Detecting Exoplanets Using Machine Learning and Radial Velocity Data

Režić, Ante

SVEUČILIŠTE U SPLITU
PRIRODOSLOVNO MATEMATIČKI FAKULTET

DIPLOMSKI RAD

DETECTING EXOPLANETS USING MACHINE LEARNING AND RADIAL VELOCITY DATA

Ante Režić

Split, rujan 2024.

Temeljna dokumentacijska kartica

Diplomski rad

Sveučilište u Splitu

Prirodoslovno-matematički fakultet

Odjel za informatiku

Ruđera Boškovića 33, 21000 Split, Hrvatska

DETEKTIRANJE EGZOPLANETA POMOĆU STROJNOG UČENJA I PODATAKA RADIJALNE BRZINE

Ante Režić

SAŽETAK

Cilj je ovoga rada kreirati sustav generiranja sintetskih podataka radijalne brzine zvijezde izazvane postojanjem egzoplaneta. Razviti će se model strojnog učenja koji će iz sintetskih periodogramskih podataka prepoznavati postojanje planetarnih signala. Model će se zatim validirati na stvarnim podacima. Ovaj će rad pružiti nastavni korak za razvoj sustava dubokog učenja za područje detekcije egzoplaneta koje se iz godine u godinu povećava.

Ključne riječi: egzoplenti, strojno učenje, sintetički podaci, python

Rad sadrži:	63 stranice, 32 grafičkih prikaza i 26 literaturnih navoda. Izvornik je na engleskom jeziku.
Mentor:	**doc. dr. sc. Divna Krpan,** *docent Prirodoslovno-matematičkog fakulteta u Splitu, Sveučilišta u Splitu*
Ocjenjivači:	**doc. dr. sc. Divna Krpan,** *docent Prirodoslovno-matematičkog fakulteta u Splitu, Sveučilišta u Splitu*
	Dino Nejašmić, mag. educ. math. et inf., *predavač Prirodoslovno-matematičkog fakulteta u Splitu, Sveučilišta u Splitu*
	Nika Jerković, *asistent Prirodoslovno-matematičkog fakulteta u Splitu, Sveučilišta u Splitu*

Rad prihvaćen: **rujan, 2024**

Basic documentation card

Master Thesis

University of Split

Faculty of Science

Department of Informatics

Ruđera Boškovića 33, 21000 Split, Croatia

DETECTING EXOPLANETS USING MACHINE LEARNING AND RADIAL VELOCITY DATA

Ante Režić

ABSTRACT

The goal of this thesis is to create a system for generating synthetic data of the radial velocity of a star caused by the existence of an exoplanet. A machine learning model will be developed that will recognize the existence of planetary signals from synthetic periodogram data. The model will then be validated on real data. This thesis will provide a developmental step for the development of deep learning systems for the growing field of exoplanet detection.

Key words: exoplanet detection, machine learning, synthetic data, python

Thesis deposited in library of Faculty of science, University of Split

Thesis consists of: 63 pages, 32 figures and 26 references.
Original language: English.

Mentor: **doc. dr. sc. Divna Krpan,** *Assistant professor at the Faculty of Science and Mathematics, University of Split*

Reviewers: **doc. dr. sc. Divna Krpan,** *Assistant professor at the Faculty of Science and Mathematics, University of Split*

Dino Nejašmić, mag. educ. math. et inf., *Lecturer at the Faculty of Science and Mathematics, University of Split*

Nika Jerković, *Assistant at the Faculty of Science and Mathematics, University of Split*

Thesis accepted: **September, 2024**

Table of Contents

Introduction

In the field of astronomy, exoplanet detection is a new and rapidly emerging field of science dedicated to finding planets around other stars. Unlike the planets in our own solar system, which are very close to us on astronomical scales, extrasolar planets pose a challenge in understanding and characterising their nature. Unusual compositions and orbits challenge our understanding of planet formation and the history of the universe.

Machine learning could prove essential in the future of exoplanet detection. Currently, only the radial velocity discovery method can tell us the exact mass and orbital parameters of planets orbiting other stars.

Orbiting planets exert a measurable and repeating force on their parent star – moving the star away and towards the earth. From these oscillations, astronomers can extract planetary data. In the real world, noise heavily disturbs this process. [3]

Stars themselves produce non-random noise which can mimic planetary influence. For example, sunspots, areas of a star where local temperature falls below the rest of the star, can be detected as a fall in brightness. There have been attempts at masking these and similar influences, but at most they were used to corroborate true positives of planetary detection.

As radial velocity is directly dependant on the mass and orbit of the planets, a system which hold one massive planet close in to the star will dominate the radial velocity measurement, or many smaller planets will have their sinusoidal traces stretched over many years of measuring.

The Milky Way contains roughly 100 billion stars, of which as much as 8.3% might contain planet discoverable though the radial velocity method. Classical processes may take as much as 427 seconds [2] to process each star system.

As part of this master's thesis, we will discuss the possibility of using machine learning models to process vast amounts of radial velocity data and measure their usefulness. We will create synthetic data and train a machine model to identify planetary peaks within the data and comment its usefulness on both synthetic and real data.

Due to lack of full system data, where all planets in a star system are known, synthetic data generation will be necessary. Validation will be done on some real-world data procured by current telescopes and measured with the F1 score, a harmonic mean of accuracy and recall.

1. Systematic Literature Review

Preferred Reporting Items for Systematic Reviews and Meta-Analyses" (PRISMA) [1] is used to confirm to proper use of compute and procedures for the thesis. The PRISMA process was centred around key questions, and the process of literature review is outlined in the following sections.

1.1. Outline

The following questions need to be answered as part of the literature review:

1. What is radial velocity and how is it measured?
2. How is radial velocity data stored?
3. How do astronomers extract planetary data from radial velocity?
4. How to generate synthetic radial velocity data?
5. How does noise and temporal sampling influence periodograms?
6. What impact did machine learning models have so far in the field of exoplanet detection via radial velocity?
7. What are the key limitations of using this method for exoplanet detection?

These questions were formulated to provide a clear path towards model development. Publications have been filtered using the exclusion criteria and the inclusion criteria found in the tables below.

Table 1 Exclusion Criteria

Exclusion Criteria
Exclude studies not written in English.
Exclude studies published before 2015.
Exclude studies that do not use only radial velocity data for exoplanet detection.
Exclude theoretical papers that do not provide empirical data or practical insights into radial velocity measurements.

Exclude non-peer-reviewed articles, editorials, letters, and conference abstracts, tutorials, presentations.
Exclude not publicly available studies.

While keeping the exclusion criteria in mind we can proceed with further refinement of our literature selection. The inclusion criteria below are designed to primarily answer our research questions.

Table 2 Inclusion Criteria

Inclusion Criteria
Include studies that explicitly investigate or utilize radial velocity measurements for exoplanet detection.
Include studies that discuss and present ways to generate realistic synthetic radial velocity data.
Include studies that apply machine learning techniques to the analysis of radial velocity data for the purpose of detecting exoplanets.
Include studies that describe methods for storing and handling radial velocity data.
Include studies that detail the processes and algorithms used by astronomers to extract planetary data from radial velocity measurements.
Include studies that assess the impact of machine learning models on the field of exoplanet detection via radial velocity.
Include studies that discuss the limitations and challenges of using radial velocity data and machine learning for exoplanet detection.
Include studies that validate machine learning models on real-world radial velocity data, especially those that use data from current ground-breaking telescopes.

1.2. Executing the literature search

Following the definition of research questions, the exclusion and inclusion criteria, a search was executed using the following keyword combinations:

Keyword combinations:

- radial velocity AND exoplanet detection OR machine learning AND exoplanet detection
- stellar noise AND exoplanet

Table 3 Literature search results

Archive name	**Total results**
arXiv	32
IEEE Xplore	69
Google Scholar	~50

These archives were processed one by one, and after a sufficient number of papers were collected – ones that are relevant to all the research questions, the search for further archives was stopped.

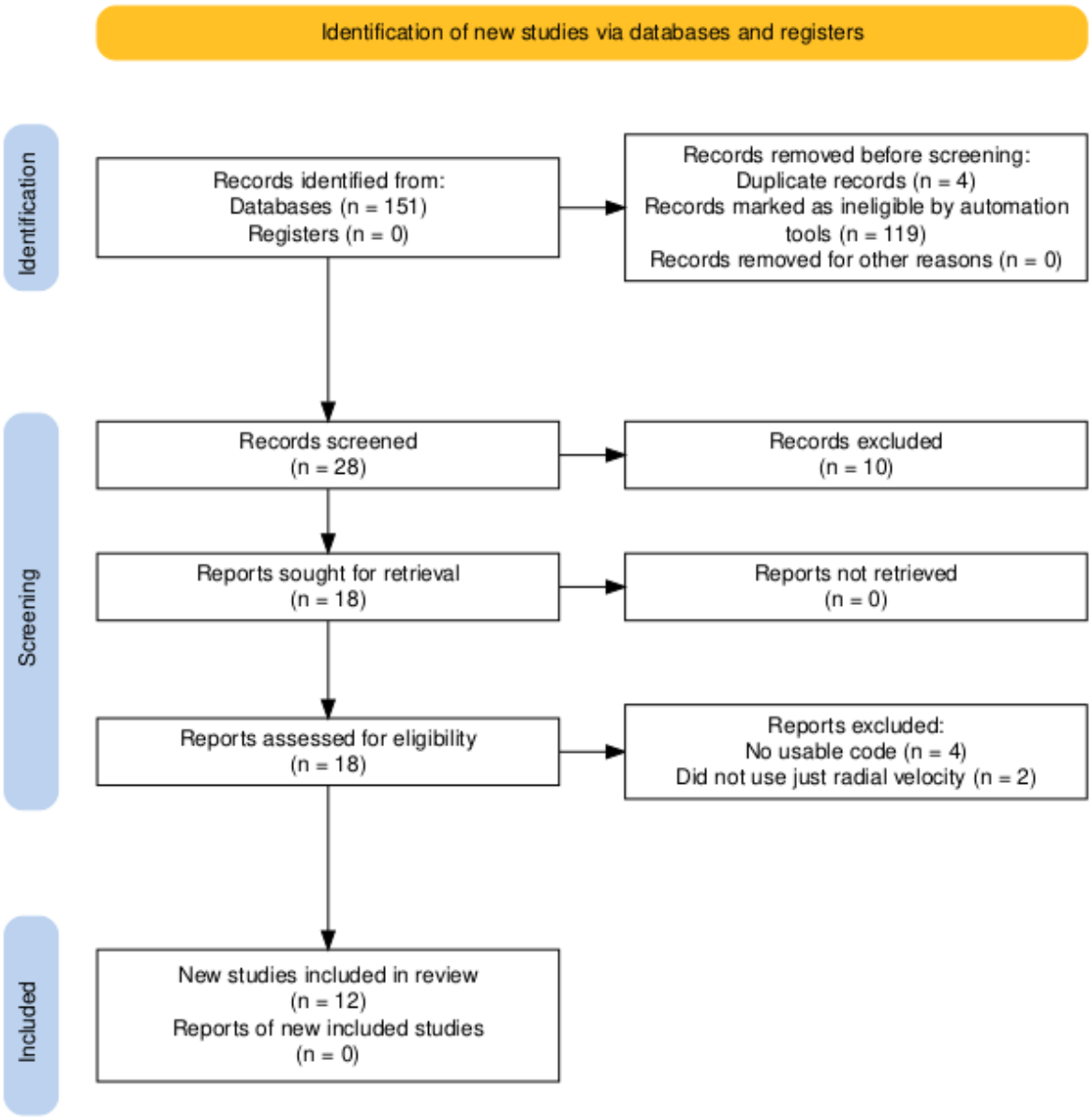

Figure 1 PRISMA diagram of publications

These are the remaining publications:

Table 4 Final selected publications

Tile	Authors	Publication Date
ExoplANNET: A deep learning algorithm to detect and identify planetary signals in radial velocity data [2]	L. A. Nieto and R. F. Díaz	June 2023

Identifying Exoplanets with Deep Learning. IV. Removing Stellar Activity Signals from Radial Velocity Measurements Using Neural Networks [4]	Zoe. L. de Beurs, Andrew Vanderburg, Christopher J. Shallue, …	August 2022
The need for a public forecast of stellar activity to optimize exoplanet radial velocity detections and transmission spectroscopy [5]	Lalitha Sairam and Amaury H. M. J. Triaud	May 2023
Exoplanet Detection: A Detailed Analysis [6]	Mahima Kaushik, Aditee Mattoo and Dr. Ritesh Rastogi	April 2024
Modelling Stellar Jitter for the Detection of Earth-Mass Exoplanets via Precision Radial Velocity Measurements [7]	Samuel Granovsky, Irina N. Kitiashvili, Alan Wray	February 2022
Randomization Inference of Periodicity in Unequally Spaced Time Series with Application to Exoplanet Detection [8]	Panos Toulis and Jacob Bean	May 2021
Statistical Methods for Exoplanet Detection with Radial Velocities [9]	Nathan C. Hara, Eric B. Ford	August 2023
Detection Limits of Low-mass, Long-period Exoplanets Using Gaussian Processes Applied to HARPS-N Solar RVs [10]	N. Langellier, T. W. Milbourne , D. F. Phillips, …	June 2021
Radial-velocity variations due to meridional flows in the Sun and solar-type stars: impact on exoplanet detectability [11]	N. Meunier and A.-M. Lagrange	April 2020

3D Magneto-Hydrodynamical Simulations of Stellar Convective Noise for Improved Exoplanet Detection [12]	S. Sulis, D. Mary, and L. Bigot	February 2020
Filtering Solar-Like Oscillations for Exoplanet Detection in Radial Velocity Observations [13]	William J. Chaplin, Heather M. Cegla, Christopher A. Watson, Guy R. Davies, and Warrick H. Ball	March 2019
A Machine Learning Approach for Correcting Radial Velocities Using Physical Observables [14]	M. Perger, G. Anglada-Escudé, D. Baroch, …	February 2023
Improving Earth-Like Planet Detection in Radial Velocity Using Deep Learning [15]	Yinan Zhao, Xavier Dumusque, Michael Cretignier, …	May 2024
A Gaussian Process Framework for Modelling Stellar Activity Signals in Radial Velocity Data [16]	V. Rajpaul, S. Aigrain, M. A. Osborne, S. Reece, and S. Roberts	June 2015
A detailed derivation of The Radial Velocity Equation [17]	Kelsey I. Clubb	August 2008

1.3. Answering research questions

1.3.1. What is radial velocity and how is it measured?

As the Earth orbits the Sun, it induces radial motion dependent on its distance and mass. Due to the Earth's much smaller mass, the Sun moves far less than the Earth – which travels more than 940 million kilometers over the course of a year. Exoplanets also disturb their parent stars, and this motion can be detected using Doppler spectroscopy. Due to the nature of waves, when an emitting source moves away from an observer, its wavelength increases, effectively red-shifting the signal. And when the source moves towards the observer, the

wavelength decreases, causing a blue shift. By measuring these line-of-sight changes, a graph can be constructed which informs us of the change in radial velocity of a star due to its orbiting bodies. [3]

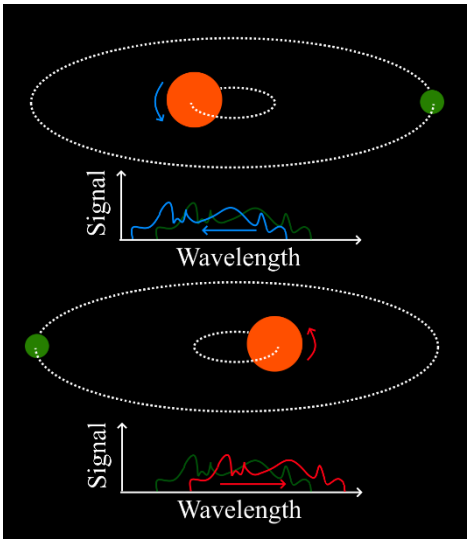

Figure 2 Radial velocity inducet wavelength shift

As a planet orbits a star, it exerts a gravitational force on the star, causing the star to move in a small orbit around the common center of mass (barycenter) of the star-planet system. This motion results in periodic changes in the star's velocity along the line of sight from Earth as seen in Figure 2 Radial velocity inducet wavelength shift.

