to the performance of the network, as for low values of this parameter the MLP was found to quickly overfit the training data. We therefore used a weight decay of 5×10^{-3}. The ReLU activation function is applied between layers and the sigmoid is applied to the final output. Each of the described networks in the following use the same activation functions.

The CNN requires the specification of more hyperparameters. In an attempt of keeping the architecture simple, we found several hyperparameter settings to result in similar performance. In the following we use a CNN with three convolutional layers, consisting of 4, 12 and 1 channels respectively, using kernel sizes of 7, 7 and 3 and a stride of 1 at each layer. After each layer, we apply batch normalization and a max-pooling operation over 3 nodes. The final layer of the network is an FC-layer of 48 nodes.

For the RNN, GRU was found to produce better and more stable results than LSTM. In Figure 4.1 we compare several architectures of the bidirectional GRU with a single layer (bi-GRU-1). The hyperparameters that were varied over were the number of hidden nodes in the recurrent cell, the number of FC-layers that follow the recurrent layer, and the number of nodes per FC-layer. For each set of parameters, the network's performance on this task was similar. For each RNN tested in following experiments, we use recurrent cells with 64 hidden nodes, and two FC-layers of 64 nodes.

[TODO: provide table with test results] The classification accuracies of each model evaluated on the validation set during the process of training are plotted in Figure 4.2, from which several things can be noted. First of all, each model seems to perform worse in the case of longer ($N = 1500$) light curve segments. This is probably due to the fact that the longer the segments, the more distracting background patterns may be present that mimic transit signals. The best performing models are GRU-based networks, which were not even trained for this specific task. The RNN that was trained specifically for this task, with a training procedure referred to as *naive*, performed considerably worse. This is in line with our expectation that only providing a single learning signal to the RNN for a given input segment is bad practice. Lastly, we find that using more than one recurrent layer does not necessarily lead to better results. Therefore, in following experiments, we adopt a bidirectional GRU with a single recurrent layer (bi-GRU-1) that was trained on the data set with $N = 1500$, and refer to this network simply as the "RNN".

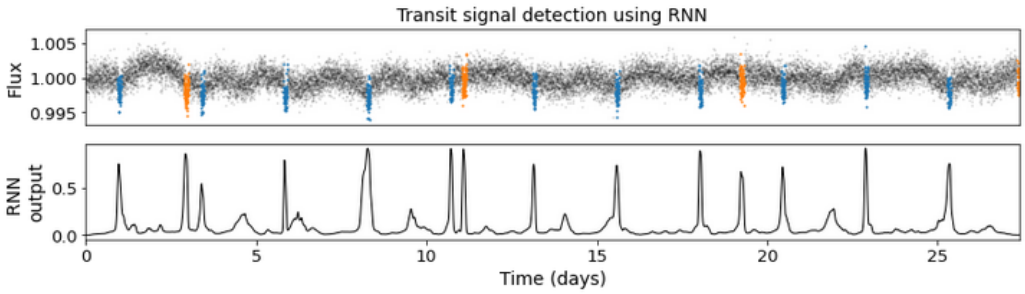

Figure 4.3: [TODO: caption]

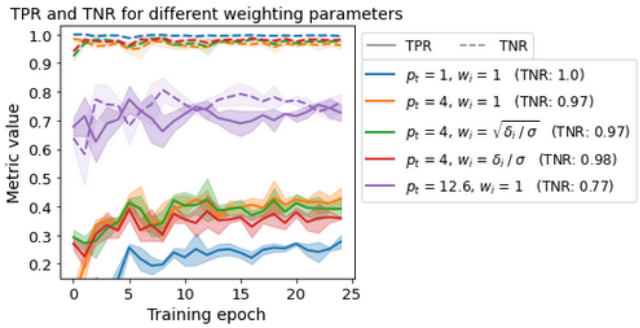

Figure 4.4: [TODO: caption]

4.1.2 Locating signals in time for inputs of arbitrary size

As opposed to the CNN and MLP, the RNN can handle arbitrary input sizes. This is illustrated in Figure 4.3, where the RNN from Section 4.1.1 is applied to a full-length light curve. In this case we show the RNN's output at each individual time step. Each peak in the resulting PTS indicates the presence of a potential transit signal at the corresponding time in the light curve.

However, data in the task of detection is less balanced than in the task of identification. This is because at the level of individual data points, we only have a few points that are labeled as signal, and many points that are labeled as non-signal. The accuracy of classification is therefore a misleading metric, as the network could already achieve high accuracies by classifying each data point as non-signal. Therefore we turn to different metrics for finetuning the network for detection. The true positive rate (TPR), or recall, is the fraction of the correctly classified positives (i.e. points labeled as signal). The true negative rate (TNR) is similar, but for correctly classified negatives instead of positives. Lastly, precision is the fraction of the samples classified as positive that were correct.

If the classes were perfectly balanced, we expect a TPR similar to the TNR. In our LCSim data set with $N = 1500$, we have 12.6 times more non-signal data points than points that are labeled as signal. Using an extra weight in the loss for transit points of $p_t = 12.6$ will therefore balance our data set. Figure 4.4 shows that this indeed the case. However, although tuning the TPR and TNR can be useful for our purposes, a lower TNR also means we falsely classify more samples as positive, thus decreasing the precision. To increase the recall, but avoid a large decrease in TNR we therefore adopt a smaller weight of $p_t = 4$. The other weighting parameter that is varied, w_i, is used to tune how much each individual transit signal is weighted compared to others. For example, we can set $w_i = \delta_i/\sigma$, which will weigh each positive data point by its corresponding transit depth δ_i relative to the time-independent noise σ. In that case, a transit with twice the depth will get double the weight. This type of weighting might, however, cause the network to place too little focus on correctly detecting shallow transits. To reduce this effect, but still apply transit-specific weighting, we can adopt $w_i = \sqrt{\delta_i/\sigma}$.

The effect of different weighting schemes on TPR and TNR are visualized in Figure 4.4. Figures 4.5 and 4.6 show the TPR against different relative transit depths, which was evaluated of test splits both data sets. [TODO: describe how Lilith-4 light curve segments were obtained.] In the following experiments, we adopt $p_t = 4$ and $w_i = \sqrt{\delta_i/\sigma}$, unless stated otherwise.

