UAB
Universitat Autònoma de Barcelona

Facultat de ciències

Departament de Física

Explainable AI for the mitigation of stellar effects in exoplanet observables

Jennifer Fabà Moreno

Treball de Final de Grau

Supervised by: Dr. Guillem Anglada-Escudé & Dr. Manuel Perger (ICE-CSIC)

Co-supervised by: Dr. Markus Gaug

Call: July 2024

DECLARACIÓ D'AUTORIA DEL TREBALL DE GRAU

Jo, **Jennifer Xintong Fabà Moreno**, amb Document Nacional de Identitat **48109023S**, i estudiant del Grau en Física i Matemàtiques de la Universitat Autònoma de Barcelona, en relació amb la memòria del treball de final de Grau presentada per a la seva defensa i avaluació durant la convocatòria de **Juliol** del curs **2023-2024**, declara que:

- El document presentat es original i ha estat realitzat per la seva persona.
- El treball s'ha dut a terme principalment amb l'objectiu d'avaluar l'assignatura de treball de grau en física en la UAB, i no s'ha presentat prèviament per ser qualificat en l'avaluació de cap altra assignatura ni aquesta ni en cap altra universitat.
- En el cas de continguts de treballs publicat per terceres persones, l'autoria està clarament atribuïda, citant les fonts degudament.
- En el casos en els que el meu treball s'ha realitzat en col·laboració amb altres investigador i/o estudiants, es declara amb exactitud quines contribucions es deriven del treball de tercers i quines es deriven de la meva contribució.
- A l'excepció del punts esmentat anteriorment, el treball presentat és de la meva autoria.

Signat:

JENNIFER XINTONG FABÁ MORENO - DNI 48109023S

Firmado digitalmente por JENNIFER XINTONG FABÁ MORENO - DNI 48109023S
Fecha: 2024.06.14 09:15:24 +02'00'

DECLARACIÓ D'EXTENSIÓ DEL TREBALL DE GRAU

Jo, **Jennifer Xintong Fabà Moreno**, amb Document Nacional de Identitat **48109023S**, i estudiant del Grau en Física i Matemàtiques de la Universitat Autònoma de Barcelona, en relació amb la memòria del treball de final de Grau presentada per a la seva defensa i avaluació durant la convocatòria de Juliol del curs 2023-2024, declara que:

- El nombre total de paraules (segons comptatge proposat) incloses en les seccions des de la introducció a les conclusions és de 6831 paraules i 7 fórmules.
- El nombre total de figures és de 15. En total el document, comptabilitza:

$$6831 \text{ paraules} + 15x200 \text{ paraules/figura} + 7x20 \text{ paraules/fórmula} = 9971 \tag{0.1}$$

  Que compleix amb la normativa al ser inferior a 10000.

Signat:

JENNIFER XINTONG FABÁ MORENO - DNI 48109023S

Firmado digitalmente por JENNIFER XINTONG FABÁ MORENO - DNI 48109023S
Fecha: 2024.06.14 09:15:47 +02'00'

Acknowledgments

I would like to thank Dr. Guillem Anglada-Escudé and Dr. Manuel Perger for their guidance and support during the writing of this work. Their valuable knowledge and experience in the field of Astrophysics has served as a perfect introduction for what it is going to become the field of research of my future professional career. Their perseverance in making me understand every phenomenon that we studied has made me acquire a strong basis in the state of art of Exoplanet detection. They have also introduced me in the Exoplanet group at Institute of Space Sciences (ICE) where I have felt really comfortable.

From this group, I would particularly like to thank Dr. Ignasi Ribas, who, actually, was my first contact at ICE and who first gave me the opportunity to join them. Also, I would like to thank Jordi Blanco for both helping me when I had coding problems as well as for letting me use some of his original GIFs and images for this work.

On the other hand, I have also appreciated a lot the useful comments that Dr. Markus Gaug gave me about this work and his useful advises for the incoming oral presentation. Finally, I would like to thank my mum, my sister Mireia and my friends for their constant support and love. Especially, I would like to thank Adolfo Hilario for his contributions to the structure of the work and for his constant inspiration and motivation and Angel Lorenzo for his thorough help along the whole degree.

Contents

Abstract

Stellar activity constitutes a barrier to precision in the radial velocity method for exoplanet detection. The main objective of this work is to use machine learning techniques to mitigate its effects in the measurement of radial velocities. We introduce autoencoders as neural networks tailored to store and compress the characteristic features of a given data set, thus providing more explanatory predictions than conventional networks. After preliminary tests with sinusoidal functions, we use simulated data from the Sun by using `Starsim` software. We find the optimal autoencoder architecture which provides a relative error of order 10^{-6}, while keeping a moderate computational work. The compressed data in the autoencoder potentially contains more information than the established so-called stellar activity indicators. Therefore, autoencoders constitute a promising method to characterise stellar activity, and hence be able to factor it out of radial velocity data.

1 Introduction

The discovery and characterisation of exoplanets (planets outside the Solar System) holds the promise to offer humankind insights of life outside Earth. Our own galaxy has between 100 and 400 billion stars, and the universe has upwards of 200 billion galaxies, making the chance for life elsewhere seem possible. Moreover, 22% of Sun-like stars harbour Earth-size planets orbiting in their habitable zones [1].

The aim of searching for Earth-like planets boosted when the first exoplanets were detected in the constellation of Virgo on 9 Jan 1992 by two radio astronomers: Aleksander-Wolszczan and Dale-Frail [2]. So far, 5664 exoplanets have been discovered (as of April 10, 2024), using different set-ups as, the space-based TESS (Transiting Exoplanet Survey Satellite) and Kepler telescopes or HARPS (High Accuracy Radial velocity Planet Searcher) and CARMENES (Calar Alto high-Resolution search for M dwarfs with Exoearths with Near-infrared and optical Échelle Spectrographs) spectrographs.