This motion of the star towards and away from Earth causes shifts in the wavelength of the star's light due to the Doppler effect. When the star moves towards Earth, its light is blue-shifted (wavelengths become shorter), and when it moves away, its light is red-shifted (wavelengths become longer).

Using the measured Doppler-shifted wavelength, and the ground truth wavelength, the following equation is applied:

$$v = c \cdot \frac{\Delta\lambda}{\lambda_{rest}} \tag{1}$$

Where:

- v is the radial velocity
- $\Delta\lambda$ is the Doppler shift in wavelength
- Λ_{rest} is the original wavelength of the light

- c is the speed of light

1.3.2. How is radial velocity data stored?

Radial velocity data usually consist of more than 100 datapoints with the corresponding timestamp, radial velocity measurement and uncertainty.

The timestamps are quasi-uniformly distributed and follow nightly observation cycles. Some contain multiple observations per night.

The intrinsic uncertainty is contained withing some abstract multivariate distribution. Within this value are all the noise sources associated with the star: such as sunspots, magnetic disturbances, convective forces on the star's surface, dust clouds, etc… [18]

1.3.3. How do astronomers extract planetary data from radial velocity?

Once radial velocity data has been gathered, usually by high precision instruments such as HARPS, SOPHIE, CARMENES or ESPRESSO, astronomer employ periodograms.[10] In the analysis of irregularly spaced time series data, such as those commonly gathered in radial velocity (RV) surveys, researchers frequently utilize periodograms to identify repeating patterns. The Lomb-Scargle technique has become a widely adopted tool for uncovering periodic signals within RV datasets.

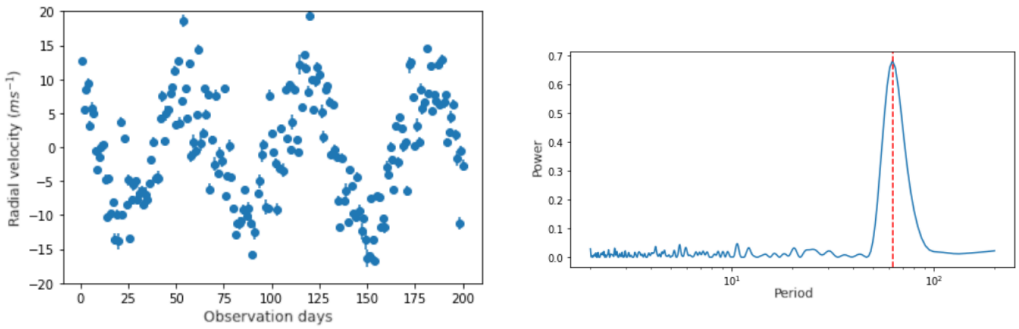

Figure 3 Periodogram from the ExoplANNET paper

Astronomers then identify significant peaks in the periodogram data, as seen in Figure 3 Periodogram from the ExoplANNET paper. This corresponds to the period of the planet, and using the mass of the star, a Keplerian model can illuminate the true mass and semi-major axis of the planet. [19]

Often multiple peaks in the data can be identified, which corresponds to multiple orbiting bodies.

1.3.4. How to generate synthetic radial velocity data?

To effectively train, validate, and test a neural network, a large dataset is essential. While the exact size requirement is not known, it stands to reason that larger training and validation sets tend to increase effectiveness.

The dataset must be representative of the scenarios the network will encounter during inference, such as data point quantity, variability patterns, and the presence of periodic signals.

Current radial velocity (RV) time series from large-scale surveys pose challenges for efficient training due to their limited number and high diversity. These datasets lack sufficient examples of different variability types and planetary system configurations. Furthermore, the uncertainty in the exact number of planets in each system introduces the problem of imprecise labeling, making it difficult to accurately categorize each dataset for the algorithm's learning process.

An alternative approach involves training and assessing the neural network using synthetic radial velocity (RV) time series [2], followed by an examination of its effectiveness on actual observational data. This method may suffice if the simulated data adequately reflect real-world conditions, though it's likely that some adjustments to the network's structure would be necessary.

We can use deterministic equations from a large number of synthetic star systems to generate the needed data. A drawback is that the significance of the study's findings is inherently tied to the realism of the simulations - the accuracy of the synthetic data directly influences the relevance and applicability of the results to real-world scenarios

To generate the radial velocity data which account for eccentricity and inclination of planets we turn to “A detailed derivation of The Radial Velocity Equation” by Kelsey I. Clubb [17]. In it, the following equation is stated:

$$K_1 = \left(\frac{2\pi G}{P}\right)^{\frac{1}{3}} \frac{m_2 \sin i}{m_1^{2/3}} \frac{1}{\sqrt{1-e^2}} \tag{2}$$

Where:

- K_1 is the radial velocity semi-amplitude of the star
- P is the period of orbiting body in seconds
- E is the eccentricity of the orbiting body
- i is the inclination of the orbiting body
- m_1 is the mass of star
- m_2 is the mass of orbiting body

The process used to generate the radial velocity data can be summarised as;

1. Generate planetary system and store them for later use. These systems should vary in size and mass and be representative of the distribution of real-life planetary systems.
2. Using the radial velocity function, calculate the ground-truth value for the velocity.
3. Add noise and uncertainty to the data; Pulsation, Granulation, Rotational modulation and instrument noise.
4. Sample the data at uneven intervals mimicking the observation schedules of real-life observatories.

1.3.5. How does noise and temporal sampling influence periodograms?

Using the generated velocity and time arrays, periodograms can be created. In Figure 4 Evenly sampled simulated periodogram with planets and masses labelled we see two graphs of the same planetary system. The first graph uses roughly daily observational intervals i.e. it simulates what an observatory might detect over the course of 7000 days while making daily measurements.

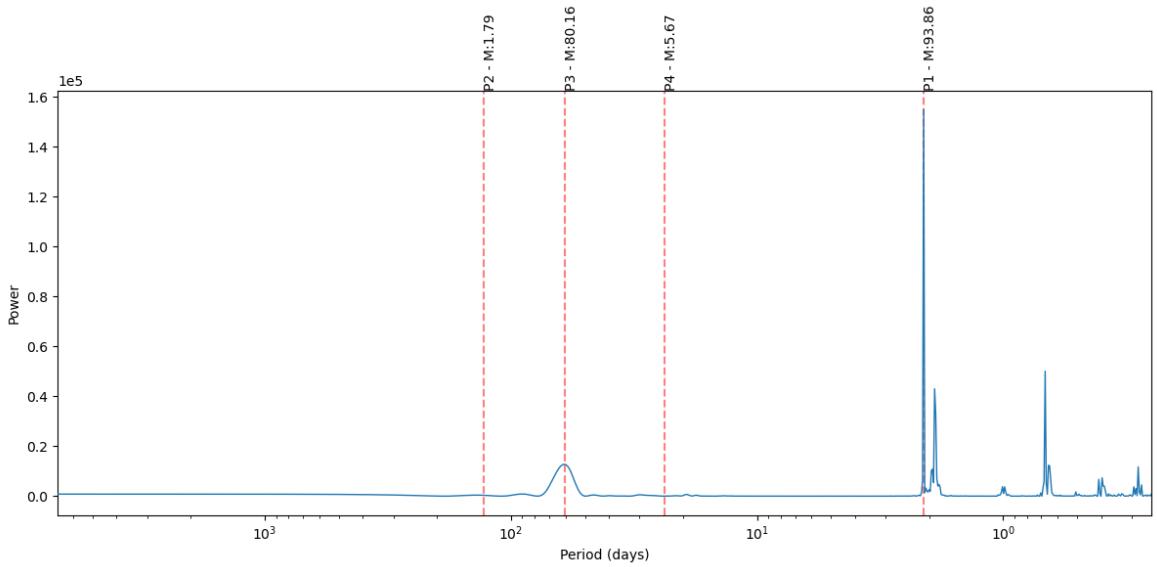

Figure 4 Evenly sampled simulated periodogram with planets and masses labelled

The power corresponds to the intensity of the signal within the data at the specific frequency (period is used in the plot for ease of understanding). We can see, labelled with red dashed lines, ground truth plants and where we would expect to see their signals. Planets 1 and 3 have clearly visible peaks.

Planets 2 and 4, who have the lower mass, do not contribute as much as the others, and so their power is significantly lower. It represents the most common cause for false negatives in exoplanet detection – low masses, which do not perturb the star. To detect it, the sampling would need to be faster, the mass ratio different, the noise absent, or the other planets removed.

Here is the system's periodogram if we removed the other planets from the system:

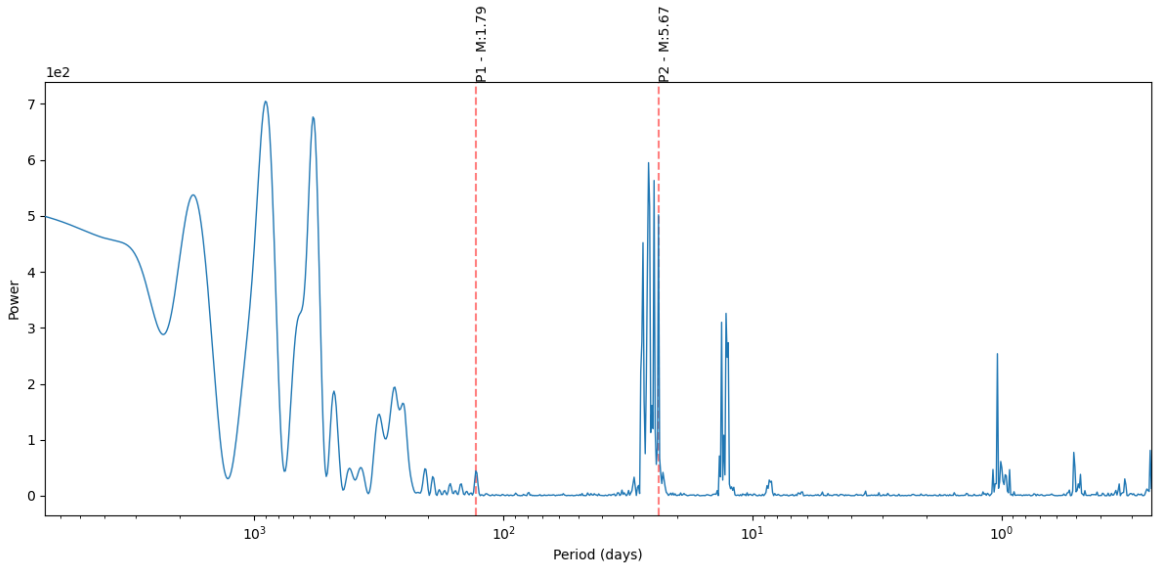

Figure 5 Small planets relative to stellar noise

Even now, the strength of the planet signal is miniscule. This systems star measures 0.72 solar masses, and the planets themselves are more massive than Earth. This clearly illustrates the difficult proposition of earth-like planet detection around even sunlike stars.

As before, we can see other peaks which are connected to the stellar noise, which is described in more depth later.

Finally, let's take a look at a periodogram of this system with realistic temporal sampling i.e. sampling we could expect from real-life observatories;

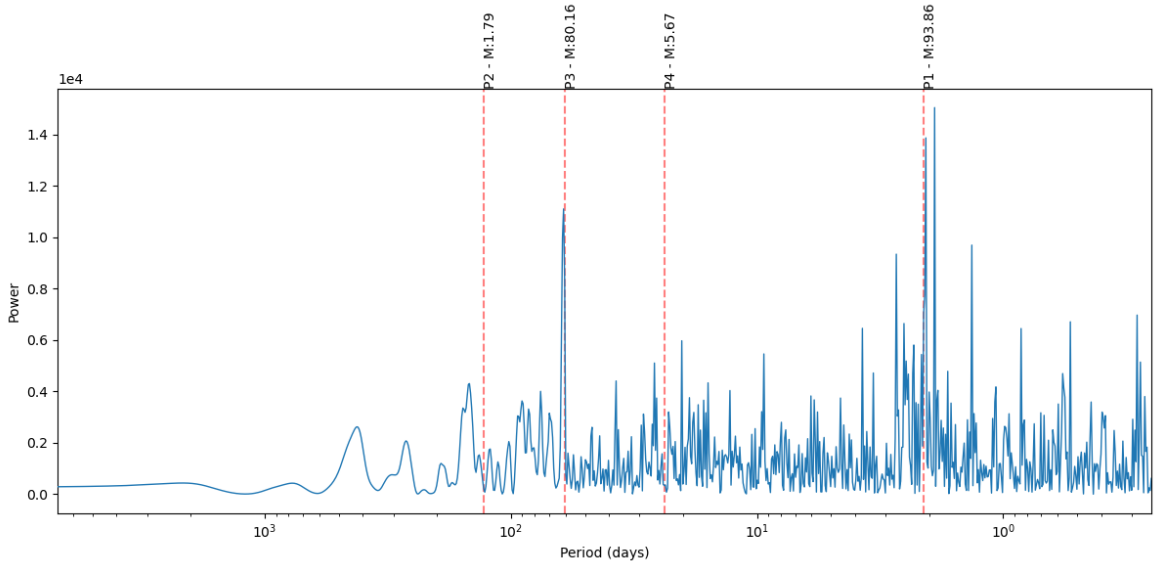

Figure 6 Periodogram with all noise sources

Many sources of noise worth together to render the periodogram messy. We can see a more ordered signal nearing 100-day periods and beyond – due to the most noise sources being shorter in their periodicity. Using our human intuition, we could only conclude the existence of a single planet, Planet 3, in the data. A machine learning model might do better.

1.3.6. What impact did machine learning models have so far in the field of exoplanet detection via radial velocity?

Due to the constraints of time and complexity of manual exoplanet detection, machine learning models can offer much greater volumes of processed data. The ExoplANNET framework achieves 28% fewer false-positive with a five orders of magnitude speedup. [2]

1.3.7. What are the key limitations of using this method for exoplanet detection?

Influenced by the variability of the star and the instrument sensitivity, the radial velocity detection method suffers greatly from the introduced noise. Alongside the fact that the inclinations of exoplanets are rarely aligned as so to make the observations as simple extraction of planetary parameters. More often than not, a planet does not orbit edge on when viewed from earth, and astronomers, as well as a machine learning model, will need to account for the angle of the planet's revolution.

The instrument precision over soon-to-be online telescopes has reached $10cm^{-1}$, which over long periods of observations, might be able to detect the influence of the earth on the sun, which is recorded at $9cm^{-1}$. [20]

Detecting low-mass, long-period planets require numerous observations to capture the entire sinusoidal pattern, a process that can span a decade or more. Identifying planets with larger masses (approximately 0.5 m s−1) using current-generation spectrographs and Gaussian Process (GP) regression would demand more than 12 years of densely sampled radial velocity (RV) observations.

2. Machine Learning for Radial Velocity

Machine Learning has had an immensely positive impact on the field of astronomy. [6] As the volume of collected data increases year over year, aided by the newly constructed observatories such as the James Webb Space Telescope, and the upcoming Vera Rubin Observatory and the Extremely Large Telescope. There has crystalized a real need to process immense amounts of data quickly and efficiently.

A potential path forward to processing this data is the use of machine learning. Specifically, in this thesis, the use of neural networks for exoplanet detection will be discussed.

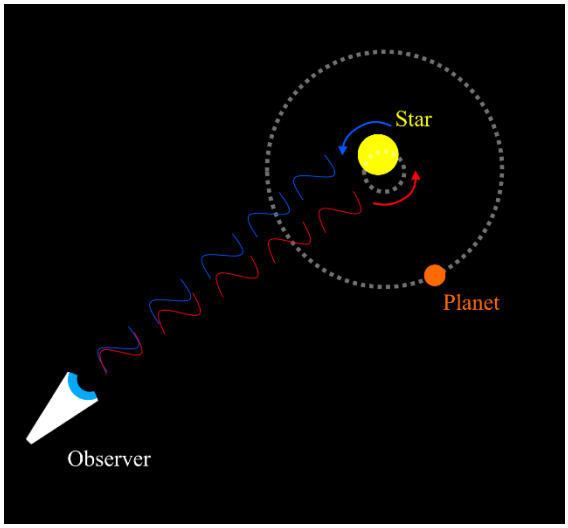

Figure 7 Radial velocity visualisation

As the motion of the exoplanet around the star presents a repeating periodic signal, one which can be processed into a periodogram, a convolutional neural network is likely to be the best option to process this data.