4.2 Preprocessing and performance

We assume the light curves segments are obtained from a median normalized light curve, e.g. by splitting the original light curve into equally sized parts. Assuming for now that no gaps are present in these segments,

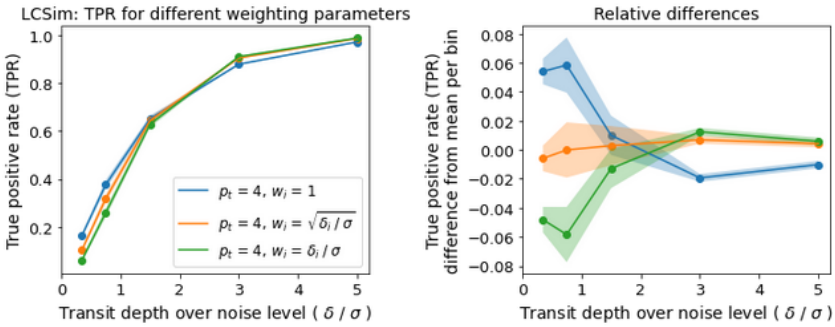

Figure 4.5: [TODO: caption]

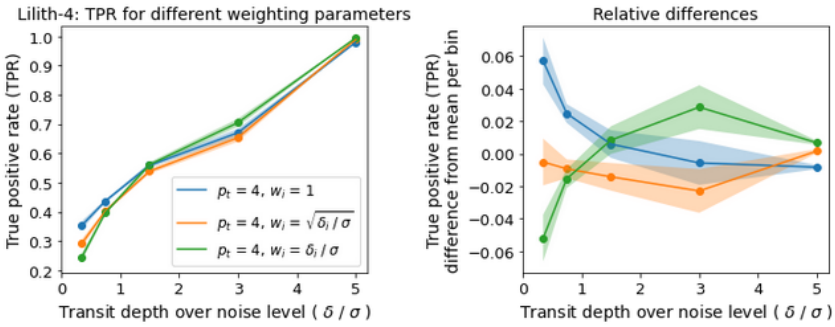

Figure 4.6: [TODO: caption]

several additional preprocessing steps can be considered. One approach is to center the segments around zero by subtracting 1 from the flux values, and doing nothing else. Additionally, one could scale 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 refer to the latter step as sigma scaling, and adopt this in following experiments. See figures 4.7 and 4.8 for the effect of these basic preprocessing steps on the performance of the network during training. [TODO: mention that each all data is standardized as last preprocessing step]

In both cases however, the range of flux values might still greatly vary between two separate light curve segments. 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. This potential problem is visualized in Figure 4.9.

We consider two approaches to resolve this issue: (i) feed the differences between flux values to the network, instead of their absolute values, and (ii) remove large-scale patterns from the original light curve, a process we will refer to as low-risk detrending (LRD). Both approaches have the desired effect of making the input ranges more consistent, with some exceptions. The benefits of (i) are that this approach can be applied to the segments directly and we lose almost no information by doing so. However, although all the information is still there, the structure of the data is completely changed which might be difficult for the network to learn. For (ii), we need to alter the original light curve prior to splitting it into segments. By doing so we lose information, but the result is more intuitive than (i).

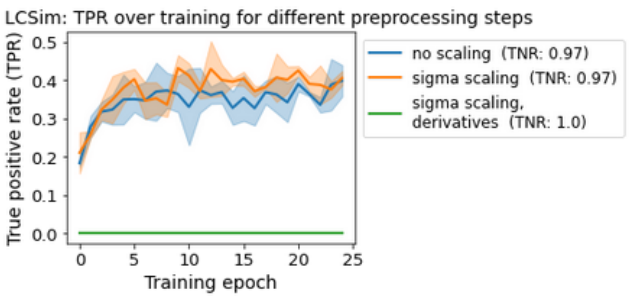

Figure 4.7: [TODO: caption]

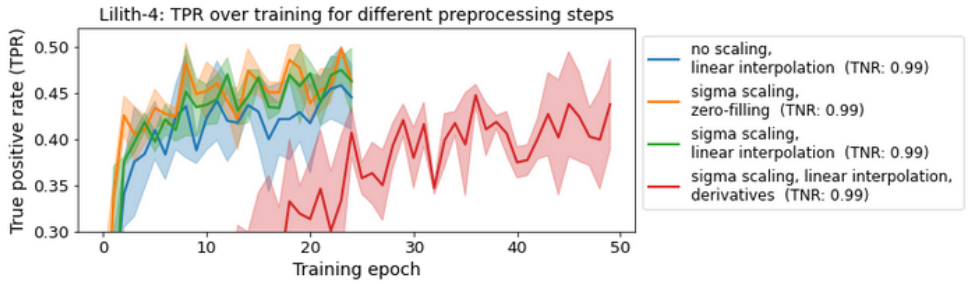

Figure 4.8: [TODO: caption]

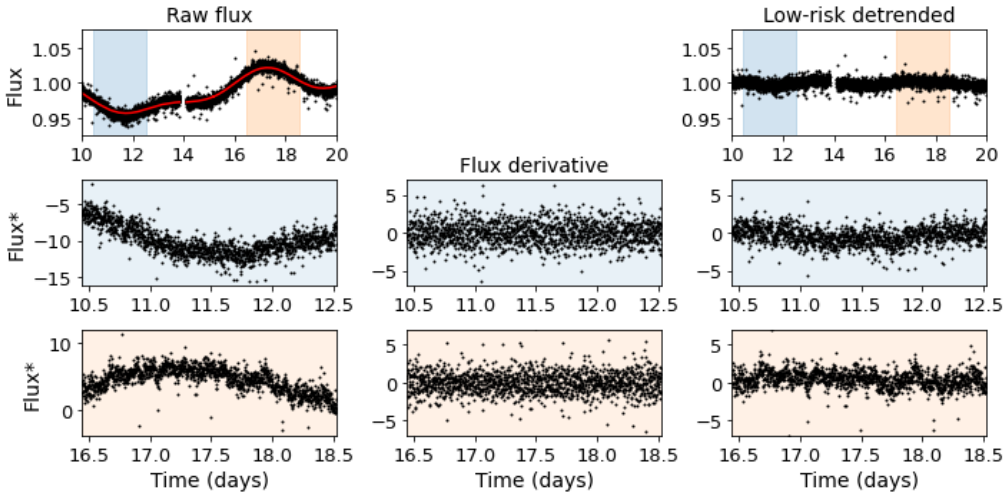

Figure 4.9: [TODO: caption]