To analyse these data some of the existing exoplanets detection methods are transit event observation, radial velocity (rv), gravitational microlensing, direct imaging and pulsar timing [3]. At the end of 1995, only four exoplanets were known, three from pulsar timing and one from rv method, on which we focus our work. Between 1996 and 2013, the sample of known exoplanets was dominated by those discovered with rv, while since 2016, planets discovered by transits have maintained a fraction of about 75% by all methods combined [4]. In spite of all the discoveries, from this collection of data there are still large datasets to analyse, hence more exoplanet candidates remain to be discovered.

The central point of this work is not the detection of a new exoplanet itself but rather how to clean the available data: as the level of the measured rv reaches a precision of $0.2\,\mathrm{m\,s^{-1}}$, stellar intrinsic sources of noise capable of hiding the signal of these planets become more important. We want to understand this noise to accurately distinguish the subtle differences between actual exoplanets and false positives. These are mainly caused by the effects of the stellar activity in the spectra. Magnetic activity is the main source of stellar activity. It is highly complex and has been measured by a variety of the so-called stellar activity indices, which are still not fully understood.

Currently, methods for detecting exoplanets have been fused with machine learning, which is a state-of-the-art approach in science. In fact, some of the members of the Exoplanet group at Institute of Space Sciences (ICE-CSIC) have developed a software called `Starsim` [5] to simulate spectroscopy of a spotted rotating star. Once the spot map of a star has been created, the cross-correlation function (CCF) is built by cross-correlating the observed spectrum of a star with a mask. Then, they fit it to a Gaussian function and obtain information such as the full width at half maximum (FWHM, $\mathrm{m\,s^{-1}}$), the relative contrast (CON, in %) or the Bisector Inverse Slope (BIS, in $\mathrm{m\,s^{-1}}$), which are some of the activity indices mentioned above. These indices are affected by the presence of spots and other stellar characteristics which are also responsible for the change in the rv. Therefore the information needed to produce a clean rv measurement could be, in principle, available in the same observation through the indices. The use of these indicators has provided satisfying results [6] taking into account the information reduction implied.

In this study, we are going to check whether neural networks (NN) approach can learn better from the CCFs than the parameterisation with the above-mentioned indices. However, one problem of neural networks is the little explainability they provide about their predictions. In the wide range of fields where they are used, a rational answer is always desirable either when concluding that a patient has a disease or whether there has been a new discovery in a planetary system. Thus, we will investigate the architecture of some NN to assess the importance of the input data features in

the prediction of the output. It is what is known as explainable artificial intelligence (XAI).

Hence we introduce in our method an unsupervised machine learning technique: the autoencoders. They are a particular NN architecture, which lets us look into the characteristics of the data: by reconstructing the input data, they learn compressed representations (encoding) at the latent space, which we can more efficiently analyse and manipulate, leading to interpretability. Latent space is an abstract, lower-dimensional representation of high-dimensional data, often used to simplify complex data structures and reveal hidden patterns. The compressed data in the autoencoder potentially contains more information than the stellar activity indicators, so the main objective of this thesis is to understand this version of the data to know which features of the CCFs are more relevant. This way, we might improve the precision on the radial velocity method for the detection of exoplanets.

To develop our strategy, we structure this work as follows. In section 2, we give a brief introduction of astrophysical concepts related to star spectra, a formal definition of the CCF and a characterisation of NN architectures. In section 3, we present our experimental setup, explaining the `starsim` code thoroughly. In subsection 4.1, we discuss the work done with sinusoidal functions. We will deepen our knowledge about the architecture of the NN and the parameters stored in the latent space. Once exploiting the potential of autoencoders in the sinusoids, we apply the different experiments to simulated CCFs from the Sun (subsection 4.2) and study the potential of the chosen autoencoder architecture. Finally, section 5 is dedicated to a summary of the work as well as to comment the possible future lines of research.

2 Theoretical and Experimental Framework

We introduce the reader the necessary concepts to understand the rv technique and the construction of the CCF.

2.1 Stellar Activity

The nucleus of a star is the hottest of its zones and where the nuclear reactions take place. The energy is carried through the star by radiation and convection until it reaches the atmosphere star. The photosphere is the atmosphere's lowest and coolest layer where light and heat is radiated from. As stars are not rigid objects, they suffer from differential rotation, which is one of the responsible for the star magnetic field, as it happens in the Sun and solar-like stars [7]. This is the origin of the so-called stellar activity. The presence of intense magnetic fields in these regions inhibits convection, leading to reduced efficiency in heat transport. They drive hot plasma upward, where it radiates in faculae, leaving also dark spots in the photosphere (cooler and darker zones than their surroundings).

Stellar activity can be seen in the spectrum of the star, which distorts the spectral line profiles used to measure the apparent rv. When a surface spot crosses the stellar visible disc, it influences the stellar brightness. One has to recall that given the spherical geometry of celestial bodies, we have to take into account that the effect we visualise in the CCF, which we will formally define in subsection 2.3, corresponds to the projection of the spots onto the plane in the line of sight of the observer and not the real spots. This plane must not be confused with the plane of sky, which refers to the 2-dimensional plane that lies perpendicular to the line-of sight and passes through the star (see Figure 2.1).

Another signal of stellar activity are the finite number of convective features (granules) across the stellar surface. For instance, for the Sun, the apparent average velocity amplitude induced by the granules is about $1\,\mathrm{m\,s^{-1}}$[8]. So we see, that even the spectrum of a hypothetical spot-less and non-rotating star with precisely known physical and chemical properties will probably still not be sufficiently stable to serve as a "standard" on our intended levels of accuracy (currently, the precision measurements achieved have an uncertainty of $0.2\,\mathrm{m\,s^{-1}}$).