2.1. Neural Networks

Neural networks are a form of machine learning consisting of many layers. Stacked on top of one another, these layers are in turn comprised of individual neurons – very much inspired by biological brains.

By nesting many of these neurons together in a layer, and in turn combining many layers together, we construct an arbitrary mathematical operation. The outputs of all the neurons of a single layer are connected to the inputs of all the neurons in the following layer.

In principle, due to the unique set of weights and biases, each input corresponds to an unique set of activations and propagations though the neural network.

2.1.1. Neurons

Smallest units of a neural network. They conduct a multiplication of its input with its internal weight. A bias is added, also internal to each neuron, and finally, an activation function maps the value according to a specific function.

2.1.2. Activation Functions

These functions are applied to the output of a neuron to transform the value from one value space to another. They are usually set on a per-layer basis and are crucial for introducing non-linearity.

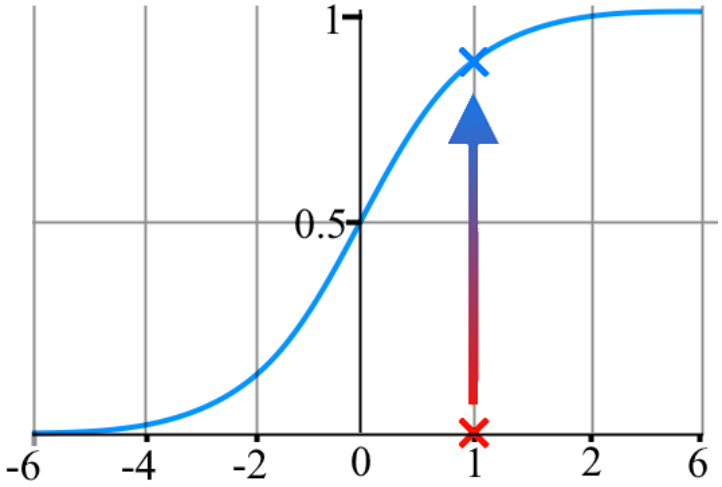

Figure 8 Logistic aka Softmax function

Visualised on the image above, the activation function maps its input X = 2 to the output value of approximately 0.88.

This particular activation function is useful for outputs of classification tasks, because it clamps the value of the output to between 0 and 1, while allowing the inputs to be practically anything. This then also serves as the neural networks confidence value for each specific class, and an output of 0.6 would be considered significantly different than an output of 0.99, even though they both technically classify as 1.

2.1.3. Backpropagation

Setting the weights and biases of a neural network is too complex a task for a human to do. We just set the initial parameters. Imagine a single neuron with a weight of 0.5 and a bias of 3, using a linear identity activation function. We want this neuron to output the next integer value that is twice the input. To train it, we prepare a dataset of input-output pairs like (1,3), (4,9), (10,21).

For the input 10, the neuron's output would be 8. We calculate the error (13) and use backpropagation to adjust the weight and bias. Repeating this process for multiple pairs, we update the neuron's parameters until the error is minimized.

In practice, the weight might become 1.999 and the bias 1.0000001. The learning rate, adjusted by an optimizer, governs the speed of these adjustments to avoid local minima and reach the global minimum.

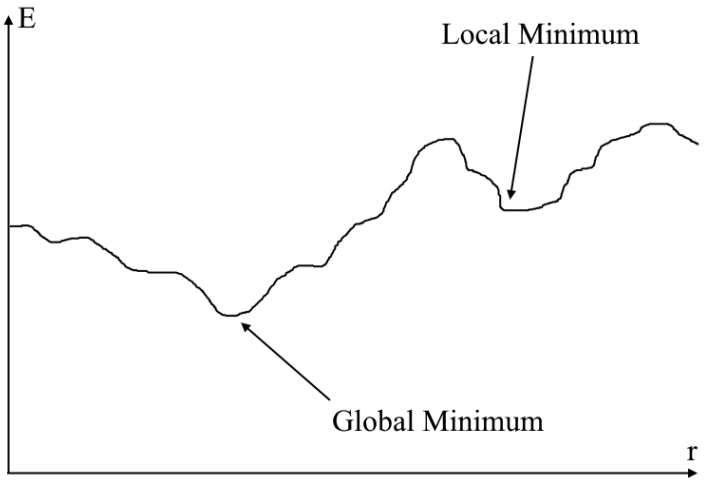

Figure 9 Local and global minima

On the image above, we see a simplified representation of the gradient space of a neural network. R represents some combination of internal parameters, while E is the error. A neural network at the start will have some random combination of r, and therefore some random location on the gradient. Trought training and backpropagation, we hope to reach the global minimum.

2.2. Types of Layers

2.2.1. Dense

A dense layer of the neural network contains neurons which are full connected to the preceding layer. They are the core of reasoning in neural networks and are often used in groups. During training, the weights and biases of dense layers are tuned to reduce the loss function.

Multiple dense layers are computationally expensive and are better used at the end of a neural network after feature extraction to make the final decisions for the network.

2.2.2. Convolution

A special type of a neural network layer is the convolutional layer. These layers learn abstract shapes within the data using a kernel. For example, a convolutional neural network identifying cats and dogs might have a kernel that identifies ears. Deeper layers may learn specific ear shapes and breed differences.

A convolutional layer slides a kernel across its input, multiplying and summing values to create a mapping. The kernel is learned during training, and its movement is defined by its stride. Padding allows the layer to maintain input dimensions and learn edge features. The number of kernels determines the number of output feature maps.

If we have a kernel shaped like a cat's ear, the convolution result would be a feature map with high values where the cat's ears are present. An activation function is applied to each feature map member before passing to the next layer. This helps the network learn non-linear features.

2.2.3. Batch Normalization

During training and inference, input values can become unstable due to successive transformations by layers. Batch normalization layers can solve this by calculating the mean and standard deviation of the batch, then normalizing the values to preserve their distribution without affecting trainability. During training, batch normalization layers learn internal parameters (gamma and beta) and store running averages of the mean and variance for use

during inference. The accuracy of batch normalization depends on the batch size, with larger batches providing more accurate mean and variance values.

2.2.4. Dropout

Dropout layers fight overfitting by randomly setting to 0 their inputs based on a single hyperparameter. It is called the dropout rate and is defined as a percentage, so a dropout layer with a dropout rate of 20% would drop 1 in 5 of its input neurons.

This operation reduces the sum of the neurons, which can negatively impact layers downstream. To preserve the approximate sum, the neurons that have not been dropped have their values increased by a value equal to the inverse of 1 minus the dropout rate. With a dropout rate of 0.2, the scaling factor would be 1.25.

During inference, no neurons are dropped, but the opposite problem as the one during training appears. We now must scale down the sum of the neurons, which we do by multiplying the values by 1 minus the dropout rate. With a dropout rate of 0.2, we would multiply the values by 0.8.

3. Radial Velocity Data

3.1. Real Radial Velocity Data

As of the writing of this paper, 1093 exoplanets have been discovered using the radial velocity data. Their radial velocity graph is hosted on the NASA exoplanet archive, and a convenient script is provided to download the data.

Once downloaded, the data contains the star name, dates and values of observations and the uncertainties.

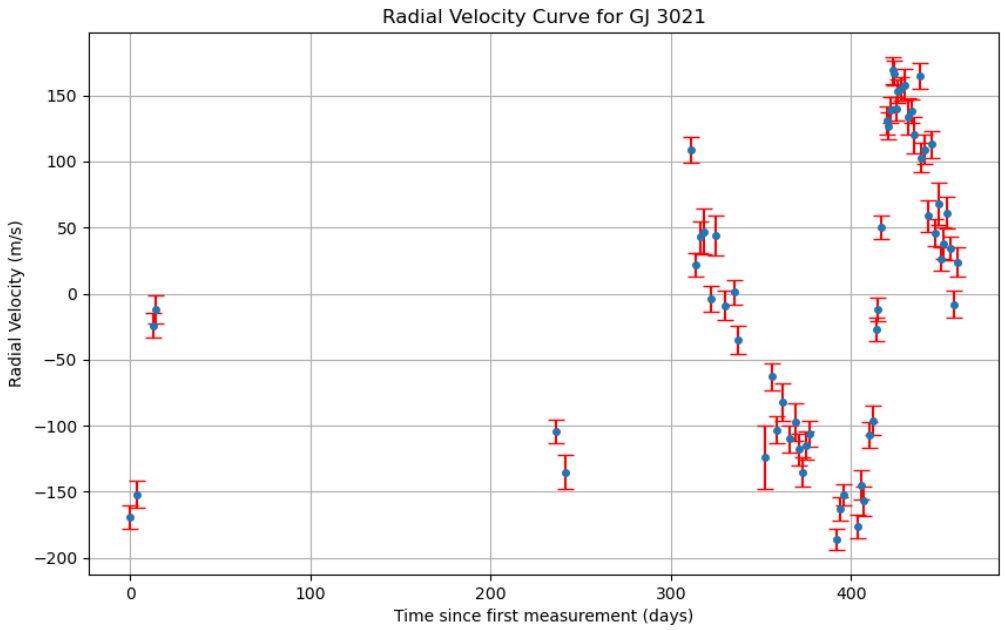

Figure 10 Radial Velocity curve of Gliese 3021 perturbed by its planet - GJ 3021 b

Figure 10 Radial Velocity curve of Gliese 3021 perturbed by its planet - GJ 3021 b shows the real-life data of the exoplanet Gliese 3021 b as it orbits its star and perturbs it by as much as 170 meters per second. A portion of the radial velocity data obtained in this way has its values offset by a large amount, such as on the above graph – the original values were about 5000 meters per second lower.

This data simply tells us the motion of the star relative to earth, and often when making observations of stars in binary systems, the velocity is dominated by the companion star.

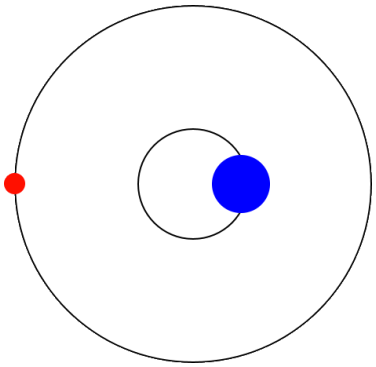

Figure 11 Binary system

Figure 11 Binary system shows the relationship between orbiting objects; If we would take radial velocity measurement so the red star in the above image, we would get exceptionally high values – caused by the more massive blue star. It is important to consider that planets which would orbit the red star, are more likely to be detected if their period is significantly smaller than the period of the red star's orbit. As we need many observations of its periodicity compared to the period of the star. [10]

Because of this, we normalised the data to have a mean of 0.

By analysing this data, we can paint a clearer picture of real-life radial velocity measurements. One of the most important questions is; What is the distribution of differences in time between subsequent observations? This can be computed in buckets, and for the sake of removing outliers, we count all times beyond 100 days as a single bucket and cap it to 365 days.

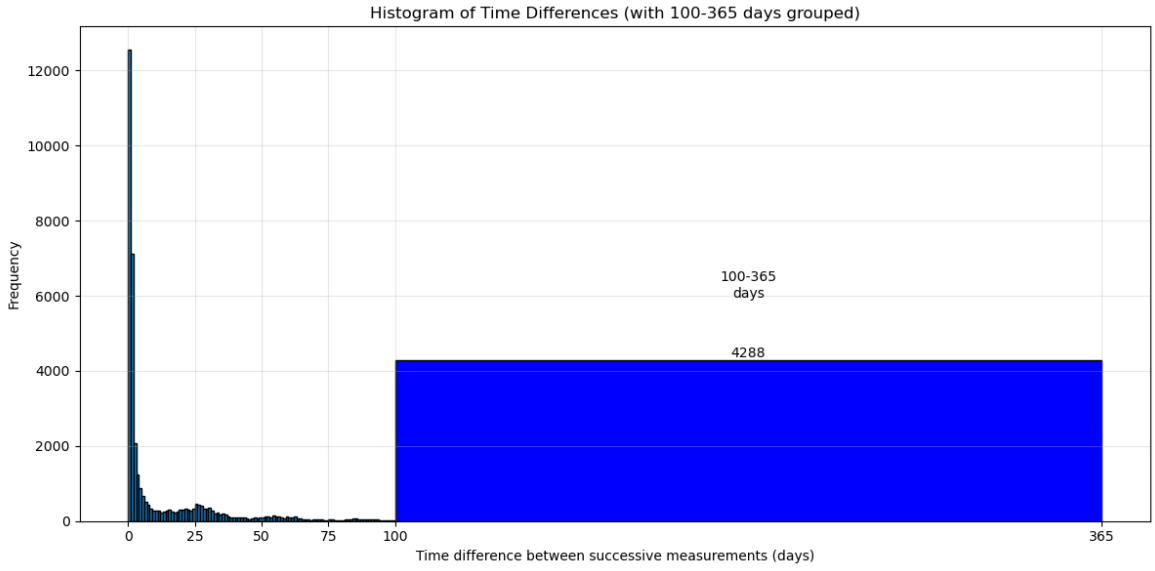

Figure 12 Histogram of delta time between observations

Figure 12 Histogram of delta time between observations is the binned histogram of the differences between two subsequent observations based on real data. We see a sharp maximum in sub-1-day observations and another peak at roughly 25 days. This will guide our synthetic observations. Below can see the stats for ou time difference observations. Evident is the very large range of values with the mean at 33.04 days and a standard deviation of 67.71.

Statistics of time differences (up to 365 days):

- Minimum: 0.00 days
- Maximum: 364.96 days
- Mean: 33.04 days
- Median: 2.92 days
- Standard deviation: 67.71 days
- Mode (rounded to nearest day): 1 days
- 25th percentile: 0.99 days
- 75th percentile: 28.01 days

This histogram will later be used to compute the temporal noise. A perfect use case would be an hourly measurement, but real-life has to contend with; Yearly motions of the earth around the sun – which prohibit observations of certain portions of the night sky. And observation schedules and maintenance of the instrument.

We can compute periodograms for these real-life systems and see if we can identify some planets with the naked eye.

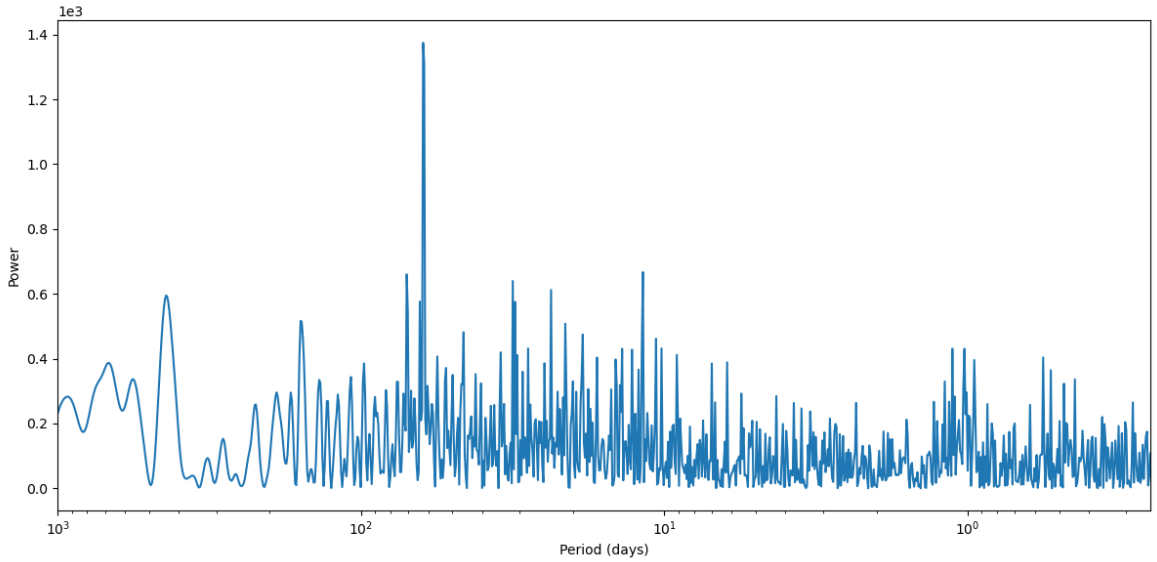

Figure 13 HD 3651 periodogram

Figure 13 shows a periodogram of star HD 3651. Above we can see a clear spike at about 60 days. This corresponds to a period of ~62 days of the planet HD 3651 b. Now that we are certain these radial velocities can be used to find planets, can move on to generating synthetic systems.