In Figure 4.10 we compare the effect of several different preprocessing steps on the performance of the network trained on Lilith-4 data. In addition to the "basic preprocessing" and the "low-risk detrending" as described above, several other approaches are included in the comparison. For example, "high-risk detrending" means the input light curves are detrended such that both large and short-scale patterns are removed. This is referred to as high-risk as it comes with more risk of partly removing transit signals. The detrending filter we use for this is a time-windowed median filter with a window of 12 hours. Since real-world data may include outliers, we also evaluate the effect of removing these from the data. Outliers are identified after the light curve is detrended with a time-windowed median filter with a window of 30 minutes. Positive outliers at more than 6 standard deviations from the median and negative outliers at more than 12 standard deviations from the median are removed from the original light curve. The detrending step and the low negative-outlier threshold are to avoid transit signals from being flagged s outlier. Lastly, since Lilith-4 provides centroid data, which we also have access to for real-world data, we test the effect of using these as additional input to the network. This means that the network input at each time step consists of three values: the flux, and two values that specify the pointing of the telescope at that time step. Before use, we center the centroid data around zero, and scale them similarly to their corresponding light curve. Subsequently all the centroids in the data set are standardized independently from the flux values. This way of preprocessing is to ensure that the relative scale between light curves and their corresponding centroid data is not altered, as this might contain relevant info for the network. The series of centroid data are split into segments together with their corresponding light curve.

The results show that outlier removal had a negative influence on the performance of the network. This was unexpected, because outliers are not expected to carry relevant information. After all, an outlier should be present regardless of whether a transit event is occurring, unless the transit itself is flagged as outlier. This is unlikely to be the case, as we observe worse performance for both deep and shallow transits. We expect therefore that the reason for this result lies in the fact that outlier removal creates missing data. Missing values were imputed for this experiment, which might have caused the network to perform worse in this case. All other preprocessing steps seem to produce similar results. This is noteworthy, as the network therefore does not have to rely high-risk detrended data, as opposed to other common detection methods.

Lastly, since Figure 4.10 does not clearly indicate whether or not using centroid data is beneficial, we take into consideration the results obtained on the validation set created from Lilith-4. These results, shown in Figure 4.11, do indicate a potential benefit of including the centroids as input to the network. This motivated

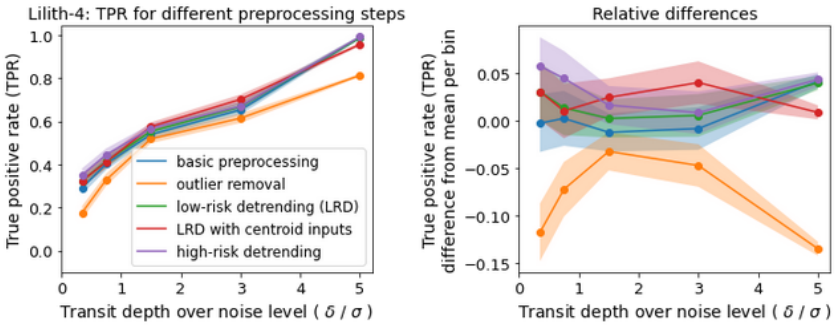

Figure 4.10: [TODO: caption]

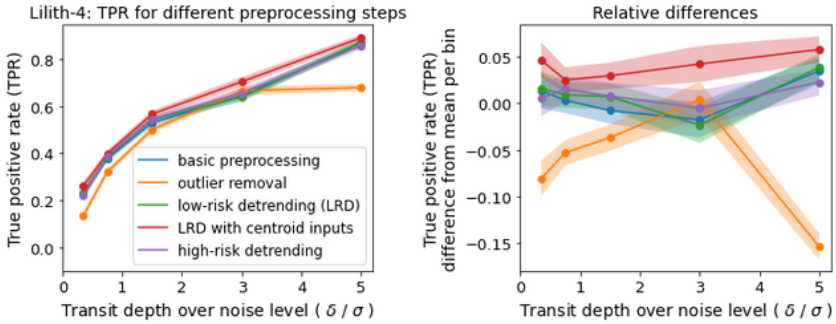

Figure 4.11: [TODO: caption]

us to adopt LRD as standard preprocessing procedure in combination with using centroid inputs when applying our network to Lilith-4 data in following experiments.

Lastly, we address the problem of missing data in greater depth. Lilith-4 data already contains data gaps, which we filled by linear interpolation up to this point, unless stated otherwise. For LCSim light curves, we manually inject NaN values in the pre-generated light curve segments to simulate data gaps. For each data split, 35% of the samples was injected with one large data gap ranging from 2 to 10 hours, and 15% was injected with two large data gaps. Apart from the larger gaps, each data point was replaced by a NaN value with 2% probability.

Three different approaches to handling gaps are evaluated. In each of the approaches, we impute the data in a different way. First, we consider replacing all missing values with zeros in the centered light curve segment. This is equivalent to replacing missing values with ones in the non-centered light curve. In the second approach, we fill the gaps by linear interpolation. When doing this, we linearly interpolate the trend of the light curve, rather than the raw flux values at the edges of a gap. The reason why we do not use the raw flux instead, is that an extreme value or outlier next to a gap could result in an apparent jump in the flux over consecutive points if it is used for interpolation. Lastly, we use the generative network as described in Section 3.2.3.1 to fill in gaps as it steps through the light curve. An example of each of the approaches is visualized in Figure 4.12.

The generative network seems to work as intended by inspecting Figure 4.12. We can take a closer look at what this network is doing in Figure 4.13. The network is bidirectional, so we get two streams of predictions. We can see that these can complement each other. If we only consider the predictions from left-to-right in the first gap and from right-to-left in the second, then the missing values are well reconstructed. However, since each unidirectional component of the network has no information of what comes after a gap, we see the predictions drift off to unrealistic values.