Stellar activity affects the symmetry of the CCF/spectral lines, therefore an apparent radial velocity shift gets induced. Refer to GitHub_TfgFisica[1], to visualise the effect that we are describing.

2.2 Exoplanet detection methods: the radial velocity technique

The rv method (also known as Doppler spectroscopy) is an indirect method of exoplanet detection based on the perturbation of the stellar motion. This movement, induced by the gravitational perturbation of a planet, is called "reflex motion". Mostly, a planet is engaged in its orbit by the gravitational field of a more massive object, in this case a star. However, the star is also subject to the gravitational force that the planet exerts on it. This means that the star moves, ever so slightly, in a small circle or ellipse, responding to the gravitational pull of its smaller companion. This represents a displacement of the spectra in wavelength, i.e. the Doppler shift, which results in an actual wavelength shift of the CCF.

Recalling Kepler and Newton's Laws, we obtain the semi-amplitude of the rv signal to be:

$$K_1 = \left(\frac{2\pi G}{P}\right)^{\frac{1}{3}} \frac{m_2 \sin(i)}{m_1^{\frac{2}{3}}\sqrt{1-e^2}} \tag{2.1}$$

[1]Image *CCF_AuMic_starsim_spot.gif*, by Jordi Blanco

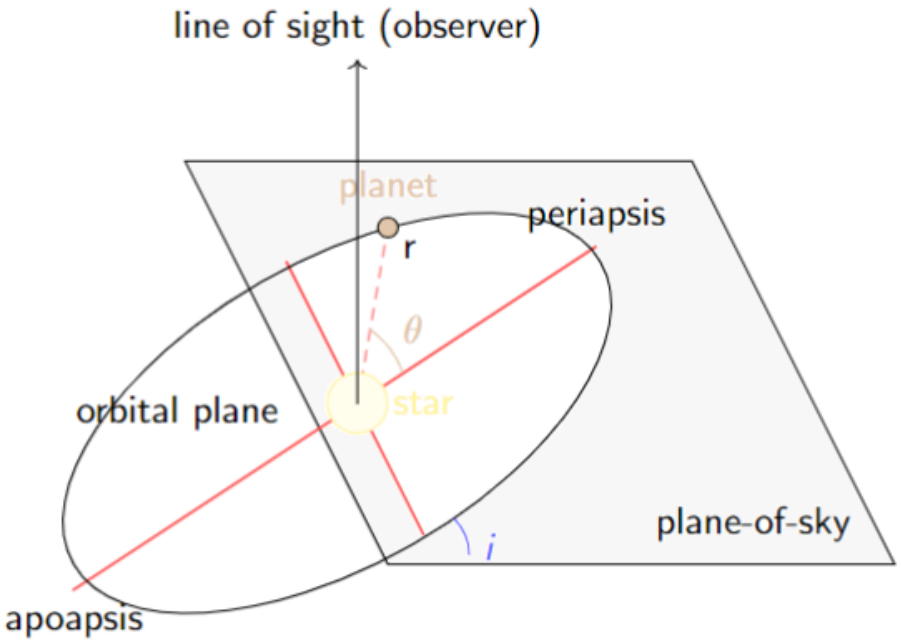

Figure 2.1: Plane of sky and orbital plane

where G is the Newton's gravitational constant, P is the period of the planet orbiting the star, m_1 is the star's mass, m_2 the planet mass, e is the eccentricity of the planet orbit and i the inclination of the orbital plane with respect to the line-of-sight plane (Figure 2.1).

From Equation 2.1, we see that by measuring K_1, P and e we can compute the minimum mass $m_2 \sin(i)$ of the orbiting planet. Although that this displacement might be (is) small, it is detectable for astronomers by using highly sensitive spectrographs attached to ground-based telescopes, like the HARPS spectrometer or ESPRESSO (Echelle SPectrograph for Rocky Exoplanets and Stable Spectroscopic Observations) at the Very Large Telescope in Chile. For instance, in the Solar System we have that Jupiter induces a $12\,\mathrm{m\,s^{-1}}$ rv signal on the Sun with a periodicity of 12 years, whereas Earth imprints a tiny $0.1\,\mathrm{m\,s^{-1}}$ signal at a 1-year period [9].

2.3 Cross-correlation function

The cross-correlation function (CCF) method involves cross-correlating the spectrum directly with a mask which represents a simple model of it. The mask (Figure 2.2), consists of a series of delta functions and it is defined using an observed spectrum of a similar star (it is also possible to use a synthetic spectrum). In this reference spectrum, we fit Gaussian functions to each minimum and select some of these to build the mask (associating a delta function to each).

Depending on how much radial velocity information each line contains, each delta can have different weights: deep and narrow lines have more information than shallow and broad lines, hence their weight in the mask is larger. Considering that the broad of lines is due to the thermal Doppler broadening, which takes into account the distribution of velocities of the particles and their temperature, the best targets are slowly rotating stars, thus they have narrower lines.

In order to construct the CCF, at each velocity step the spectrum is multiplied by the mask, hence only the spectrum pixels that overlap with a mask line (non-zero values) contribute to the CCF value.