Two scripts have been created that generate and preprocess the data, these are: main_generate_systems.py and main_preprocess.py. These are available at my GitHub[26]. The quality of the radial velocity data underpins this entire project – we need to ensure both synthetic data contains enough real-world equivalence, so that the eventual machine learning model may create an accurate internal world-model.

3.2. Planetary System Generation

The first step in generating our radial velocity data, is creating a realistic pipeline for crating star systems. These star systems should contain the orbital characteristic of the planets, their mass and the mass of the star. With these parameters we can later simulate the data. We must choose an assortment of ranges for our random generation.

First the stellar masses, a value between 0.05 and 2 solar masses should account for the vast majority of stars. This will serve a sort of seed value for much of the remining generation. More massive stars usually have more material around them for forming planets and as such are expected to have higher mass planets orbiting it.

Then we must choose to semi-major axes of the planets and their count. The planetary count is defined as a random value a mean of 7 and a standard deviation of 3.3. We discard any planet counts below 2.

A system's compactness can be described as the ratio of the semi-major axis of the first and final planet in astronomical units. Since they have the same units, this value is dimensionless For example, the solar system has a compactness score of 77.758 due to the orbits of Mercury and Neptune. We choose this value at random, between 55 and 88.

Another important value is the semi-major axis of the first planet, this can be thought of as the first semi-major axis. It can also be thought of as a ratio of the first orbit and the star's mass. For the solar system, since the sun is exactly 1 solar mass, this value is 0.387098 AU/Solar mass – the exact value of Mercury's orbit. Ultimately, we have decided to choose this value at random between 0.01 and 1.2, to cover all shapes and sized of planetary systems.

The maximum semi major axis is the value of the first semi major axis multiplied by the compactness.

As we finally have the lower and upper bounds of the system, we begin choosing the semi major axes of the planets within. Starting with the minimum semi major axis, we choose a random spacing value. This is a dimensionless value which corresponds to the ratio of two neighbouring orbits. Its distribution has been calculated using the values in the solar system, the Trappist-1 system, and the Draco-90 systems. We have a mean of 1.614 and a standard distribution of 0.5941. By successively multiplying the orbits with this spacing value we get the full list of planetary orbits.

The first important caveat to this approach is the inherent oversimplification of real-world truths. There may yet exist some unknown relationships between the planetary orbits, the mass of the planets and stars [21], which can influence the process of generating systemic data. Given that such deep truths are unknown to us, and due to the constraints of time and compute, we are forced to assume this naïve approach to system generation. For example, we do not take into account orbital resonances, which force planets over many orbits to fall into stable resonances with each other, such as Pluto who orbits twice for every three orbits of Neptune.

We also don't take into account the effects of general relativity, for example, the orbit of Mercury precesses i.e. rotates around the sun due to the inherent curvature of space-time caused by the sun's mass. Though this effect is small, about 0.001 degrees per century, it nevertheless represents another point of simplification. [1][22]

An earth-like ratio corresponds to the percentage likelihood of a planet having earth-like mass, or mass closer to the massive planets like Jupiter and Saturn. It is chosen at random between 0 and 1.

The for each of the chosen semi-major axes, we choose a mass – a log-normal distribution with mean of 1 earth masses and standard deviation of 0.85, or 0.6 mean Jupiter masses with a deviation of 0.85. By doing this we achieve a total distribution of expected planetary masses in real-life observations.

Then for the eccentricity, i.e. how circular the orbit is, we choose a random value using the Rayleigh distribution with a peak on 0.1. We further limit the eccentricity to 0.4. Rayleigh distribution is used because it favours low values. Higher eccentricates are rare.

Another planetary parameter is inclination. It relates to the angle at which the planet orbits from the perspective of earth. An inclination of 0 means the orbit crosses exactly between earth and the star, in such a perfect case the plants mass can often also be calculated. Usually, astronomers are not so lucky, so we also must choose an inclination. A random value between 0 and 30 degrees is chose, with a high bias towards lower values. A high inclination system has little hope for discovery using the radial velocity method, and as such, inclination above 30 degrees is ignored.

Then we finally choose a random argument of periapsis and phase offset. The latter governs the position of the planet in its orbit, and the other the angle at which the planet is inclined relative to earth. These values are purely random as their real-life counterparts do not depend on anything expect the orientation of the observer.

The final caveat to this generation is the lack of other sources of radial velocity disturbances. Such as moons, which dynamically orbit their planets and in turn disturb their parent star. We also lack dust clouds and asteroid field, which also influence their stars, tough much less than the planets.[23] Calculations of these too would bring little benefit to our study, as the sensitivity of real-life instruments in nowhere near to being able to read motions on the scale of moons and asteroids.

All this allows us to quickly generate realistic star systems.

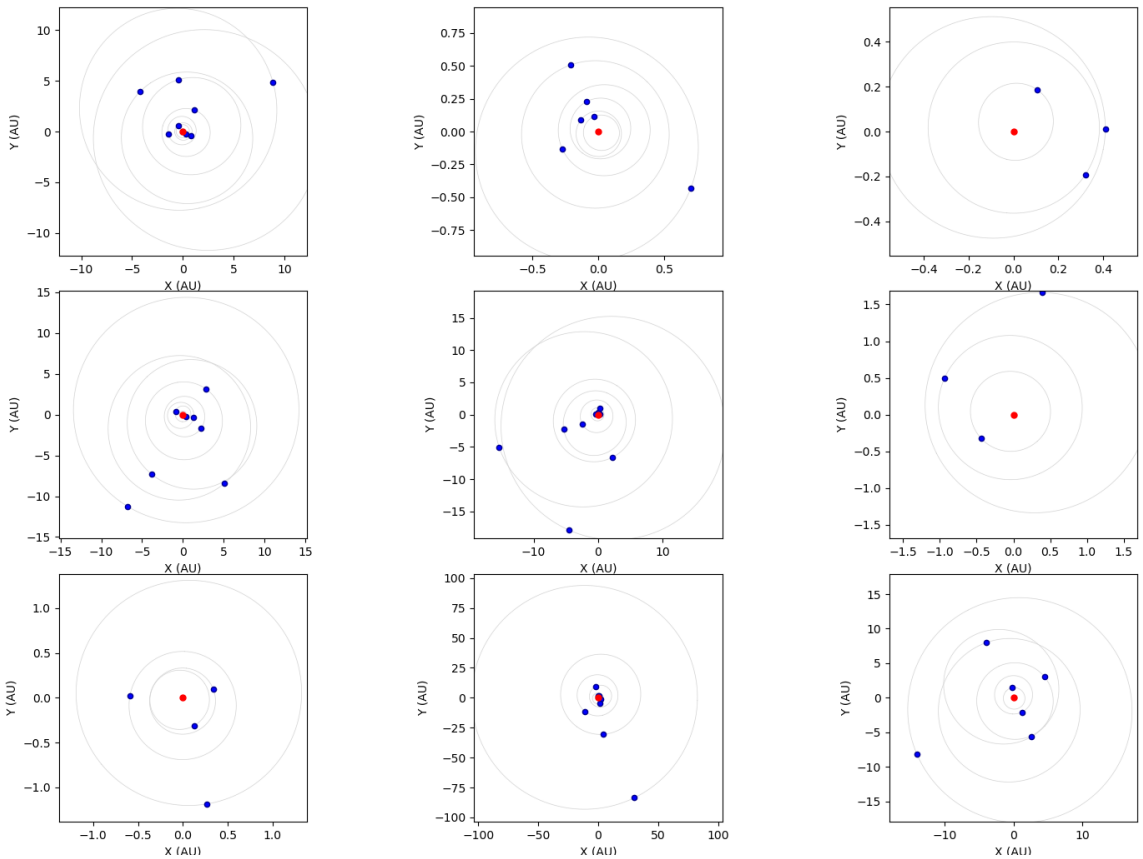

Figure 14 Examples of generated systems

Figure 14 Examples of generated systems shows 9 random samples of generate star system using our main_generate_systems.py script. While the approach is generally successful, we can see cases where planets cross each other's orbits. This is impossible in real life, as such arrangements would cause instability and migration of both planets. This can then be turned into radial velocity using the equation (1) found in A detailed derivation of The Radial Velocity Equation [17]. The code for this is shown below:

```
def calculate_rv(system, time_tensor, device):
    mass_star = torch.tensor(system['star_mass'], dtype=torch.float32,
device=device)
    rv_total = torch.zeros_like(time_tensor)

    for planet in system['planets']:
        mass = torch.tensor(planet['mass'], dtype=torch.float32,
device=device)
        P_days = torch.tensor(planet['P'], dtype=torch.float32,
device=device)
        e = torch.tensor(planet['e'], dtype=torch.float32, device=device)
        w = torch.tensor(planet['w'], dtype=torch.float32, device=device)
        phase_offset = torch.tensor(planet.get('phase_offset', 0),
dtype=torch.float32, device=device)

        M = 2 * torch.pi * time_tensor / P_days + phase_offset
        E = M + e * torch.sin(M)
```

```
        theta = 2 * torch.atan2(torch.sqrt(1 + e) * torch.sin(E / 2),
torch.sqrt(1 - e) * torch.cos(E / 2))

        K = calculate_K(mass, P_days, e, mass_star, device)
        rv = K * (torch.cos(theta + w) + e * torch.cos(w))
        rv_total += rv

    return rv_total

def calculate_K(mass_planet, period, e, mass_star, device):
    G = 6.67430e-11
    G_tensor = torch.tensor(G, dtype=torch.float32, device=device)
    period_sec = period * 86400
    K = (2 * torch.pi * G_tensor / period_sec) ** (1 / 3) * (
            mass_planet / (mass_star + mass_planet) ** (2 / 3)) /
torch.sqrt(1 - e ** 2)
    return K
```

Code 1 Radial velocity calculation code

Code 1 calculates the total radial velocity (RV) of a star influenced by its orbiting planets using their orbital parameters. The calculate_rv function takes a system dictionary containing the star's mass and details of its planets and a tensor of time values and a device for computation (CPU or GPU). For each planet, it computes the mean anomaly and eccentric anomaly, then derives the true anomaly. Using these anomalies, it calculates the radial velocity contribution from each planet based on a derived value K and sums these contributions to yield the total radial velocity at the specified times..

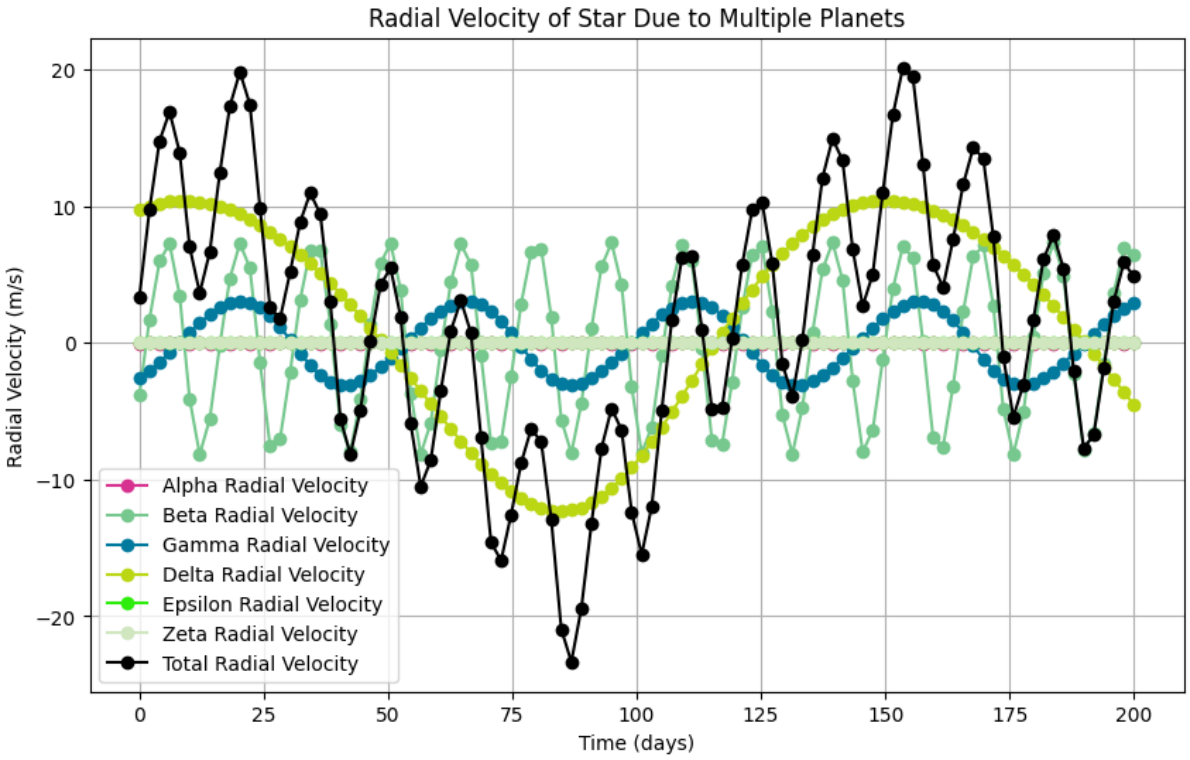

Figure 15 Radial Velocity graph with no noise and evenly distributed sampling

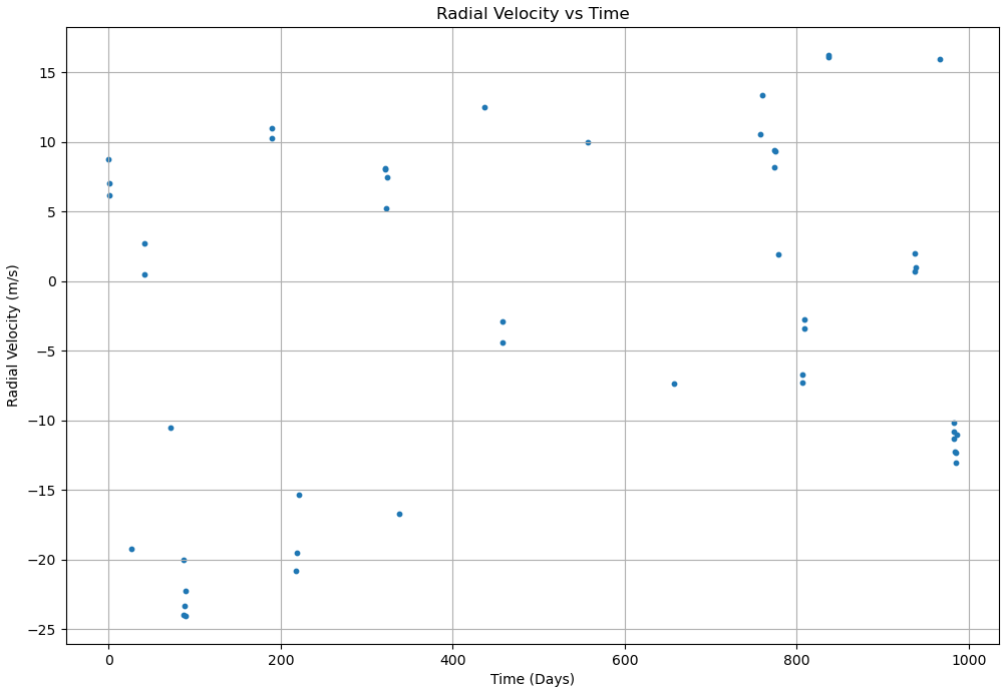

Figure 16 Radial Velocity graph with temporal, stellar, and instrument noise

3.3. Synthetic Radial Velocity Data

Due to the low resolution and highly chaotic nature of radial velocity data, there exists an inscrutable gap between simulated and observational data. Within our simulated environment, we can classify and measure effects of increasingly small source of information, such as miniature planets. While in real-life, we lack such detail. This prohibits us from looking at a simulated system and comparing it to a real one – as we do not have a clear picture of the latter.