After evaluation on the gap-injected test set, of which the results are presented in Figure 4.14, we find that

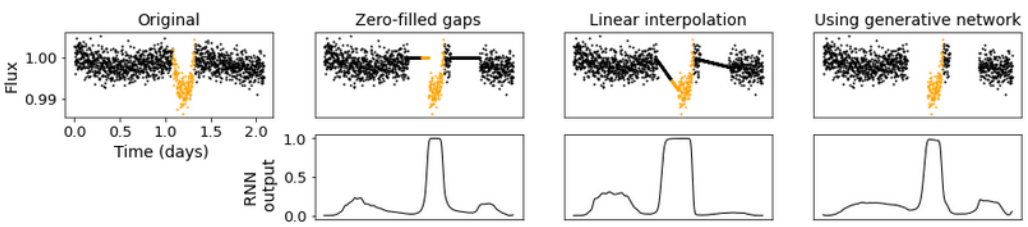

Figure 4.12: [TODO: caption]

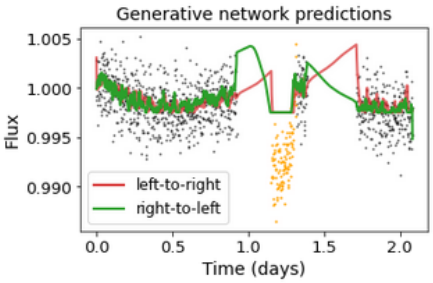

Figure 4.13: [TODO: caption]

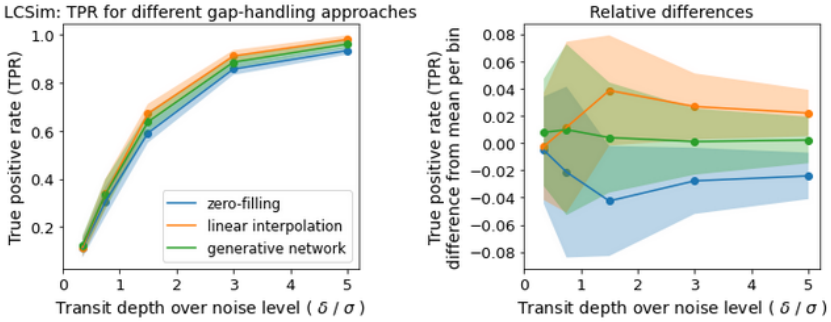

Figure 4.14: [TODO: caption]

each approach to handling gaps leads to similar network performance in terms of TPR, at the same value for TNR. However, on average linear interpolation seems to work slightly better than the other approaches, which we therefore use in following experiments.

4.3 Algorithm comparison: searching for periodic signals

If one were to focus their search on monotransits, no algorithm to determine the period is needed. Any value in the PTS above a certain threshold can be considered as potential detection (see Figure 4.15). In most attempts of finding new planet candidates, the search is guided by the repetition of transit signals. In the following, we take a closer look at the algorithms described in sections 3.3.1 and 3.3.2 and refer to these algorithms as PTS-Peak and PTS-Fold respectively [TODO: give these name in Methodology already].

Figure 4.16 shows an example LCSim light curve with transit signals from a single planet. Both PTS-Fold and PTS-Peak are able to correctly determine the period P and epoch t_0 of the signal, using the PTS produced by the RNN. For PTS-Peak, we have adopted a peak threshold of 0.25.

In order to search for successes and failure cases of both methods, we had LCSim generate 250 light curves of 27.4 days, each with 3 to 5 transits of a single planet. The first and only run of this experiment resulted in 62 failure cases of both methods. For 12 cases only PTS-Fold was correct, and for 1 case only PTS-Peak was correct because PTS-Fold's predicted t_0 was off by 2 hours. One of the cases in which only PTS-Fold was correct is shown in figure 4.17. As can be understood from this figure, the peaks in the PTS corresponding

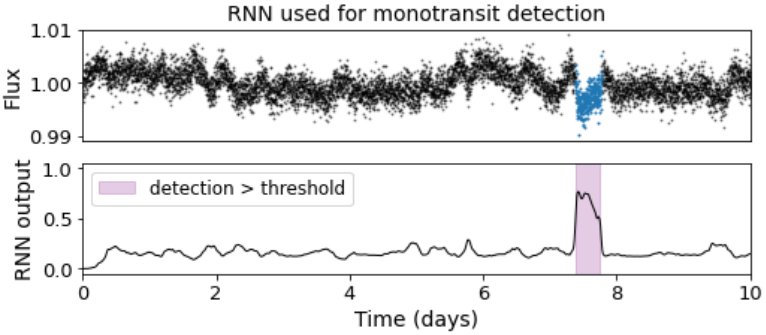

Figure 4.15: [TODO: caption]

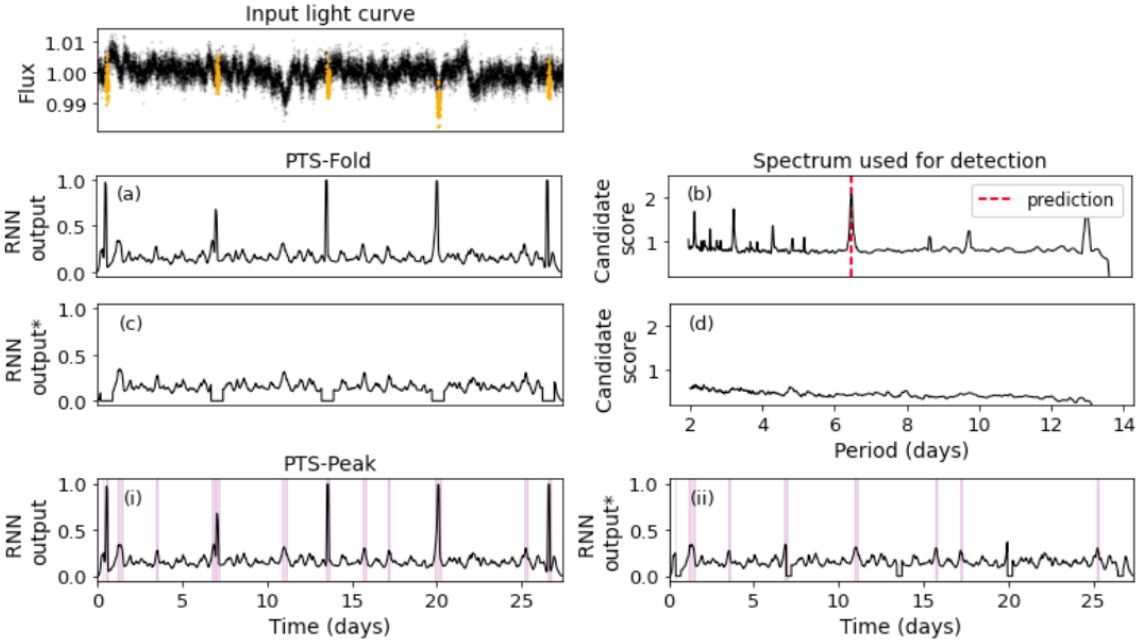

Figure 4.16: [TODO: caption]

to the true signals need to be distinct from the noise for the PTS-Peak to work well. However, sometimes a few transit events could go undetected, false positives might be produced by the RNN, or the peaks are just not strong in general. In those cases, PTS-Peak might match the wrong peaks together, or just prefer smaller periods than the true signal period due to the noise in the PTS.