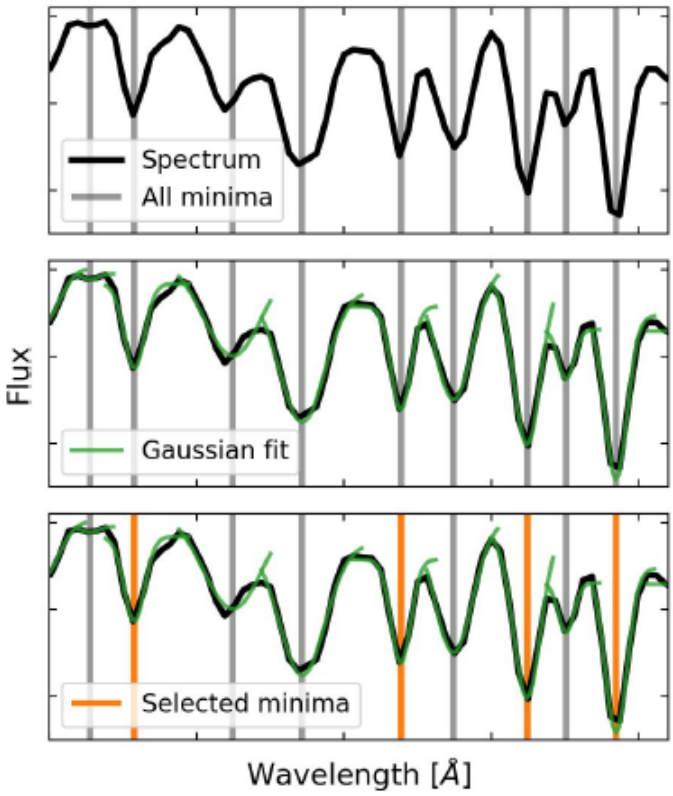

Figure 2.2: Construction of the mask. Image done by Jordi Blanco.

For a given velocity shift v, the cross-correlation of a spectrum consisting of n pixels with a mask with m lines is computed according to:

$$CCF(v) = \sum_{l=1}^{m} \sum_{x=1}^{n} w_l f_x \Delta_{xl}(v), \tag{2.2}$$

where f_x is the flux of the spectrum at pixel x, w_l is the weight of the mask line l and Δ_{xl} is the fraction of pixel x covered by the mask line l when the mask is shifted by v. Mask lines are defined as having a width of 1 pixel, which corresponds to the minimum interval containing relevant information. In the upper image of Figure 2.3, the mask is passed through the target spectrum and at each step, the overlap between them is measured and it gives a point in the graphic of the CCF (where the $x-$axis refers to the velocity step of the mask). Thus, the higher the order of the CCF value, the more similar is the real spectrum to the mask of selected lines.

From Equation 2.2, we see that the CCF represents an average of the profiles of all the stellar absorption lines that are present in the mask used in velocity space. The different processes in the stellar atmosphere that affect the average position and shape of the individual spectral lines are reflected in the CCF. Therefore, the shape of the CCF contains information of the distortions caused by stellar activity.

The next step consists in extracting the rv and the CCF activity indices by fitting a Gaussian function to the CCF profile:

- Radial velocity of the target star: corresponds to where the peak of the Gaussian is located (mean of the distribution).
- Activity indicators: used to measure temporal changes in the spectrum.
  - Full-width-at-half-maximum (FWHM): difference between the two values of v at which the CCF is equal to half of its maximum value.

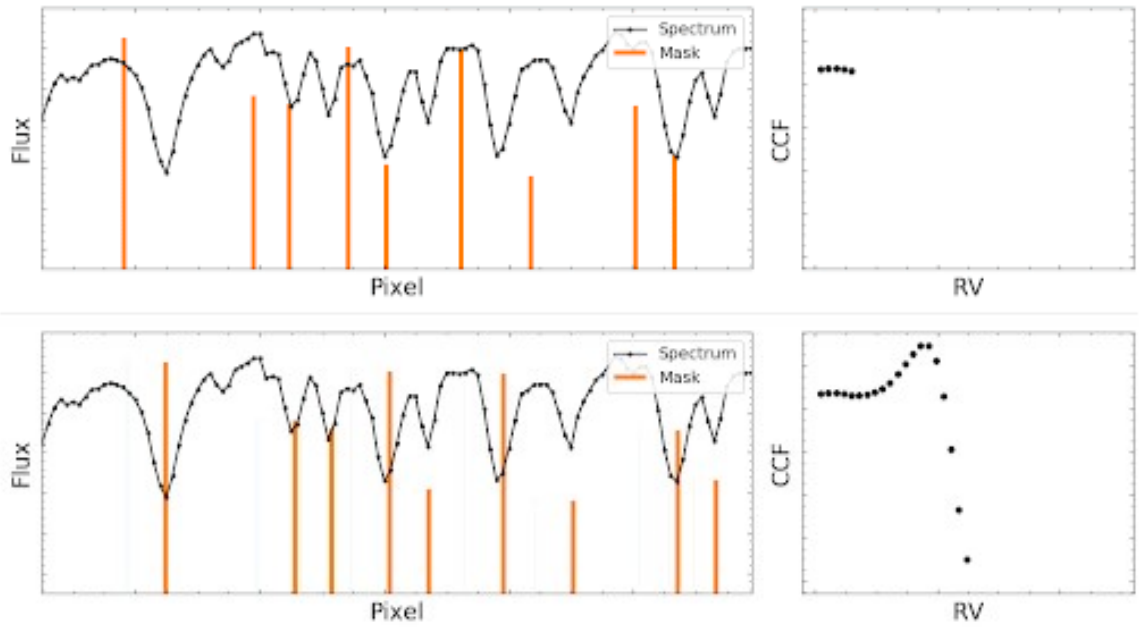

Figure 2.3: construction and evolution of the CCF (masks starts in the upper image and moves to the lower one). Image done by Jordi Blanco. GitHub_TfgFisica, Image: *ccf_mask.gif*

  - Contrast relative to continuum (CON): maximum amplitude (in absolute value) of the Gaussian fit.