Because of this we turn to Kepler's equations of motion, which allow us to approximate our lack of data.

The next step is to generate the radial velocity graph for our generated systems, we do so with the main_preprocess.py script. It will intake our generated planets and create periodograms for each system. Chief among the processes is the generation of various sources of noise. They perturb the ground truth radial velocity from the ground truth case, shown on the left on the image below, to a real-life case, shown on the right.

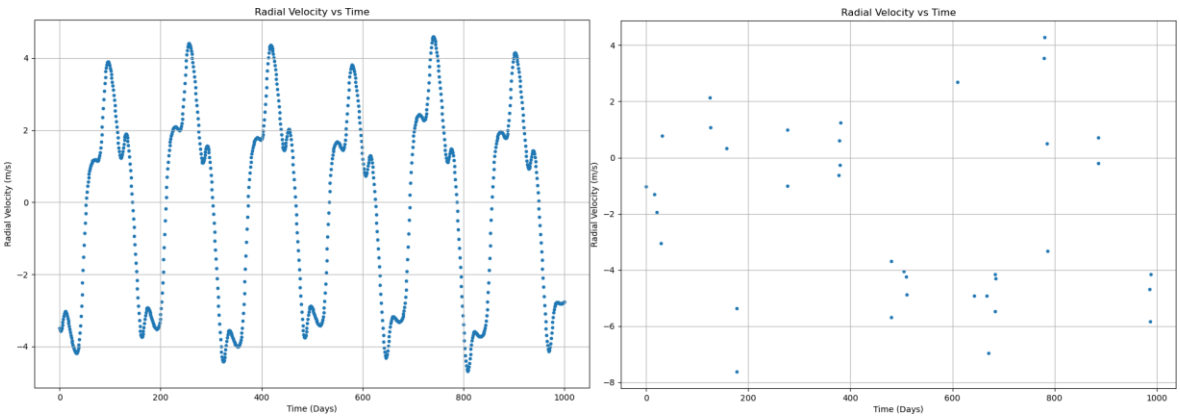

Figure 17 Radial velocity before and after noise

Figure 17 Radial velocity before and after noise shows the decrease in human-readability when we remove observation uniformity.

3.3.1. Temporal Noise

Firstly, we must choose a temporal distribution of observations. Broadly speaking, we can have regular observations intervals, life-like observation interval we have prepared previously, and everything in between.

```
# Generate time array based on the ratio
ratio = random.uniform(0, observation_entropy)
time_array = [0]
while time_array[-1] < time_span:
    if random.random() < ratio:
        # Use time sampler
        delta_time = time_sampler(1)[0]
    else:
        # Use 1 day interval
        delta_time = 1 * random.uniform(0.95, 1.05)
    time_array.append(time_array[-1] + delta_time)
```

Code 2 Observation times generation

We define a random sampling ratio between 0 and 1. Then for each delta time we sample a random uniform value, and if it falls below our sampling ratio, we use our histogram distribution.

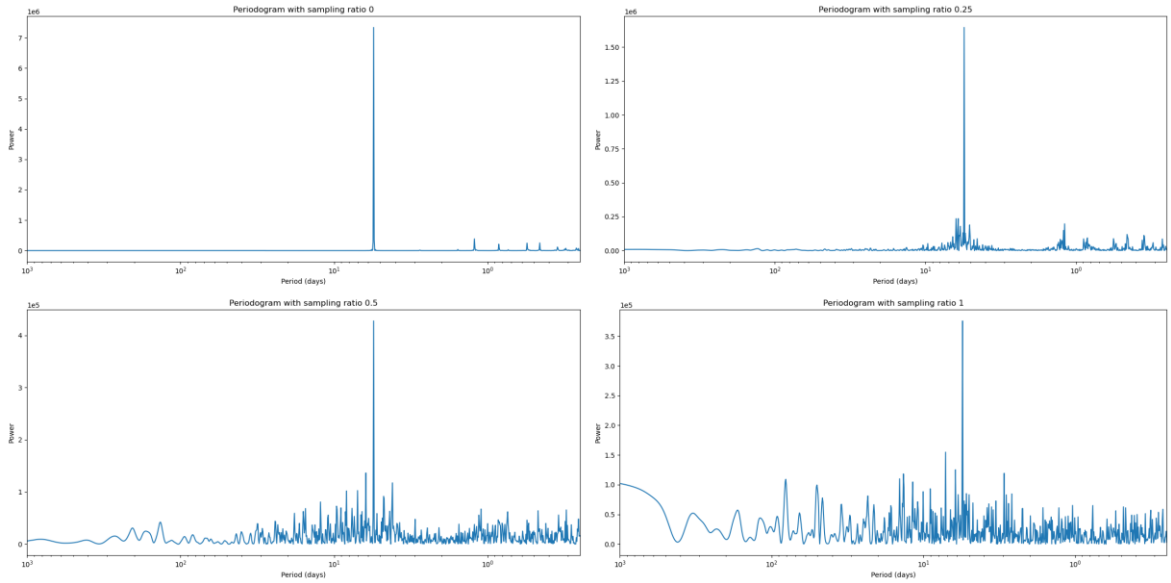

Figure 18 Periodograms with sampling ratios 0, 0.25, 0.5, and 1

Figure 18 Periodograms with sampling ratios 0, 0.25, 0.5, and 1 shows a clear decrease in accuracy due to observational entropy. The less uniform the observation, the more chaotic the resulting periodogram. We generate this array of temporal observations until we reach the maximum observational period, for this we chose a random value based on real-life data shown in the code below:

```
def generate_time_span():
    mean = 1784.78
    std_dev = 1567.18
    min_span = 10
    max_span = 7000

    while True:
        span = random.gauss(mean, std_dev)
        if min_span <= span <= max_span:
            return span
```

Code 3 Observation duration generation

Now that we have an array of our observation times, we can generate the stellar noise.

3.3.2. Stellar Noise

Many factors contribute to the noise generated by the star itself, among those are star-spots. Area of unusually lower temperature compared to the rest of the star. [11] These spots are cooler, and therefor the star emits less blue light, which is associated with higher temperatures. This mimics blue-shift i.e. the star appears to be moving away. Couple this with the periodicity of these spots – they rotate with the star; we have an effect mimicking orbiting planets.

We generate this activity using the approach outlined in the ExoplANNET paper [2]:

$$k_{\mathrm{QP}}(t_i, t_j) = A^2 \exp\left(-\frac{(t_i - t_j)^2}{2\tau^2} - \frac{2}{\epsilon}\sin^2\left(\frac{\pi(t_i - t_j)}{\mathcal{P}}\right)\right) \quad (3)$$

There are four hyperparameters in the equation [2]:

- A – the covariance amplitude sampled using a gamma distribution. Γ(2.0 , 0.5)
- P – the recurrence time i.e. the rotational period of the star. Uniformly sampled between 10 and 40.
- τ – decay time, the average lifetime of an active region. Normal distribution dependant on the recurrence time. N(3*P, 0.1*P)
- ϵ – structure factor, associated with the count of active regions. Uniform distribution between 0.5 and 1.

The code below has been created for this purpose:

```
def generate_stellar_noise_gpu(time_tensor, device):
    t = time_tensor.to(device)
    n_points = len(t)
    time_span = t[-1] - t[0]

    # Check if time array is strictly increasing
    if not torch.all(t[1:] > t[:-1]):
        # Force strict increase by adding a small increment
        t = torch.cumsum(torch.abs(torch.diff(t,
prepend=torch.tensor([0.0], device=device))), dim=0)

    mean_uncertainty = 1.0
    uncertainties = torch.normal(mean_uncertainty, 0.3, (n_points,),
device=device)
    uncertainties = torch.clamp(uncertainties, min=0.5 *
mean_uncertainty)
    intrinsic_errors = torch.normal(0, uncertainties)

    freqs = torch.linspace(1/time_span, 10, n_points, device=device)
    pulsation_power = 1 / (1 + (freqs/0.1)**2)
    granulation_power = 10 / (1 + (freqs/0.01)**2)
    total_power = pulsation_power + granulation_power
    pulsation_granulation = torch.normal(0, torch.sqrt(total_power))

    P = torch.rand(1, device=device) * 30 + 10  # Uniform between 10 and
40
    A = torch.distributions.Gamma(torch.tensor([2.0]),
torch.tensor([0.5])).sample().to(device)
    tau = torch.clamp(torch.normal(3*P, 0.1*P), min=P*0.1)  # Ensure tau
is positive and not too small
    epsilon = torch.rand(1, device=device) * 0.5 + 0.5  # Uniform between
0.5 and 1.0

    K = quasi_periodic_kernel_gpu(t, A, tau, epsilon, P)

    # Scale the matrix to improve numerical stability
    scale = torch.max(torch.abs(K))
    K_scaled = K / scale

    # Add a larger regularization term
```

```
    K_scaled += torch.eye(n_points, device=device) * 1e-3

    try:
        L = torch.linalg.cholesky(K_scaled)
        rotational_modulation = torch.matmul(L, torch.randn(n_points,
device=device)) * torch.sqrt(scale)
    except RuntimeError:
        print("Warning: Cholesky decomposition failed. Using diagonal
approximation.")
        rotational_modulation = torch.normal(0,
torch.sqrt(torch.diag(K)))

    total_noise = intrinsic_errors + pulsation_granulation +
rotational_modulation
    return total_noise.cpu().numpy()
```

Code 4 Generating stellar noise

```
def quasi_periodic_kernel_gpu(t, A, tau, epsilon, P):
    diff = t.unsqueeze(1) - t.unsqueeze(0)
    return A**2 * torch.exp(-(diff**2) / (2*tau**2) - 2/epsilon *
torch.sin(torch.pi*diff/P)**2)
```

Code 5 Quasi periodic kernel

Code 4 and the function shown in Code 5 requires to GPU to be run and each system takes around ~0.75 seconds on NVIDIA RTX 3050 Laptop GPU.

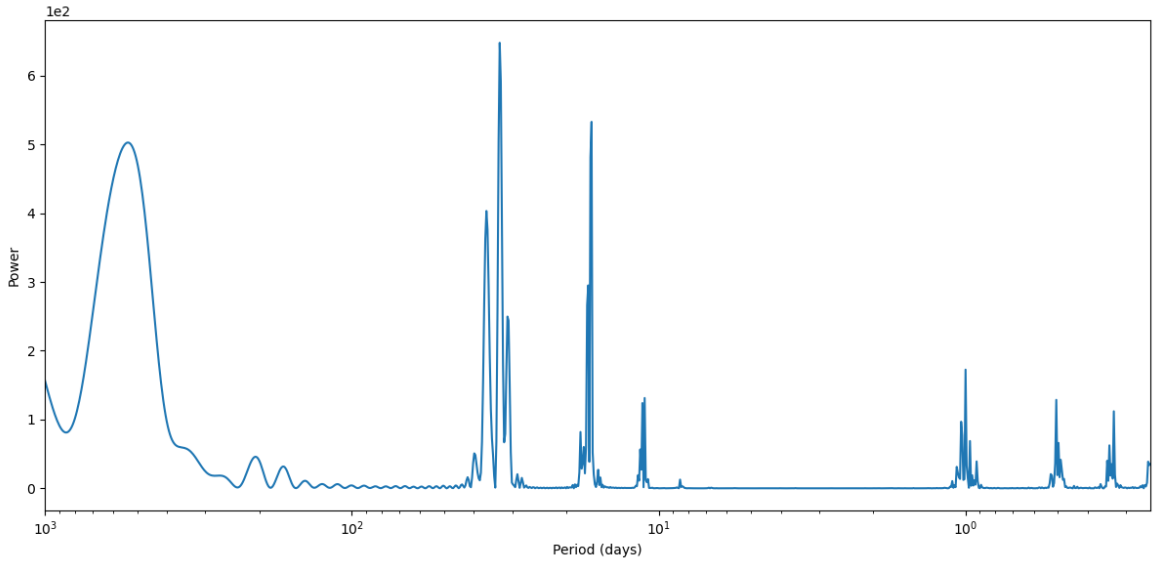

Figure 19 Periodogram of stellar noise – no planets or other noise sources

On Figure 19 Periodogram of stellar noise – no planets or other noise sources we see a clean periodogram with just stellar noise. In practice we never expect to see such a clean periodogram, due to other noise sources and the certain existence of objects orbiting each star.

3.3.3. Intrinsic Errors

Another class of errors is the cumulative and unknown errors from many sources, such as temperature variation within observation instrument, interstellar dust, and atmospheric disturbances. [2] These errors are defined as having a mean of 1, and a standard deviation of 0.3.

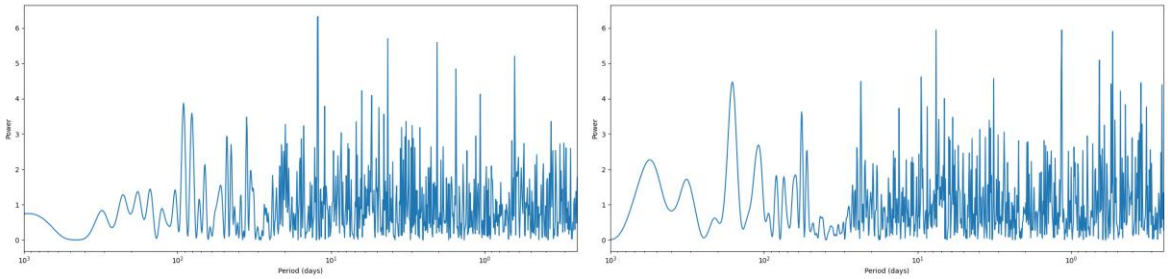

Figure 20 Two examples of periodogram with just intrinsic errors

Figure 20 shows the incredibly chaotic periodograms of intrinsic error, expected since they are sampled from a normal distribution.

3.3.4. Pulsation and Granulation

This final source of noise seeks to capture short-term variations using stellar pulsation, and long-term changes with granulation. These two values are used together to perturb the ground truth frequencies within the radial velocity. They are defined as:

```
freqs = torch.linspace(1/time_span, 10, n_points, device=device)
pulsation_power = 1 / (1 + (freqs/0.1)**2)
granulation_power = 10 / (1 + (freqs/0.01)**2)
total_power = pulsation_power + granulation_power
pulsation_granulation = torch.normal(0, torch.sqrt(total_power))
```

Code 6 Pulsation and granulation implementation

Where freqs is an array of frequencies within the observation period, starting at 1/time_span and ending at 10 Hz.