Nevertheless, in many cases PTS-Peak correctly retrieved the hidden signal. We therefore explore an extension to this method, which has the potential of resolving the issue of mismatching peaks in a given PTS. The network we use for this is the representation network described in Section 3.2.3.2. For the sake of illustration, the network was trained to learn 2-dimensional representations of signals. Recall that the network outputs a representation at each time step, so the representation of a signal is the aggregation over several time steps. The angle between two representation vectors represents their dissimilarity, and since the angle does not depend on the magnitudes of the vectors we can visualize the representations of signals on the unit circle.

From best to worse, figures 4.18, 4.19 and 4.19 illustrate how the use of signal representations could have prevented mismatching of peaks by PTS-Peak that led to incorrect detections. [TODO: Describe what else these figures say, e.g. transit depth seems to be captured, can't say much about shapes specifically. further analysis needed in further work]. [TODO: explain these results were obtained with simple noisy straight lines with injected transits – not certain whether these results will hold under disturbing background patterns].

4.4 Monotransit retrieval

As first real-world application, we consider the task of monotransit detection in a data set consisting of full-length light curves. For this experiment, 5000 LCSim light curves were generated, each of 27.4 days without missing data. 50% of the light curves were injected with a single transit signal, and 50% were left without signals. The task was to retrieve the transit signals by providing the correct transit time t_0, which was allowed to be any time within the duration of the transit.

The baseline method that we use for this task is based on Foreman-Mackey et al. (2016). In their work, a search algorithm similar to BLS is used to search for single transit events in Kepler data. In short, this algorithm fits a box function of given duration at each given point in time. The times at which the function is fitted are defined by the resolution of the search, e.g. one could choose a resolution of 0.25 times the trial duration. At each of these times, the depth of the box is varied until a best fit is found. Then the signal-to-noise ratio is calculated as $\text{SNR} = \delta/\sigma \cdot \sqrt{n}$, where δ is the depth of the signal, σ the noise per time step and n the number of data points the signal covers. The SNR is stored at each point where the function was fitted, so the result is an SNR time series, which is similar to the PTS. We will refer to the SNR time series as SNR-TS in the following. One adjustment that we made to the algorithm of Foreman-Mackey et al. (2016), is that instead of using a single trial duration of 0.6 days, we iterate over durations between 1 and 13 hours, with steps of 1 hour. For all these fits, we take the maximum over SNR's at each time step to define the SNR-TS. Subsequently, we identify peaks in the SNR-TS after standardization, i.e. subtracting its mean and dividing by its standard deviation.

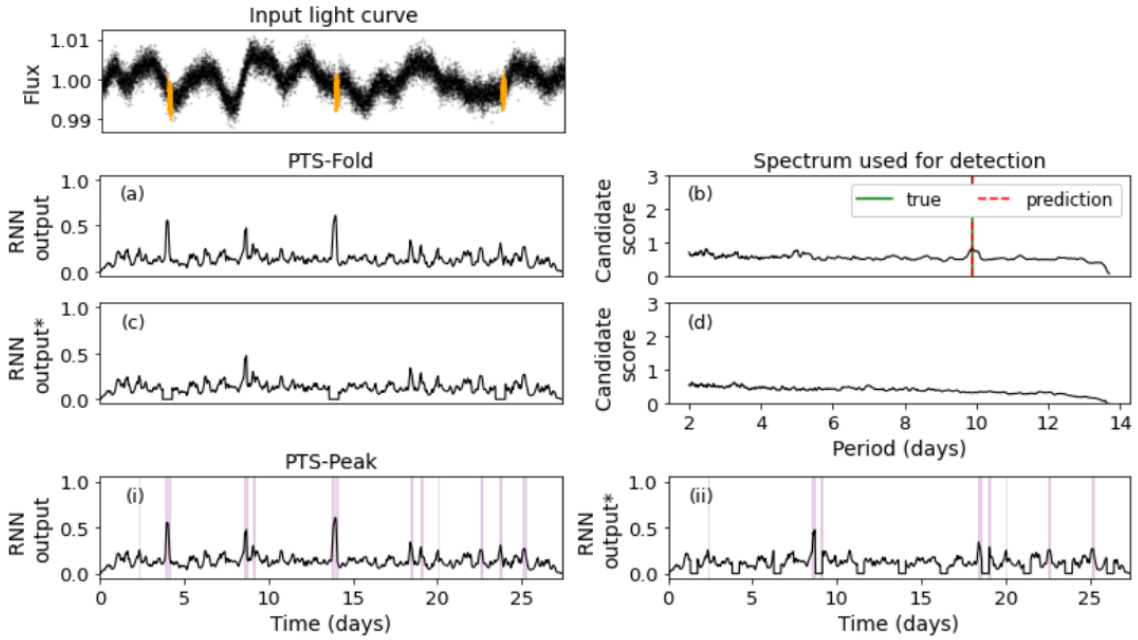

Figure 4.17: [TODO: caption]