- Bisector Inverse Slope (BIS): difference between the average velocity of the top region of the CCF (e.g. from 60 to 90%) and the average velocity of the bottom region (e.g. from 10 to 40%).

The presence of the same periodic signal in the rv and these parameters, may indicate the presence of activity-induced variations in the rv. For a more complete description of the construction of the CCF, refer to [10].

2.4 A General Introduction to Neural Networks

Neural networks (NN) can be seen as finite sets of structured neurons (nodes) which can be connected with its neighbours as well as with themselves. The information flows inside these connections from the *input* (first) layer through the network in the "hidden" layers and until the *output* (final) layer. In this layer, we obtain the output values subtracted from our initial information. We see how the term *hidden* supports the "black box" concept related to NN and their predictions.

Before training the NN one divides the initial dataset in 3 subsets: training, validation and test sets. In *supervised* learning methods, the input data contains the target values (labels) to which the network will compare its predictions; in *unsupervised* methods, the sets are unlabelled data which allows the network to discover patterns and insights without any explicit guidance. In both cases, the test set will not be used in the training process but rather will serve to assess how well the network generalises to new data. The training set is used to train the model (adjusting the parameters of the network) and the validation one aids in model selection and hyper-parameter tuning. The terms parameters and hyper-parameters do not refer to the same thing: whereas the former are learned during training (and random at initialisation), the latter are set prior to training and are not learned from the data. Some of the hyper-parameters that we can modify are:

- Loss function: function that the network will try to minimise during the training process. We use the function mean squared error (MSE) which corresponds to the mean of the squared

differences between the predictions and the actual values.

- Learning rate: controls how much change the model in response to the estimated error each time the model weights are updated.
- Batch size. A batch is a subset of the entire training dataset. The batch size, therefore, refers to the number of data points that the model is exposed to in one iteration of training.
- Number of epochs. An epoch defines the number of times that the learning algorithm will work through the entire training dataset.

NN can be classified in feed-forward or recurrent, according to the kind of connections that exists between the nodes. Feed-forward NN make the information flow in one direction, while recurrent ones let the architecture to have loops and allows the output information from some nodes to affect nodes of previous layers. In this work, we are going to deepen in the feed-forward type, particularly, in the following architectures: fully connected and convolutional neural network.

2.4.1 Fully Connected Neural Networks (FCNN)

Let us give a characterisation of a FCNN [11].

Let the inputs of the network be $\mathbf{x} \in \mathbb{R}^M$. The initial step is described by:

$$f_i^{(1)}(\mathbf{x}) = \sum_{j=1}^{M} \omega_{i,j}^{(1)} x_j + b_i^{(1)}, \tag{2.3}$$

where the superscript refers to the first layer of the network and the subscript ranges from 1 to H_μ, which denotes the number of nodes of the layer μ. From Equation 2.3 and from Figure 2.4, we see that the value in each node of the first *hidden* layer is a weighted combination of the input information plus a bias (b_i) associated to it. These weights $\omega_{i,j}$ and biases are the parameters that the network will adjust to reduce the loss.
Before the information gets to the next layer, we apply an *activation function* which is called *non-linearity*. It is defined by:

$$g_i^{(\mu)}(\mathbf{x}) = \phi(f_i^{(\mu)}(\mathbf{x})),$$

where i ranges from 1 to H_μ. The most typical non-linearities are: Rectified Linear Unit (ReLU), hyperbolic tangent and linear. The main objective of applying a non-linearity to data is not changing radically its behaviour but introducing a bit of complexity to the model to achieve more accurate results. In this case, the hyper-parameters that we are going to tune are the number of nodes and the activation function.

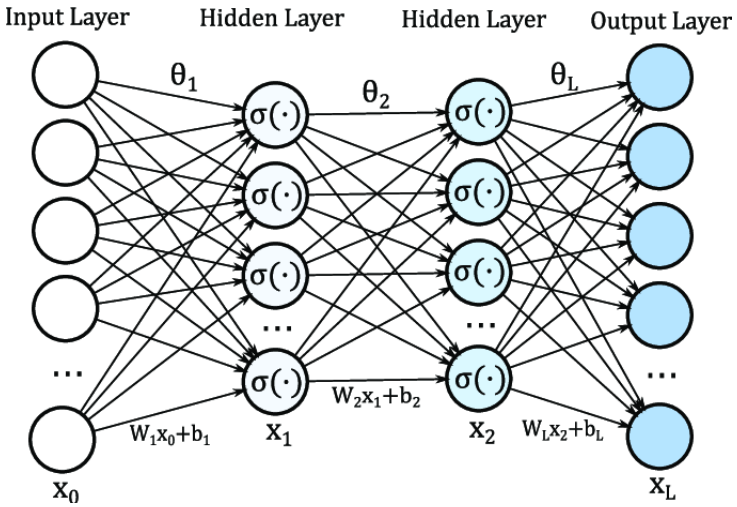

Figure 2.4: FCNN architecture. σ represents the activation function. Image taken from [12].

For a network with D *hidden* layers the recursion is:

$$f_i^{(\mu+1)}(\mathbf{x}) = \sum_{j=1}^{H_\mu} \omega_{i,j}^{(\mu+1)} g_j^{(\mu)}(\mathbf{x}) + b_i^{(\mu+1)},$$

where i ranges from 1 to $H_{\mu+1}$ (except in the case of the final activation where the top value is L). As the network has D hidden layers, $f^{(D+1)}(\mathbf{x}) \in R^L$ corresponds to its output value.