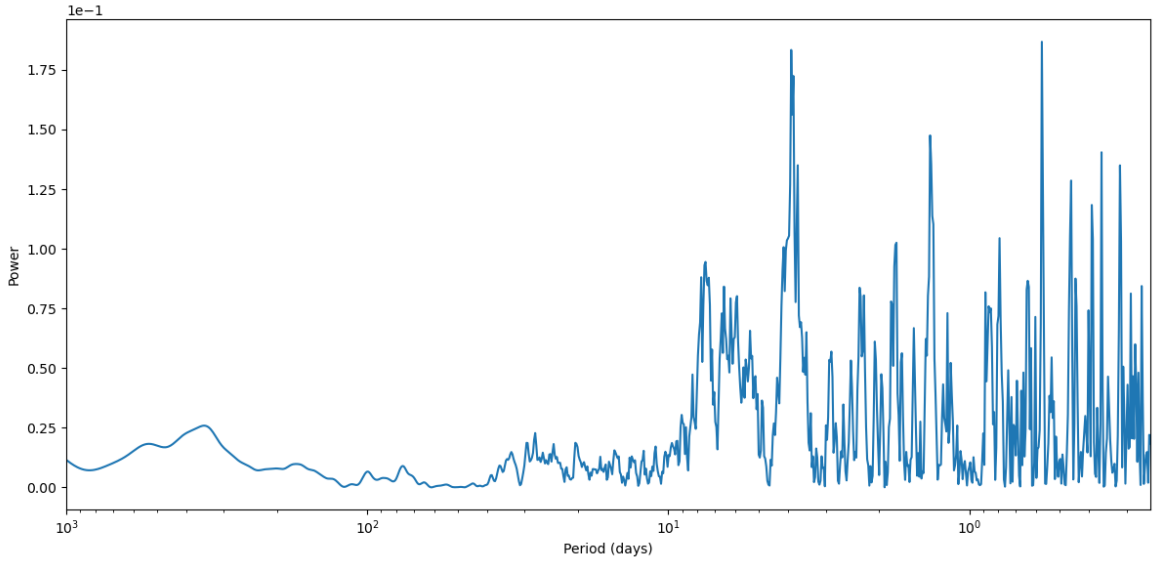

Figure 21 Frequency Pulsation and Granulation

Figure 21 shows a chaotic but small periodogram noise generated die to pulsation and granulation.

3.4. Data Preprocessing and Feature Selection

The most important factor in exoplanet detection in radial velocity is the inherent periodicity within the signal, and as such we have prepared the periodograms as inputs for our models. Now we must normalize them and do further feature selection.

3.4.1. Peak Selection

An emergent feature of a periodogram is the appearance of sloped peaks around certain frequencies. Astronomers identify these peaks and infer the existence of planets using their power and overall periodogram composition. [6] When preparing our data, we must be careful to choose peaks that represent the distribution of candidate exoplanet peaks, while also providing enough examples of true negatives.

To this end, we prepare a function that selects the indices of all peaks by first finding their peak then masking surrounding frequencies which belong to that peak. We define “belonging” as having a power which is smaller than the power of the previous frequency.

We mask all these frequencies so that we don’t return to them when choosing other peaks. In the end we get a list of all peaks within the periodogram sorted by power.

By comparing the frequencies of these peaks and the periods of the ground-truth planets we can ascertain which of these peaks are in fact planets. We add all of them to the list of planetary peaks.

Next, we select N random high-power peaks which do not belong to a planet. These will serve as true negatives that help tech that model not all high-powered peaks are caused by planets.

Finally, we must account for all the low-powered planets within the data. Planets such as the one highlighted with an arrow:

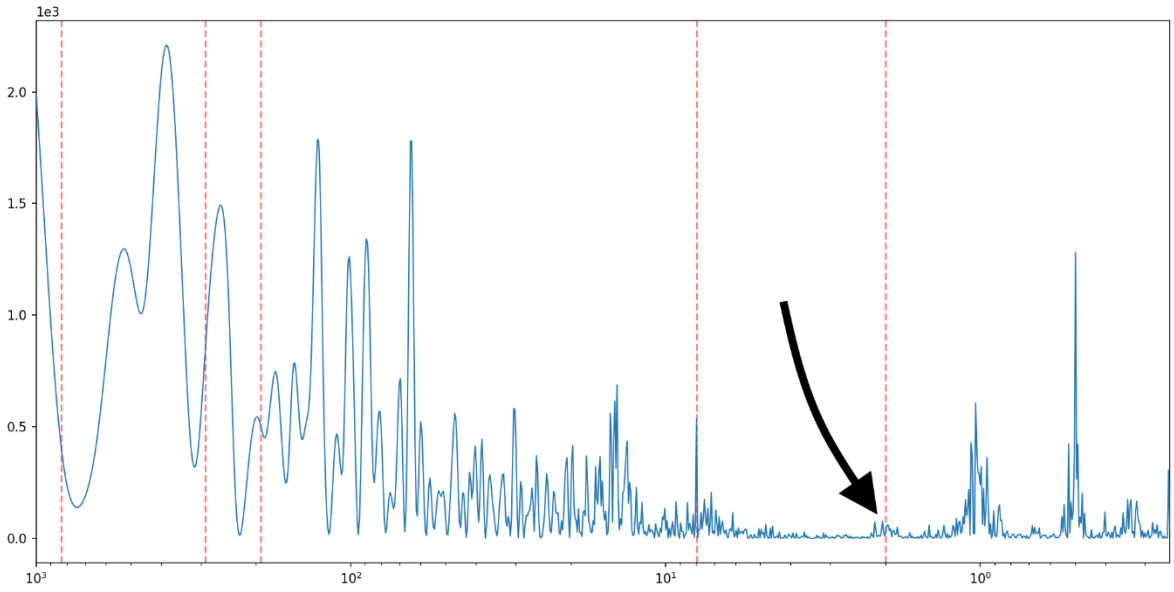

Figure 22 Low-power planets

Figure 22 shows planet peaks (highlighted with red lines) in an example periodogram. Here we can also see that often the exact period of the planet does not correspond to any peak within the periodogram. Mostly due to irregularities caused by the join effect of temporally uneven observations and intrinsic errors.

Therefor we say a planet belongs to a peak if its true period is within 10% of the peak's frequency. The model at the end will not learn if there is absolutely a planet at the selected frequency, only if there is any planets within 10% of the value.

On the far left we see two planets whose long periods and relatively low mass cause substantial errors in the periodogram. There is little hope of detection for these planets, unless more temporally consistent observations are made.

As for the low-powered short-period planets, their true negatives are chosen at random from the upper 75% of the peaks – ignoring all low powered noise-induced peaks. During all this, we are careful to keep away from the 10% limit around true planetary periods.

In the end, we may get something like this:

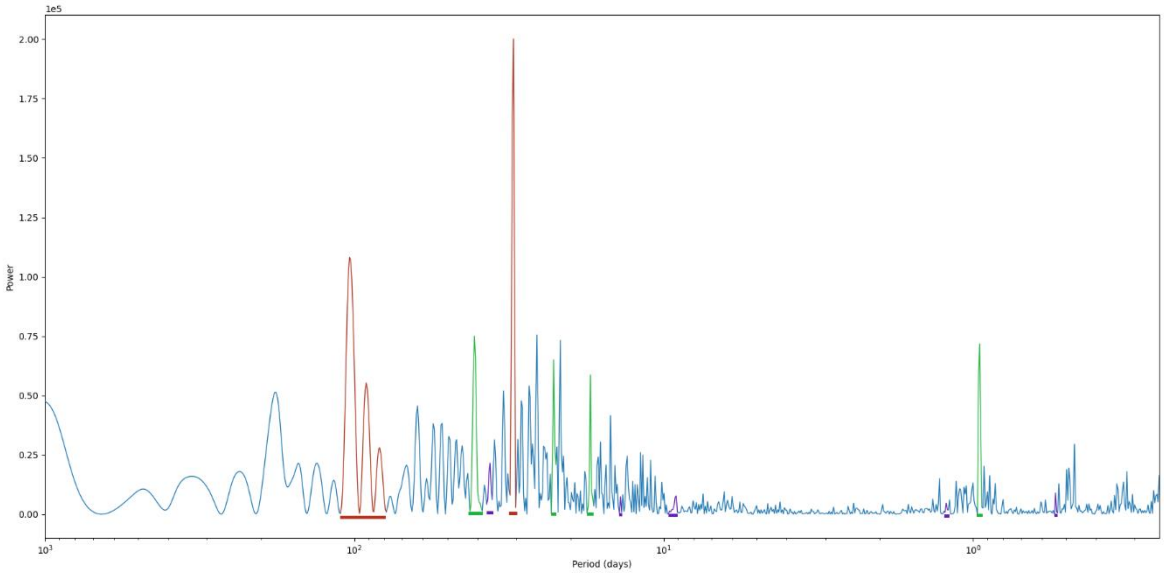

Figure 23 Four selected planetary peaks (red), four random high-power peaks (green), five random low-power peaks (purple)

These peaks are labelled accordingly, and their surroundings extracted into a separate 61-length array. It contains the peak at the centre, and the surrounding frequencies. Padding is done to maintain constant length. The peak is normalized to a range from 0 to 1, and its peak frequency recorded alongside its power.

3.4.2. Periodogram Normalization

The values of frequency within the periodogram are static, ranging from 6 hours to 7000 days. A period of 7000 days would represent the very limit of our detection capabilities.

The power of each frequency, i.e. the strength of its presence within the periodogram can range from less than 1 to 10e8 so while we normalize this range to a between 0 and 1, we must also provide the absolute value of the peak. We do this by logarithmically normalizing the power of the selected peak and preparing it as an additional input feature for the model. Effect of this is shown on Figure 24:

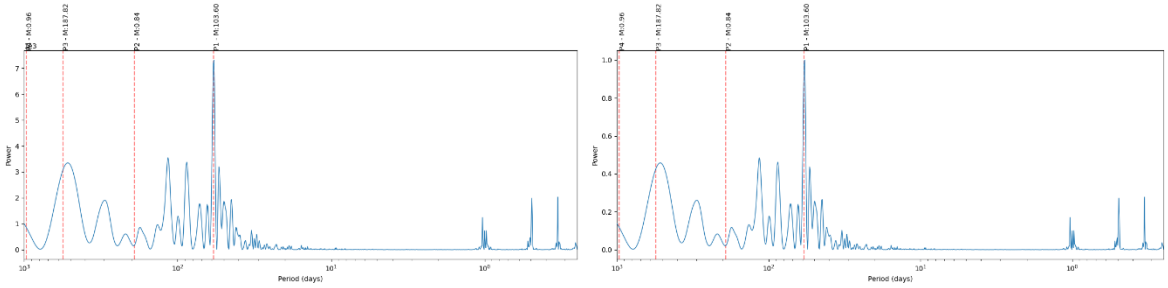

Figure 24 Normalized periodogram

4. Model Design and Training

The provided features are as follows: The periodogram of the entire planetary signal, the mass of the star – normalized to solar masses, and the peak frequency and power – labelled as planet or not.

These three separate inputs must be processed separately by the branches of the model, before their outputs are combined into a series of dense layers which make the final decision on the existence of planets within each selected peak.

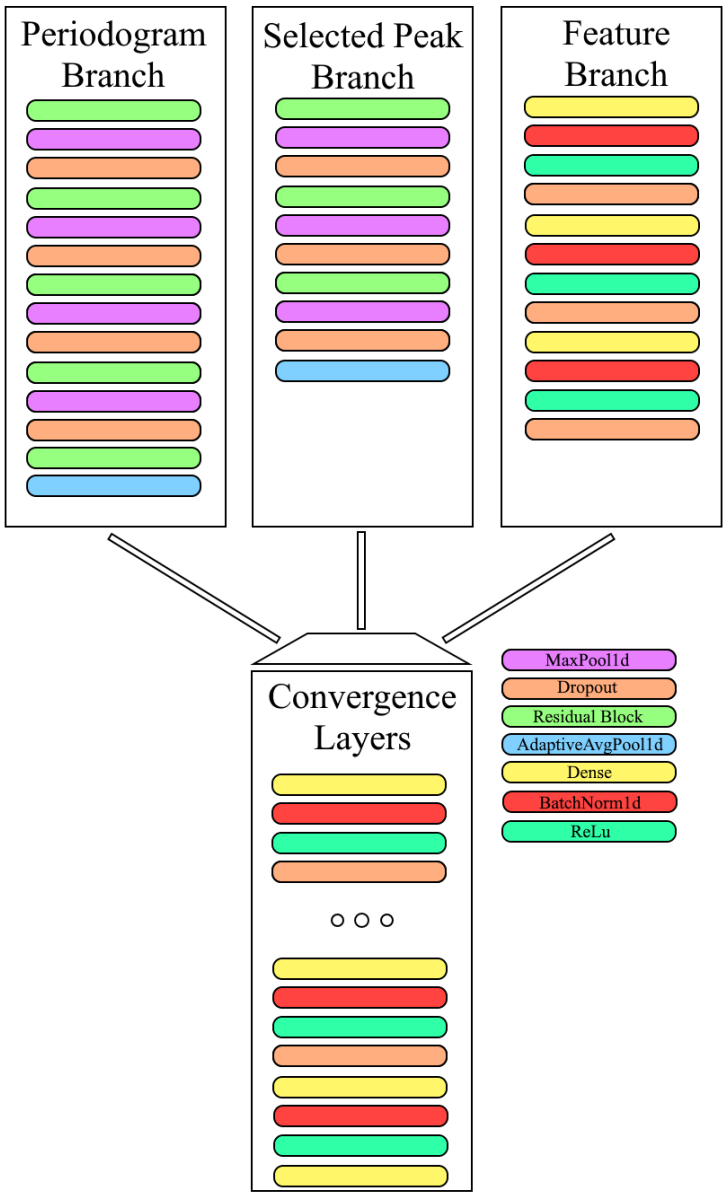

Figure 25 Overview of the model

4.1. Periodogram Branch

The Periodogram Branch of our model is responsible for processing the frequency and powers of our data. Its abstract task would be to learn the overall structure of the star system that is being considered, and as such this branch would need to learn some representation of planetary periodicity.

The input powers are scaled according to our preprocessing steps and passed on into the periodogram branch.

Within the periodogram branch we employ the use of residual blocks. Each consists of two convolutional layers, batch normalization, and a shortcut connection to the input of the residual block.

Residual blocks bring their input though a shortcut connection all to way to the output. Therefore, it allows the gradient during training to cross freely though the architecture. Since our model is of moderate depth, it helps with training.

The shortcut path contains a simple convolution which works to align the dimensions of the input and output.

After each residual block we employ a max pooling layer and a dropout layer:

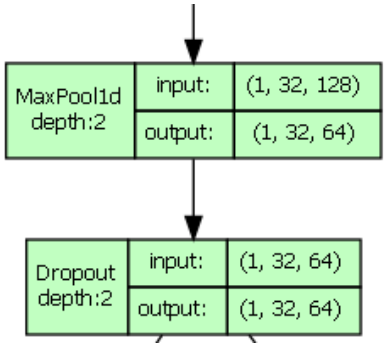

Figure 26 Max pooling and dropout

Dropout layers randomly set a percentage of neurons on their input to 0, and pass the rest of the neurons unchanged to their output. This forces the neural network to learn information from the overall representation of neuron values, instead of relying on a couple of high-importance neurons. This prevents overfitting, where the model learns the noise instead of the general representation of data. A dropout rate of 10% has been maintained though the entirety of the periodogram branch.

Table 5 Hyperparameters of the ResBlocks in the periodogram branch

ResBlock	Input channels	Output channels	Kernel	Padding
1	2	32	3	1
2	32	64	5	2
3	64	128	7	3
4	128	256	9	4
5	256	512	11	5

We have chosen an increasing number of parameters for the ResBlocks due to their inherent ability to learn more complex features with depth. The first layer starts of with two input channels, one for frequency and one for power of the input periodogram. This gets increased to 32 channels by the two convolutional layers within the block. By increasing the kernel and the padding, we are explicitly instructing the model to learn features of increasing size.

4.2. Selected Peak Branch

The purpose of this branch is to process the specific selected peak and learn its features and shapes, so that we may later decide if it is significant enough to classify as a planet. This branch is mostly identical to the periodogram branch save for the number of residual blocks and their hyperparameters. As the peak branch only needs to learn the representations of a single peak, its size is justified in being much smaller.

Table 6 Hyperparameters of the ResBlocks in the selected peak branch

ResBlock	Input channels	Output channels	Kernel	Padding
1	1	16	3	1
2	16	32	5	2
3	32	64	7	3

An additional hyperparameter relating to the peaks is the count of datapoints surrounding the peak. By visual analyses it was discovered that a value of 30 is sufficient. This value selects the peak power and adds 60 more surrounding frequencies to the peak – 30 on each side – bringing the total to 61 points.