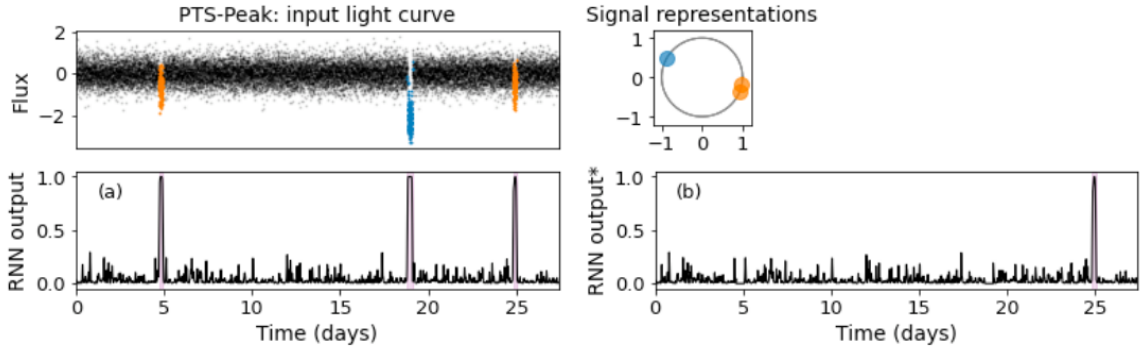

Figure 4.18: [TODO: caption]

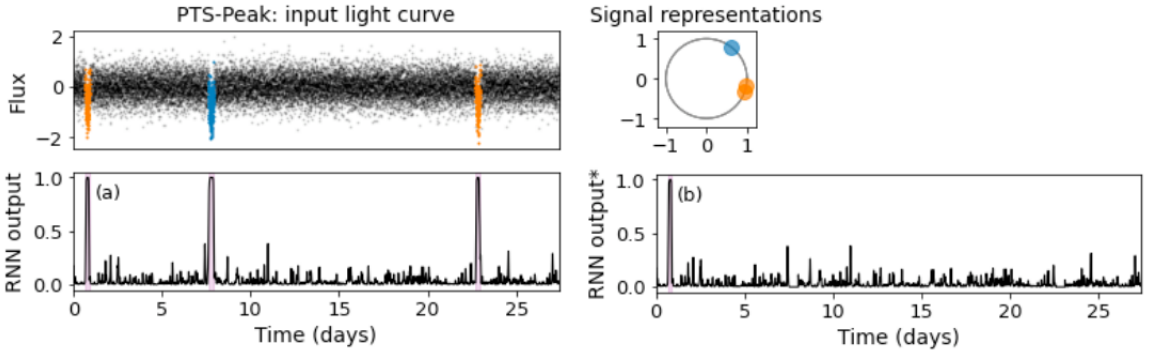

Figure 4.19: [TODO: caption]

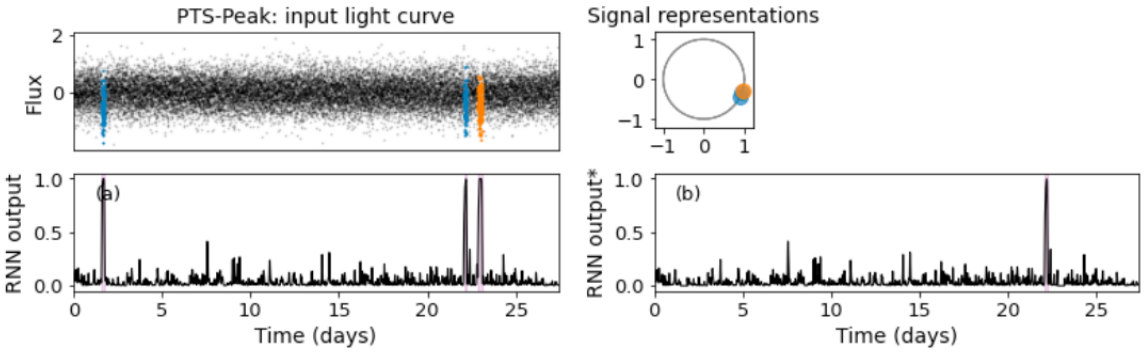

Figure 4.20: [TODO: caption]

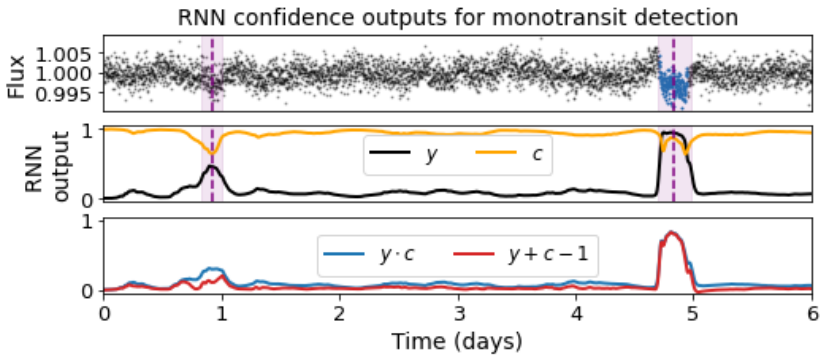

Figure 4.21: [TODO: caption]

This is also different than the approach taken by Foreman-Mackey et al. (2016), as they compute the baseline of the SNR-TS with a running median filter, which is then subtracted from the SNR-TS prior to the search for peaks. However, we found that standardizing the SNR-TS allowed for better determination of detection thresholds. Before applying the algorithm, we detrend input light curves with a time-windowed median filter with a window of 12 hours. Altogether, we refer to this algorithm as Mono-BLS-12h.

Our method is based on the pre-trained RNN from previous sections. As shown in Figure 4.15, a simple approach to detecting monotransit would be to consider each peak in the PTS above a certain threshold as a candidate. After a set of candidates has been collected this way, we give each candidate a score that is equal to their maximum PTS value. This maximum is used as indication of the network's certainty of a candidate relative to other candidates, and is thus used to set a threshold on which candidates are returned as detection.

To extend on this approach, we explore the use of the extra outputs c_i at each time step i that are given by the confidence network described in Section 3.2.3.3. We use these values in two different ways, both with the aim of reducing the precision of the network detections. Figure 4.21 shows an example of the different ways c is used in combination with the standard network output y, to define a new value, say z, which is used to set a detection threshold. [TODO: test what results would be obtained for different peak thresholds with/without conf values, and with/without smoothing]

Figure 4.22 shows the precision-recall curves for each method in the given task. From this figure, we observe that the RNN performed better than Mono-BLS-12h in this experiment. Moreover, no large differences were observed between the different RNN-based approaches. The use of an additional confidence parameter was therefore not of great influence on the results. Nevertheless, the fact that the RNN performed better overall than the box-fitting approach in this experiment, highlights the potential of our approach for this task.