2.4.2 Convolutional Neural Network (CNN)

The most general architecture has some convolutional layers followed by fully connected ones to predict a single output (classification tasks). In convolutional layers, biases are the same for all the neurons in the same layer and each node is not connected with all of the ones in the following layer. Each layer might have different channels where a filter captures a feature of the input data. We obtain a *feature map* after convolving each filter with the input data of the layer: a summation of an element-by-element multiplication of the filters and input is calculated as the output of each channel (Figure 2.5). Some of the hyper-parameters that we can modify in this kind of layers are the quantity of the filters, the dimension of the kernels and the activation function.

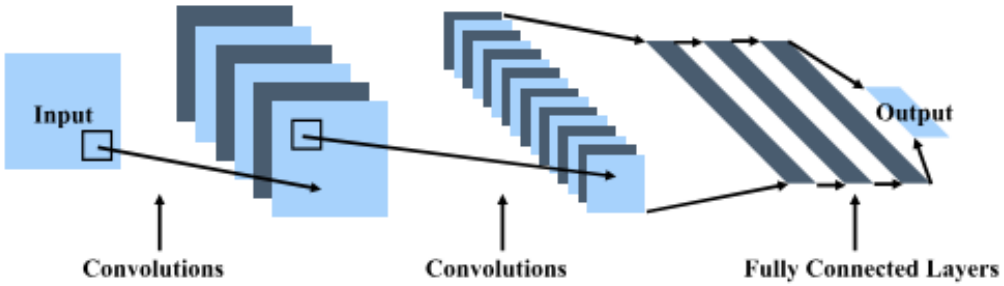

Figure 2.5: CNN architecture. Image taken from [13].

To enlighten the reader, let us introduce an example. Imagine our filter is a 2×2 matrix (2-dimensional kernel) and our input is a 3×3 one. First, the network places the filter in the upper left corner and it multiplies an element of the input by the element of the filter that coincides. After having done this with all the elements, the results are added up. This sum represents the first element of the output matrix. Then, the filter is moved $k-$steps (elements on the matrix) to the right, where k is a hyper-parameter that we can choose. In our naive example, the filter moves one step. Thus, the output matrix of this filter will have only 2 columns. Now that all the elements of the first row of the input have been analysed, the filter descends k-steps and repeats the process with the next row. In our case, the output matrix will be a 2×2 matrix, but the dimension of the output will not usually coincide with the filter.

2.4.3 Autoencoders

One of the main drawbacks of NN is the little explainability of their outputs. That is why we introduce the concept of autoencoders [14]. Deepening in this architecture helps us in subtracting the most relevant information of the input data to mitigate the problem of unreliability in NN.

Autoencoders are a type of NN in which the input is compressed to the 'bottleneck' layer, the so-called *latent space* (a representation with compressed data), and then uncompressed to make the output match the input dimension. They belong to the unsupervised learning methods category and, in this case, the loss function computed during training will also be the MSE introduced in subsubsection 2.4.1.

The particularities of these architectures are based on a double training process, hence the autoencoder might be considered as two independent networks: the *encoder* and the *decoder*. As its names suggest, the encoder reduces the dimensional representation of the input data to represent it in the latent space. On the other hand, the decoder works with the information stored in the latent space as input data and predicts the original input data of the autoencoder. We might use components separately for different purposes: the former is used to focus on the latent space and analyse the correlation between latent dimensions and the initial data: the encoder decides which details deserve to be retained; the latter would be used to reconstruct data from a compressed version.

One might use both fully connected layers as well as convolutional ones to construct the autoencoder, taking into account that the architecture of the encoder and the decoder must match in reversal.

3 Methodology

The main goal of this study is to work with the stellar effects on the CCF to distinguish between a surface stellar activity and exoplanets. We start by studying sinusoidal functions to understand the different NN architectures and their hyper-parameters. We work with FCNN and CNN and then we look into the latent space of autoencoders. Then, we adapt the architectures to the CCFs. In this case, we also start by a simpler problem, rather than to directly tackle the set of CCFs, we perform the experiments with one CCF as input. Finally, we work with the evolution in time of the CCFs, in order to consider the rotation of the star. I use Python as the programming language (refer to GitHub_TfgFisica).

3.1 Creating Data

The sinusoid dataset consists of 100 data points from the following function of time t (in days):

$$y(t) = A \sin\left(2\pi \frac{t}{T} + \phi\right), \tag{3.1}$$

where $A \sim U(A_1, A_2)$ is the amplitude , $\phi \sim U(\phi_1, \phi_2)$ is the phase and $T \sim U(T_1, T_2)$ is the period [2]. We consider the data to be taken each day for 100 days and the x-axis represents time in days.