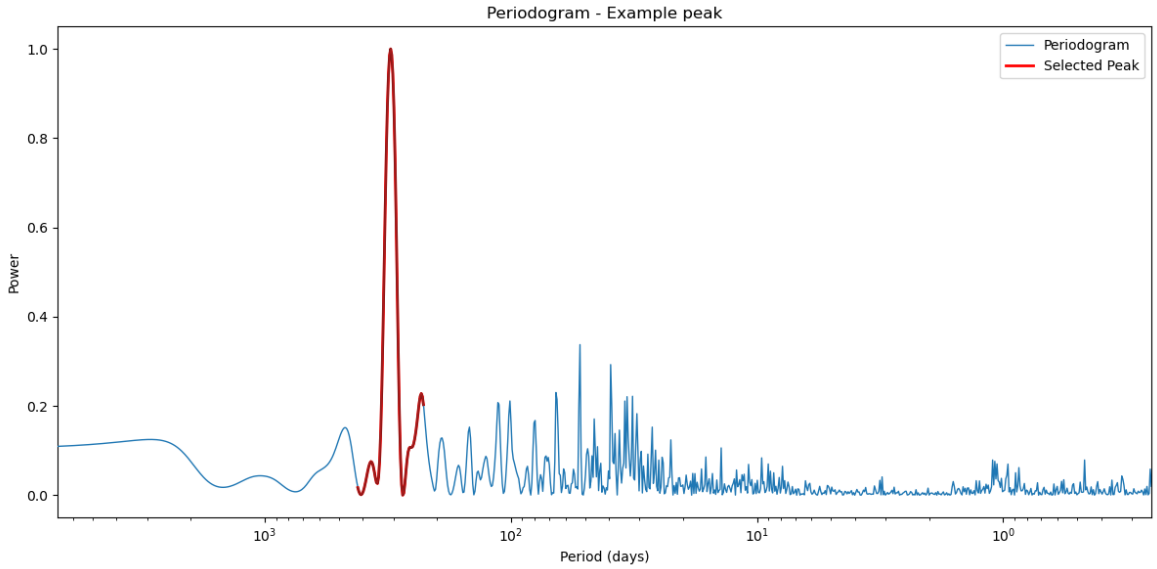

Figure 27 Example selected peak with 61 points centred on peak power

Figure 27 shows the selected peak and its surrounding points. The selected peak branch only has the normalized power values on its input – and normalized to the maximum of the peak itself, not the periodogram as a whole. The frequencies are not present in the input, they are implied by the sequential nature of the power values. By doing this we are forcing this branch to focus solely on the shape of the peak and its neighbourhood. The actual power and frequency of the peak are also relevant, and are provided to the model in the final branch.

His branch also contains the max pooling layers and dropout layers se to 10%.

4.3. Feature Branch

The feature branch ingests the additional features relevant to the periodicity of the signal – these are the frequency and power of the peak. Supposedly star mass could be added here too, distance to the observed star system, the coordinates in the galaxy, the type of observatory and its location. These features could provide additional insights to the model, but since we lack the data to generate them convincingly, they have been left out.

This branch contains densely connected layers connected with batch normalization, ReLu and dropout layers. The dropout layers are set to 20%, which is higher than the periodogram and selected peak branches. The function of this branch is to prepare the feature information for processing by the convergence layers, so this branch is the smallest of the three.

4.4. Convergence Layers

The outputs of the three branches are joined together and passed into a series of densely connected layers which will make the final decision on the presence of an exoplanet within the periodogram data. The input size of the first dense layer is 640, which matches the 512 output neurons of the periodogram branch, the 64 neurons of the selected peak branch, and the 64 neurons of the feature branch.

Table 7 Input and output dimensions of dense layers in the convergence branch

Dense Layer	Input	Output
1	640	512
2	512	512
3	512	256
4	256	256
5	256	64
6	64	6
7	6	1

As the dimensions of the dense layers decrease, the information becomes condensed into smaller and smaller representations. The dropout rate between these layers also decreases from 20% at the start to 0% between the last two layers. At the end we apply a sigmoid activation function which limits the output to a range between 0 and 1. These final values represent the model's confidence in the presence or lack of a planet within the selected peak.

4.5. Architecture choices

Due to the nature of the periodogram, where the relationships between powers found within are the thing that holds information, convolutional layers were chosen. They have proven to me most effective with learning relationships between data and utilizing the available compute.

We chose the use residual blocks due to their ease of training and better performance with rising depth [24].

During the model design process several other architectures and layer configurations were tested. By process of elimination, this final architecture was chosen, as described in the previous section. Here are the most relevant things tested, their impact, and the reason they were discarded;

- Basic Convolutional layers instead of Residual Blocks: Simpler and faster to compute and train. During testing they provided marginally worse results.
- Bottleneck Block. Similar to Residual blocks but they reduce the input before applying larger kernels. They provided worse results, mostly due to them being useful only with deeper networks. [24]
- Average Pooling or Strided Convolution instead of Max Pooling: these can provide better performance with smoother feature learning. Strided Convolution also has more learnable parameters. However, during testing neither performed better than Max Pooling.
- SpatialDropout1d: This could replace Dropout. However, during testing, it took significantly longer to achieve the same performance. Therefore, it was dropped due to the increased training time, which would impact the already lengthy training runs.
- DropConnect can be useful with deeper layers instead of using Dropout [25]. Didn't provide improvement to our model, likely due to our model's smaller size.
- GlobalMaxPool1d instead of AdaptiveAvgPool1d. Would keep only the globally maximal values. As this is the final layer in each branch it is unwise to remove information from the network at this stage. Testing confirmed this, and AdaptiveAvgPool1d was kept.
- LeakyReLu and Swish instead of ReLu. LeakyReLu and Swish did not provide noticeable improvement during testing.

Other macroscopic architecture choices were:

1. Long Short-Term Memory (LSTM). Since we transform a graph of sequential data in the form of radial velocity measurements, into a static periodogram, LSTM lose their main advantage – their ability to capture temporally distributed information. During early model design, they were tested using raw radial velocity data – they were unable to achieve useful learning.
2. Temporal Convolutional Networks (TCNs) were too large to fit on the available hardware.

3. Transformers. Also tested on raw radial velocity data. Their training times were too large to be useful for experimentation, upwards of 4 hours. Smaller sized did not learn effectively.

5. Training

The model was set to train on 30 000 synthetic star systems which yielded 365 756 datapoints for training. 18.86% of these were real planetary signals, while the rest were a mix of high-powered peaks, random low-power peaks, and purely random peaks, all true negatives.

The data was split into training and test with a test size of 0.2. Training was set to last 250 epoch with patience set to 15 epoch with regard to the F1 score on the validation set. If the F1 score didn't improve for 15 epochs, the training would stop and the highest F1 score epoch would be presented as the final model. Each epoch was also saved to maintain reproducibility of the F1 scores.

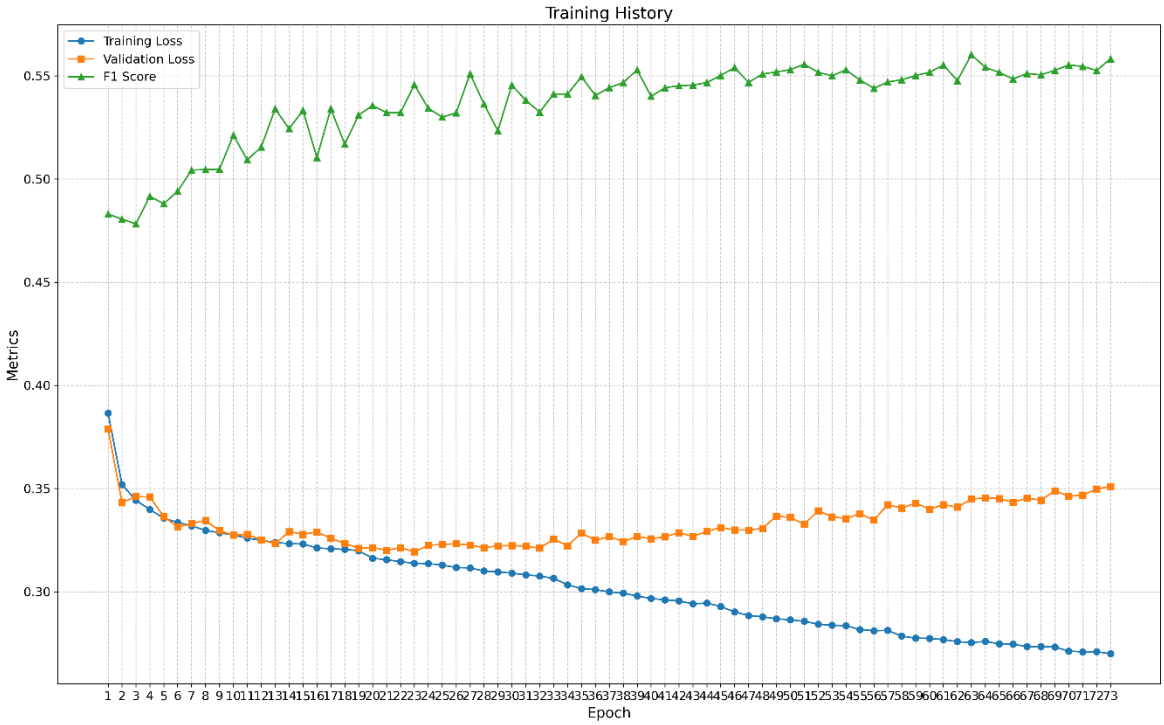

Figure 28 Training graph

Figure 28 show our training run, in the end, training was manually stopped after overfitting was observed. Curiously, the F1 score has continued to improve on the validation set even in the overfitting regime. And by testing epochs between 23rd and 60th the 51st epoch was chosen as the best.

The training took 109 minutes with each epoch taking ~1.5 minutes.

6. Evaluation

We prepare an evaluation script for the model with the aim of simulating a data pipeline astronomers might use once they procure the radial velocity data of real stars. This includes converting the data into a periodogram, and then selecting n top peaks within the signal. There are type types of data that can be used to evaluate the model. Real data and synthetic data we have generated in previous sections. For real data, a fair amount of manual data preparation is necessary, so first we take a look at synthetic evaluation.

6.1. Synthetic Evaluation

Temporal sampling, the distribution of delta times between subsequent observations greatly influences the look and quality of the periodogram. Another hyperparameter that comes into play here is the detection threshold – the value we consider to be the minimum required to classify a peak as a planet. The heatmaps below have been generated on ~1000 synthetic star systems. Daily observations represent a daily observation cycle, where we observe each simulated star once per day – this yields higher resolution data. This improves the model performance. Real-life observations mimic the chaotic observation cycles of existing exoplanet data. The observed increase in the F1 score is significant, as it underscores the increase in performance with cleaner data. It also shows that with 3 peaks we maintain state of the art performance on recall.

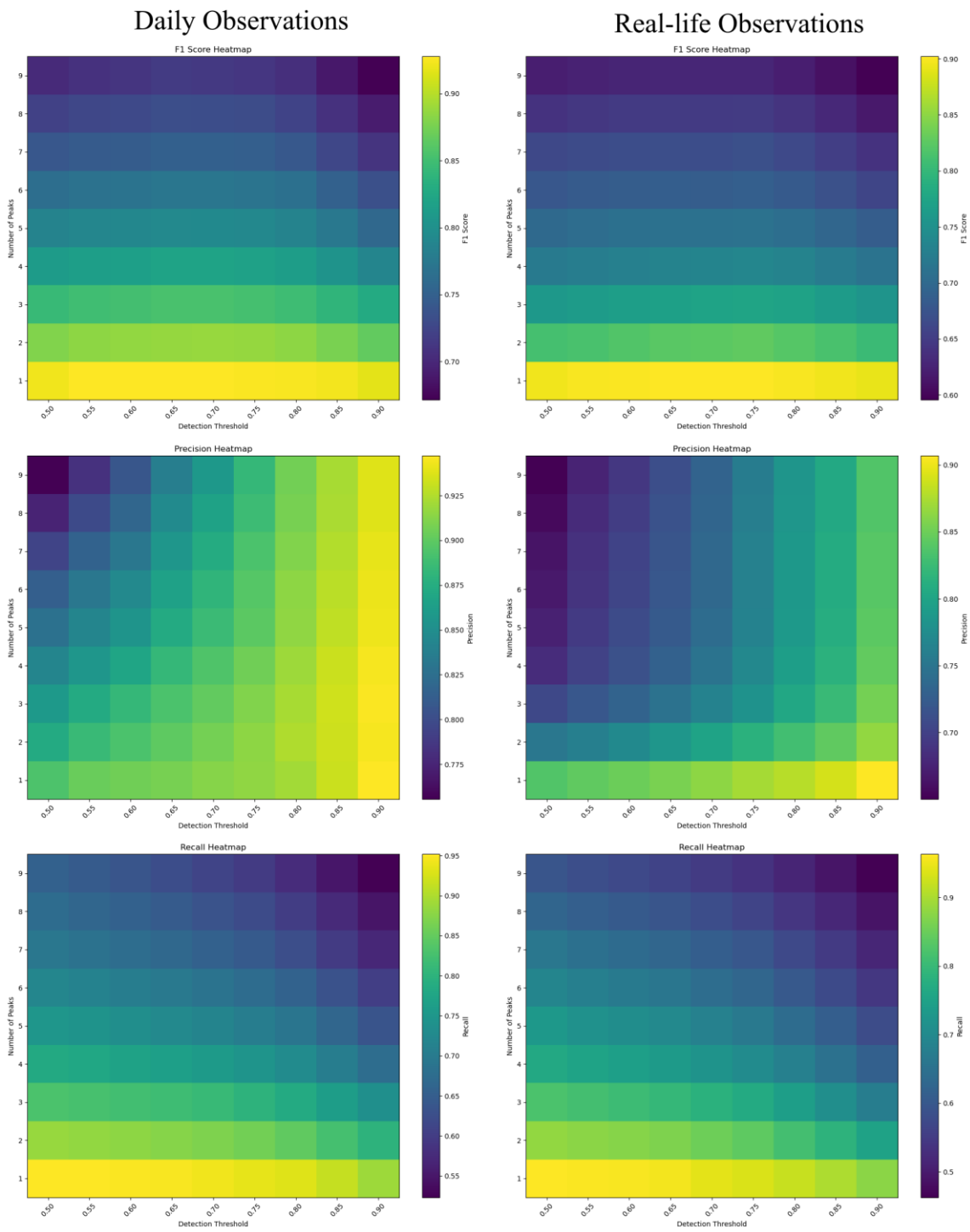

Figure 29 Heatmaps of number of peaks vs detection threshold

Figure 29 shows the heatmaps for pairs of number of peaks and detection thresholds, for both daily observations and chaotic observations. The current state of the art in exoplanet detection via periodograms and machine learning [2] achieves a recall of 0.741, a precision of 0.934, and a F1 score of 0.8264. These values are achieved on evenly sampled data i.e.

daily observations and the peaks are selected using a “virtual astronomer” whereby the authors select significant peaks which get removed from the signal and the process repeats.