4.5 Single planet retrieval

Similar to the experiment of retrieving monotransits from LCSim light curves, we now evaluate several methods on their ability of retrieving repeating signals from light curves. To do so, 5000 LCSim light curves were generated, each of 27.4 days and without missing data. 50% of the light curves were injected with at least three transit signals from a single planet, and 50% was left without signals. The task was to retrieve the correct epoch t_0 and period P of the transit signal. t_0 was correct if it is anywhere within the duration of the first transit signal, and P if it was off by no more than 1% of the correct period.

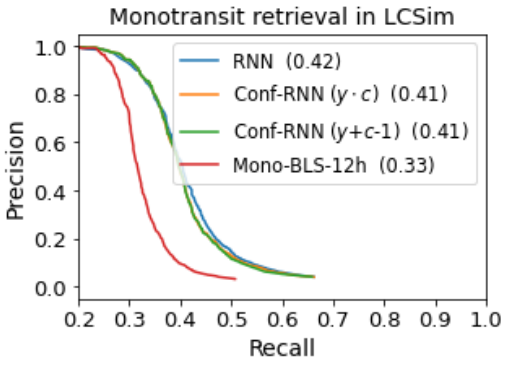

Figure 4.22: [TODO: caption; between brackets shows area under the curve (AUC), also interpreted as average precision] [TODO: explain that we have no uncertainties, and these results therefore only tell something about this specific experiment. future research should confirm these findings.]

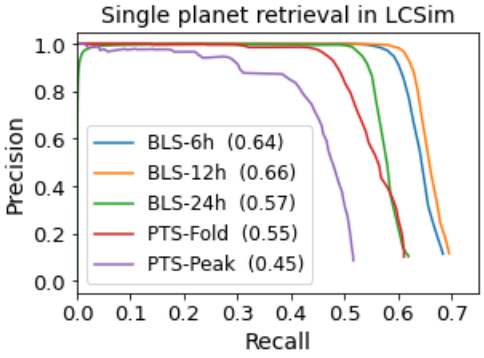

Figure 4.23: [TODO: caption; between brackets shows area under the curve (AUC), also interpreted as average precision] [TODO: explain that we have no uncertainties, and these results therefore only tell something about this specific experiment. future research should confirm these findings.]

Again, we used BLS as baseline for this task, which in this case is the standard algorithm that takes into account periodicities. We used `astropy`'s implementation of BLS[1], with build-in heuristics for the trial periods that are searched over. The trial durations of the fitted box function range from 1 to 13 hours with steps of 30 minutes. Prior to applying BLS, the input light curves were detrended with a time-windowed median filter. Window sizes of 6, 12 and 24 hours were used, which are referred to with BLS-6h, BLS-12h and BLS-24h respectively.

We evaluate both PTS-Peak and PTS-Fold in this task, and set the peak threshold for PTS-Peak to 0.25. For each of the methods, the search is limited to signals with at least three repetitions and a minimum period of 2 days, because this was the minimum period used in the simulations. Moreover, each method was only allowed to return a maximum of three candidate detections per input light curve, each accompanied by a value that was used to set a detection threshold.

Figure 4.23 shows the precision-recall curves for each method in the given task. The curves drop down steeper than those in Figure 4.22, which is likely because in this case periodicities are taken into account, which reduce the chance of false detections. In this case however, BLS performed better than the RNN-based detection methods. This was for each of the chosen window sizes for detrending, although a window of 12 hours showed the best results, which is in line with the results of Hippke et al. (2019). Lastly, the PTS-Fold algorithm performed better than PTS-Peak, as expected from Section 4.3.

4.6 Multiplanet retrieval

Finally, we turn from LCSim light curves to more realistic Lilith-4 light curves, which contain gaps and in some cases transit signals from multiple planets per light curve. The task in this experiment was to provide correct t_0 and P for subset of transit signals in a subset of light curves of Lilith-4.

The light curves that were used, were not used for training, validating or testing of the RNN as they were set aside for this experiment from the beginning. The first selection of light curves contained all light curves

[1] https://docs.astropy.org/en/stable/timeseries/bls.htmls

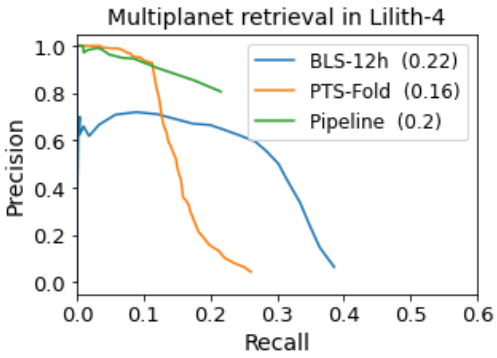

Figure 4.24: [TODO: caption; between brackets shows area under the curve (AUC), also interpreted as average precision] [TODO: explain that we have no uncertainties, and these results therefore only tell something about this specific experiment. future research should confirm these findings.]

that had more than one sector of data available. Subsequently, all light curves with EBs or BEBs were filtered out, as well as light curves that contained no transit signals. Counting each sector individually, we are then left with 6511 light curves. Only a subset of the transit signals is used for evaluation. For example, we ignored all transit signals that occurred less than three times in a single light curve, or had a period smaller than two days. For these choices, 3841 of the light curves contained no transit signals that were used for evaluation, 2130 contained transit signals of a single planet, 474 of two different planets, 57 of three different planets and 9 of four different planets.

One baseline in this experiment is BLS-12h from Section 4.5. Another baseline is the TESS pipeline, for which the detections for this data set are publicly available[2]. However, as opposed to BLS, we have no control over the amount of candidates that are returned by the pipeline. In fact, the pipeline has already set a detection threshold, so we cannot anymore evaluate the entire range of thresholds to analyze the trade-off between precision and recall. [TODO: explain light curve preprocessing for BLS (outlier removal, detrending)].

We only used PTS-Fold for this experiment, because it was most competitive to BLS in the previous experiment. The gaps in the low-risk detrended light curves were filled by linear interpolation, and centroid data was used as additional input to the RNN that is used as the basis of PTS-Fold. Again, both PTS-Fold and BLS-12h were allowed to return a maximum of three candidate detections, even though there are a few samples with four separate transit signals. This was mainly to reduce the computation time for BLS.