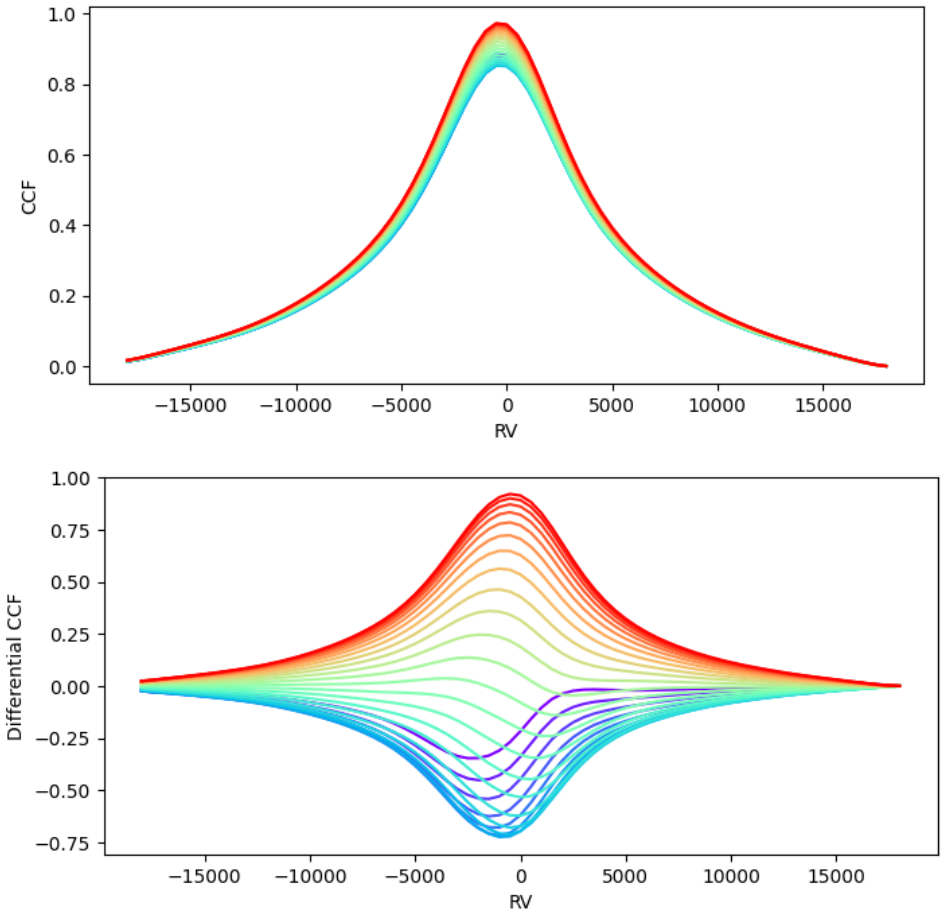

Figure 3.1: Upper: Example of a CCF. Lower: the differential version of the CCF

We create the simulated data with `Starsim`. We create a collection of 2000 CCFs from 1 spot (in each spotmap) with radius $R \equiv U(40, 60)$ and longitude $l \equiv U(270, 360)$ and we record the value of the CCF twice for 13 days: each CCF consists of 73 data points and each set has 27 CCFs. We set

[2] $X \sim U(a, b)$ means X is a random uniform variable with values between a and b

the longitude to be in the 4th quadrant angle to let the spot cross the centre ($l = 0$) of the star. We reshape the data to a matrix of 27×73, where each row represents a CCF. From the CCFs, we also compute the differentially to work with the change of the CCF in each time step with respect to the mean (see Figure 3.1).

The process of constructing the differentials follows these steps:

- Computing the mean of each point of the set of CCF. Each set has m CCFs and an element of the set, that is, a CCF, consists of n points. We can consider that we have a matrix of dimension $m \times n$. Then, the mean vector is an $n-$vector whose components are computed by adding up the elements of a column and dividing the result by m.
- Subtracting the vector of means to the original data. We subtract one-by-one the mean vector from the original $n-$vectors representing a CCF.

3.2 Experiments

We base our selection of the initial architecture of the neural networks on the examples found on the literature, such as in [15], and by the experience acquired during the development of this project.

1. Prediction of amplitude, phase and period of the sinusoids (GitHub_TfgFisica, File: *sinus_CNN*). We see how each activation function affects the predictions and assess the quantity of nodes required given the dimension of the input data. When performing these experiments with convolutional neural networks, we focus on the kernel's shape (see Appendix A). In this experiment, we normalise data using min-max feature scaling:

$$X' = \frac{X - min(X)}{max(X) - min(X)},$$

   where X' represent the normalised data.

2. The next experiment consists of studying the latent space of an autoencoder with the sinusoidal functions (GitHub_TfgFisica, File: *sinus_autoencoder*). To establish the dimension of the latent space we perform experiments on the sinusoid period and we adjust the hyper-parameter of the network. Once we have set the number of nodes in the bottleneck layer, we plot the latent space in 3D plots and analyse the effect of the initial parameters (amplitude, phase and period) in each dimension of it, by setting 2 of them to be constant and varying the left one.

3. We reproduce experiment 2 on the differential CCFs. In this case, we normalise the data diving it by the mean of the maximum absolute values of the CCFs, in order the data to be in between -1 and 1. This way, we choose all the activation functions to be hyperbolic tangent in these experiments. We apply stellar parameters from the Sun (refer to Appendix B, Table B.1).

   First, we choose one CCF as input data, to reduce the complexity of the problem. As the longitude of the spot is uniform and random, the selected time of the simulation from where we choose the CCF is not relevant. In this case, the encoder consists of 2 convolutional layers followed by a fully connected one and the decoder has the same structure but in reversal (GitHub_TfgFisica, File: *1_spot_1CCF_diff*). To set the initial architecture we set the dimension of the latent space to be 70, since as the input data has 73 points, we have almost one neuron per input point, so the autoencoder will (almost surely) provide satisfactory predictions.

   Finally, we make the analysis with the whole set of CCFs to see how the evolution of time affects the latent space (GitHub_TfgFisica, File: *1_spot_diff*). In addition to the experiments carried for 1 CCF, in this case we also choose between using 1D convolutional layers (`Conv1D`) or 2D ones (`Conv2D`).

In some of the experiments, NN got stuck, that is, there are times when the network does not achieve the global minimum of the loss function but rather stay around a local minimum. We solve this by reducing the learning rate of the optimiser (function that change the attributes of the network to minimise the losses), which allows it to explore the vicinity of the local minima more thoroughly.

3.3 Starsim

`Starsim` is the software we use to simulate spectroscopy of a spotted rotating star. It has two main modes:

- Forward mode: From a custom spot map and stellar parameters, allows simulating spectroscopy (RV, FWHM, BIS, contrast, CRX) at the desired times of observation.
- Inverse mode: From multiple (or single) time series of different observables and instruments, derive the spot map and the stellar parameters that best reproduce the input parameters.