Our model naively selects top n peaks and achieves a F1 score on par with exoplANNET at n = 3 peaks with a detection threshold of 0.73. At these hyperparameters we get the following metrics and confusion matrix at 9893 evaluation systems:

Table 8 Metrics naive approach

	exoplANNET	Our work
Precision	0.934	0.935
Recall	0.741	**0.78**
F1 score	0.826	**0.851**

Table 9 Confusion matrix at n = 3 and threshold = 0.73

True Negatives: 14394	False Positives: 777
False Negatives: 3182	True Positives: 11326

The beauty of this machine learning oriented approach is the ability to tweak hyperparameters according to our desired behaviour. Let’s take a look at how the confusion matrix changes with the detection threshold:

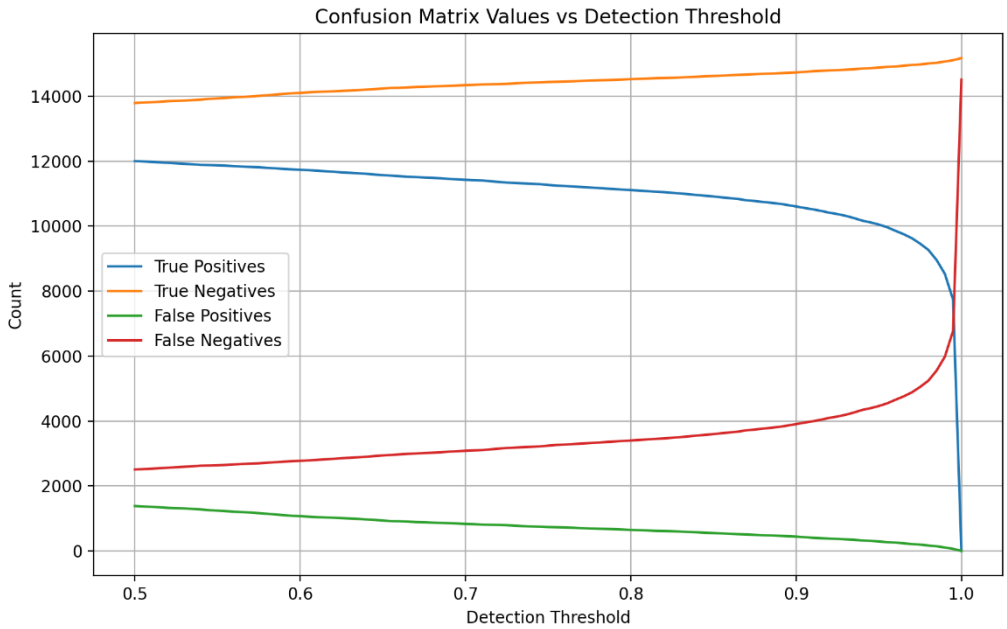

Figure 30 Confusion matrix values against detection threshold at n = 3

Figure 30 shows how our confusion matrix changes with the detection threshold. Based on model requirements the network can be set up to be biased against false positives, which are in our case quite costly. It would be more convenient to miss an exoplanet detection that it would be to claim a planet where there is none. Another point to consider is that the radial velocity measurements are often dictated by existing measurements of the star. If an astronomer detects a clearly visible signal within the first dozen observations, they are more likely to request subsequent observation time. These observations would then be done in quick succession, and the planet within the signal, likely a large short-period object to be detected easily, would be classified as a discovery.

Our current pipeline does not take into account such a scenario – where the detection itself becomes more or less likely depending on the mass and period of the planet, and the mass of the star. We can see this discrepancy clearly here:

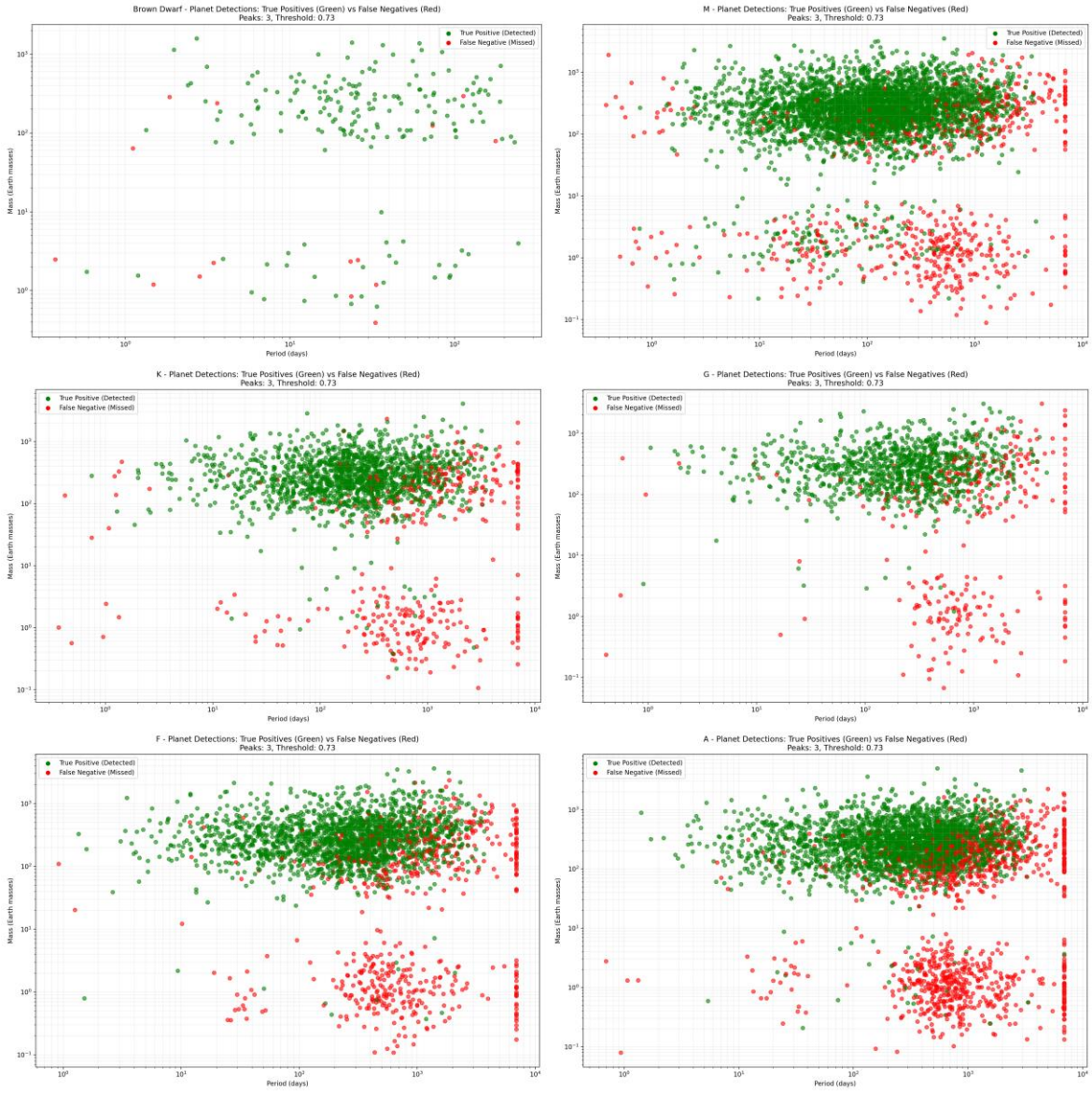

Figure 31 Detections based on star mass

Figure 31 has been generated based on star mass. As the mass of the star increases the change of detecting low mass and higher period planets decreases. This is expected, as the force with which the planet pulls at the stars is proportional to the mass of the planet, and inverse proportional to the distance. Also of note are the two vertical groupings at the 7000-day period mark on the right of each graph. These are remnants of system generation.

6.2. Observational Data

After downloading and preparing the radial velocity data from the NASA exoplanet archive [2], we have procured 552 unique star systems which contain planets. Then we use the NASA exoplanet archive API to match the names of the stars to all confirmed planets. We

select only those discovered using the radial velocity data. By doing this we have discarded 230 planetary systems due to missing data. In the end we get this confusion matrix at n = 3 and threshold = 0.73:

Table 10 Confusion matrix of real data with n = 3 and tr = 0.73

True Negative: 722	False Positive: 10
False Negative: 183	True Positive: 51

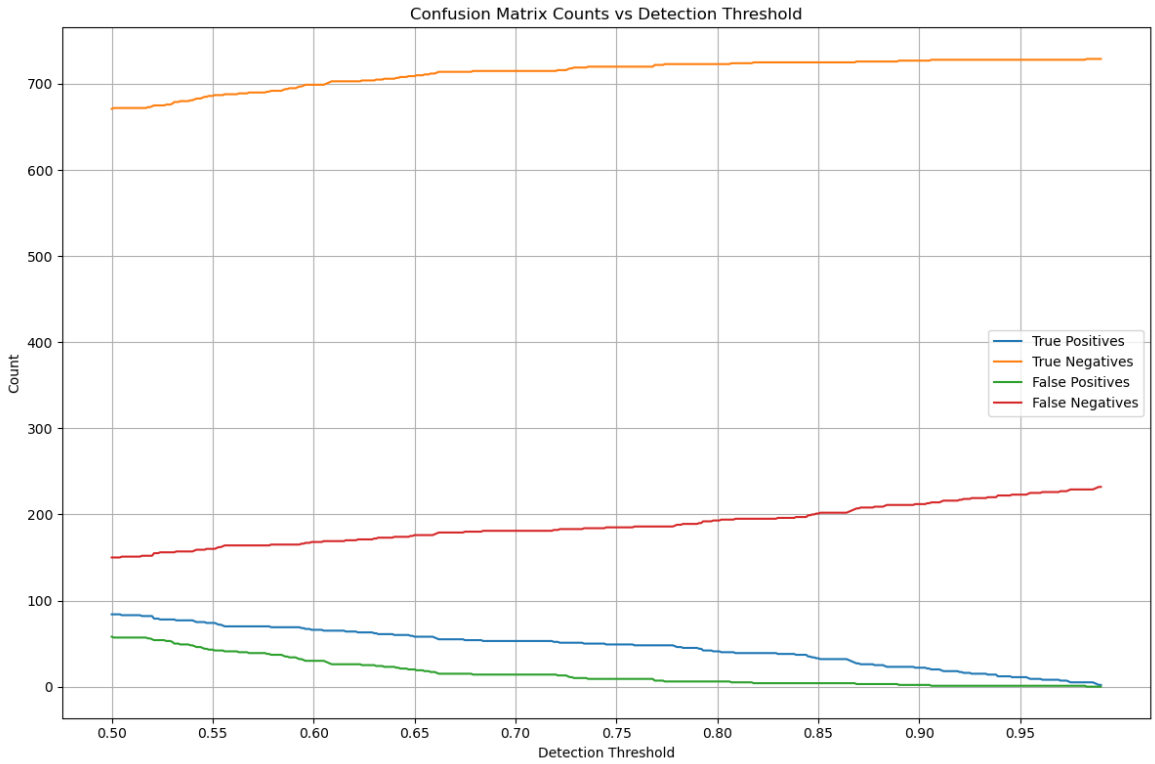

Figure 32 Confusion matrix vs detection threshold at n = 3

The resulting precision equals 0.8361 with a recall of 0.2179. We mostly discover high mass planet and those with a large number of evenly spaced observations.

7. Conclusion

I have developed an algorithm that can generate realistic star system compositions. The code provides masses of stars which are within currently detectable mass ranges for radial velocity, while maintaining enough randomness to generalize to most star systems. A noise algorithm for stellar activity was also developed, which randomly assigns periodic disruptions to the radial velocity signal.

An inherent drawback, due to computational constraints, is the lack of simulated influences of relativity. In a more accurate simulation, the star system would go through a period of stabilization where n-body physics would act in a virtual 3D environment. Time in this simulation would move planetary orbits towards an equilibrium, which would then be reflected in the periodogram data and would allow the model to regress to some deeper understanding of planetary orbits.

I have created a machine learning model which works by processing a selected peak from the radial velocity periodogram, which is filled with stellar and instrument noise. The model is split into three branches, the first processed the entire 1000 point wide periodogram. The second contains just the selected peak, and the last intakes the true values of the power and frequency of the periodogram. The outputs of these branches are processed by a series of dense layers which output the final probability of the planet within the selected peak.

Trough evaluation, the model was found to match state of the art in precision and has achieved a higher recall rate i.e. a lower number of false negatives. When evaluating or real data procures from the NASA exoplanet archive, the model maintains a high precision but suffers in recall.

The model presents promise when it comes to executing a large search across millions of stars. But further work should focus on relativistic effect of gravity within planetary systems and n-body physics. The simulation of stellar noise while accurate so some degree, would benefit from higher accuracy inherent to deeper physical simulation.

Literature

[1] HADDAWAY, N. R., Page, M. J., Pritchard, C. C., & McGuinness, L. A. (2022). PRISMA2020: An R package and Shiny app for producing PRISMA 2020-compliant flow diagrams, with interactivity for optimised digital transparency and Open Synthesis Campbell Systematic Reviews, 18, e1230. https://doi.org/10.1002/cl2.1230

[2] ExoplANNET: A deep learning algorithm to detect and identify planetary signals in radial velocity data. L. A. Nieto and R. F. Díaz. June 2023

[3] Radial Velocities as an Exoplanet Discovery Method, Jason T. Wright, July 2017

[4] Identifying Exoplanets with Deep Learning. IV. Removing Stellar Activity Signals from Radial Velocity Measurements Using Neural Networks. Zoe. L. de Beurs, Andrew Vanderburg, Christopher J. Shallue, et al. August 2022

[5] The need for a public forecast of stellar activity to optimize exoplanet radial velocity detections and transmission spectroscopy. Lalitha Sairam and Amaury H. M. J. Triaud. May 2023

[6] Exoplanet Detection: A Detailed Analysis. Mahima Kaushik, Aditee Mattoo, and Dr. Ritesh Rastogi. April 2024

[7] Modelling Stellar Jitter for the Detection of Earth-Mass Exoplanets via Precision Radial Velocity Measurements. Samuel Granovsky, Irina N. Kitiashvili, Alan Wray. February 2022

[8] Randomization Inference of Periodicity in Unequally Spaced Time Series with Application to Exoplanet Detection. Panos Toulis and Jacob Bean. May 2021

[9] Statistical Methods for Exoplanet Detection with Radial Velocities. Nathan C. Hara, Eric B. Ford. August 2023

[10] Detection Limits of Low-mass, Long-period Exoplanets Using Gaussian Processes Applied to HARPS-N Solar RVs. N. Langellier, T. W. Milbourne, D. F. Phillips, et al. June 2021

[11] Radial-velocity variations due to meridional flows in the Sun and solar-type stars: impact on exoplanet detectability. N. Meunier and A.-M. Lagrange. April 2020

[12] 3D Magneto-Hydrodynamical Simulations of Stellar Convective Noise for Improved Exoplanet Detection. S. Sulis, D. Mary, and L. Bigot. February 2020

[13] Filtering Solar-Like Oscillations for Exoplanet Detection in Radial Velocity Observations. William J. Chaplin, Heather M. Cegla, Christopher A. Watson, Guy R. Davies, and Warrick H. Ball. March 2019

[14] A Machine Learning Approach for Correcting Radial Velocities Using Physical Observables. M. Perger, G. Anglada-Escudé, D. Baroch, et al. February 2023

[15] Improving Earth-Like Planet Detection in Radial Velocity Using Deep Learning. Yinan Zhao, Xavier Dumusque, Michael Cretignier, et al. May 2024

[16] A Gaussian Process Framework for Modelling Stellar Activity Signals in Radial Velocity Data. V. Rajpaul, S. Aigrain, M. A. Osborne, S. Reece, and S. Roberts. June 2015

[17] A detailed derivation of The Radial Velocity Equation. Kelsey I. Clubb. August 2008

[18] Noise Sources in Photometry and Radial Velocities, M. Oshagh, November 2017

[19] Determining the true mass of radial-velocity exoplanets with Gaia, F. Kiefer, G. Hébrard, A. Lecavelier des Etangs, E. Martioli, S. Dalal and A. Vidal-Madjar, January 2021

[20] Planetary detection limits taking into account stellar noise, X. Dumusque, N. C. Santos, S. Udry, C. Lovis and X. Bonfils, March 2011

[21] Planet Formation Theory in the Era of ALMA and Kepler: from Pebbles to Exoplanets, Joanna Drazkowska, Bertram Bitsch, Michiel Lambrechts, et al. May 2023

[22] Explanation of the Perihelion Motion of Mercury from General Relativity Theory, Albert Einstein, 1915

[23] Inhomogeneity within Local Interstellar Clouds, Jeffrey L. Linsky, Seth Redfield, Diana Ryder, and Adina Chasan-Taber, August 2022.

[24] Deep Residual Learning for Image Recosgnition, Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, December 2015

[25] Regularization of Neural Networks using DropConnect, Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. 2013

[26] GitHub. “diplomski_public”. Available Online. https://github.com/anterezic1999/diplomski_public

Tables

Figures

Code