Figure 4.24 shows the precision-recall curves for each detection method in the given task. Again, the curves differ strongly from the ones in figures 4.22 and 4.23. For the pipeline detections, this is because we only have detections above the pre-selected threshold available. The results shown here were obtained by setting different thresholds on the SNR provided with each pipeline detection, to define new sets of detections for each threshold. For PTS-Fold and BLS-12h, it is notable that both approach a recall of 1, for gradually decreasing precision. The reason for this behaviour is not entirely clear [TODO: edit text according to new figure. there was a bug which is now fixed and the figure is updated]. Another interesting point is that there is no clear winner between PTS-Fold and BLS-12h in this experiment. Although PTS-Fold has the highest precision for low values of recall, BLS-12h seems to be in favor for larger recall values. In addition, the area under the curve (AUC) for both methods indicate that BLS-12h had a slightly higher average precision over the full range of recall values. [TODO: write a bit more, also mention how many planets were found by one method which were not found by the other]

[2] https://archive.stsci.edu/missions-and-data/tess/data-products/lilith-4

Chapter 5

Discussion

[TODO: write short chapter summary and structure]

5.1 Implications of results

In the end, the most important result of the experiments is not whether there was a winner with slightly more correct detections than the other methods. The fact is that in each case, our RNN-based detection algorithm was able to find planets that BLS could not find, given our implementation of this algorithm.

[TODO: explain and emphasize that complementary algorithms are of great importance in the search for new exoplanets]

[TODO: furthermore, RNN did show a larger potential in the task of monotransit detections, which is considered harder than multi transit detection. RNN does not rely on periodiciies and detrending, and thus is ideal for monotransit detection. Suggestion: future research focus on monotransit detection]

5.2 Limitations and suggestions

[TODO]
• simulated data != real data - though in 4.3 we provide examples using real data
• Lilith-4 was generated before flight (see differences (https://iopscience.iop.org/article/10.3847/2515-5172/ab35e0/meta) with real TESS data)
• nevertheless, simulators provide a great tool for training our method, provided good quality
• uncertaintainty over P, t0 not provided by the model
• some design choices depend on previous choices - all choices are considered in combination with each other (e.g. fully tuning each network architecture) gives more complete and reliable comparisons, but was computationally infeasible for this project
• transits could overlap (as described in 3.1.1.5.) - model could be able to handle this - but also it was not trained on such events, so it might get confused
• longest transits - need long term memory (all in-transits points classified as signal), instead one could have the model only detect ingress and egress of transit signals (begin and end), and define detections by consistent pairs - this might also help for overlapping transit

5.3 Future directions

[TODO] • though we have only experimented with plain transit signals, one could simulate only, e.g. disintegrating rocky exoplanet transit signal to conduct a more focused search
• predict and utlize parameters such as duration, depth, etc to be used for computing SNR (= depth/noise * sqrt(duration)) to subsequently sett a threshold for detected candidates
• then an extension could be to provide uncertainties over these parameters, though tricky
• furthermore, aim to include periodicity into output of the model (struggled with during thesis)
• develop a benchmark dataset + clearly defined task (e.g. requiring determination of P and t0), so AI practitioners from outside this field could contribute further

Chapter 6

Conclusions

[TODO]
- summary of thesis
- summary of results
- link to introduction - catchy outro

References

RL Akeson, X Chen, D Ciardi, M Crane, J Good, M Harbut, E Jackson, SR Kane, AC Laity, S Leifer, et al. The nasa exoplanet archive: data and tools for exoplanet research. *Publications of the Astronomical Society of the Pacific*, 125(930):989, 2013.

Ahmed Alaa and Mihaela Van Der Schaar. Frequentist uncertainty in recurrent neural networks via blockwise influence functions. In *International Conference on Machine Learning*, pages 175–190. PMLR, 2020.

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.

David J Armstrong, Don Pollacco, and Alexandre Santerne. Transit shapes and self organising maps as a tool for ranking planetary candidates: Application to kepler and k2. *Monthly Notices of the Royal Astronomical Society*, page stw2881, 2016.

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.

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.

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.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

Christina Hedges, Nicholas Saunders, Geert Barentsen, Jeffrey L Coughlin, Josè Vinícius de Miranda Cardoso, Veselin B Kostov, Jessie Dotson, and Ann Marie Cody. Four small planets buried in k2 systems: What can we learn for tess? *The Astrophysical Journal Letters*, 880(1):L5, 2019.

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.

Seong Jae Hwang, Ronak R Mehta, Hyunwoo J Kim, Sterling C Johnson, and Vikas Singh. Sampling-free uncertainty estimation in gated recurrent units with applications to normative modeling in neuroimaging. In *Uncertainty in Artificial Intelligence*, pages 809–819. PMLR, 2020.

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.

Miguel Jara-Maldonado, Vicente Alarcon-Aquino, Roberto Rosas-Romero, Oleg Starostenko, and Juan Manuel Ramirez-Cortes. Transiting exoplanet discovery using machine learning techniques: A survey. 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.

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

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.

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.

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.

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.

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.

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.

N Schanche, A Collier Cameron, G Hébrard, L Nielsen, AHMJ Triaud, JM Almenara, KA Alsubai, DR Anderson, DJ Armstrong, SCC Barros, et al. Machine-learning approaches to exoplanet transit detection and candidate validation in wide-field ground-based surveys. *Monthly Notices of the Royal Astronomical Society*, 483(4):5534–5547, 2019.

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.

Susan E Thompson, Fergal Mullally, Jeff Coughlin, Jessie L Christiansen, Christopher E Henze, Michael R Haas, and Christopher J Burke. A machine learning technique to identify transit shaped signals. *The Astrophysical Journal*, 812(1):46, 2015.

Cheng Wang, Carolin Lawrence, and Mathias Niepert. Uncertainty estimation and calibration with finite-state probabilistic rnns. *arXiv preprint arXiv:2011.12010*, 2020.

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.

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

A.1.1 LPSC

• pre-recorded oral presentation at Lunar and Planetary Science Conference
• published abstract https://ui.adsabs.harvard.edu/abs/2021LPI....52.2080R/abstract
pdf: https://www.hou.usra.edu/meetings/lpsc2021/pdf/2080.pdf

A.1.2 EAS

• live oral presentation at European Astronomical Society congress (June 29)
• abstract [first author .. ? - author list is switched up .. all symbols are question marks ...]:
https://eas.kuoni-congress.info/2021/programme/pdf/paperToPdf.php?id=1918

A.1.3 EMM

• several 5-10 minute presentations at EuroMoonMars workshops