We use the forward mode by choosing a target star and selecting the number of spots on its surface. A self-explanatory file `starsim.conf` [16] is available, where all the characteristics of the studied star can be changed (such as radius, temperature of the photosphere and rotation period, to name a few) as well as all the parameters related to the CCF (CCF velocity range, radial velocity step and template used in the CCF computation,...), spots (law for the evolution of the spot) and planet (period, time of transit,...) that orbits the star. We can add an arbitrary number of spots whereas we can only add one orbiting planet.

4 Results and Discussion

4.1 First Example. Sinusoid

4.1.1 Predicting Amplitude, Phase and Period

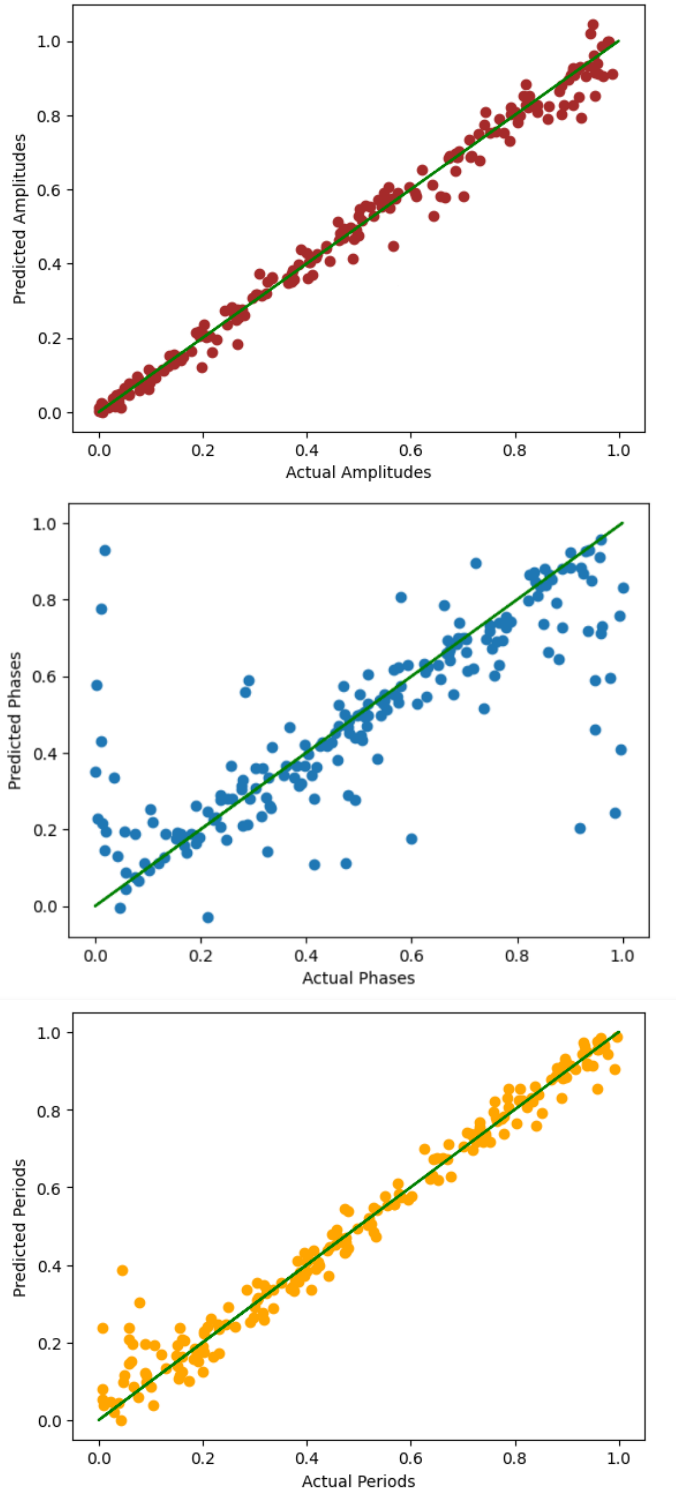

Figure 4.1: Predicted values of (from top to bottom): amplitudes, phases and periods, with respect to the expected (original) labels.

From the first plot of Figure 4.1, the amplitude is the parameter with more accurate predictions: relative error is of order 10^{-3}. The relationship of A with $y(t)$ is linear for fixed ϕ and T, hence it is easier for the network to learn this relationship. Moreover, when training networks with only this value as output, we polish results (Appendix C, Figure C.1).

On the other hand, regarding the centre plot of Figure 4.1, predictions of low and high values of phase are the one with greater error associated. Although the network might have learnt the periodicity of this parameter, the predictions appear to be random since, in order to coincide, low values of the actual phases should have either predictions around 1.0 or 0 and vice-versa, but what we see is a vertical dispersion of points. We might have obtained better results by normalising data differently or changing the architecture of the network, but we decide not to extend this experiment (since we are going to deepen in NN architecture in the next section).

Regarding the period plot, we see that low periods are the ones which presents the greatest dispersion. When working with autoencoders, we will perform a deeper analysis of this phenomenon, which is called *aliasing*, and for now we refer the reader to Appendix D or to wait until subsubsection 4.1.2.

Overall, as the amplitude is the only parameter outside the sine function, it is expected that its predictions are more accurate, since its relationship with the input data is simpler, and that is what we can see by comparing the three plots.

4.1.2 Autoencoders

Now, we merge the autoencoder architecture with the sinusoidal functions.

First, we establish the interval of (sinusoidal) periods for which the autoencoder gives more accurate predictions. To acquire numerical data, we train the autoencoder and keep the validation loss of the last epoch: in Figure 4.2 we can see its evolution with respect to the extrema of the interval in which we have defined the sinusoid period.

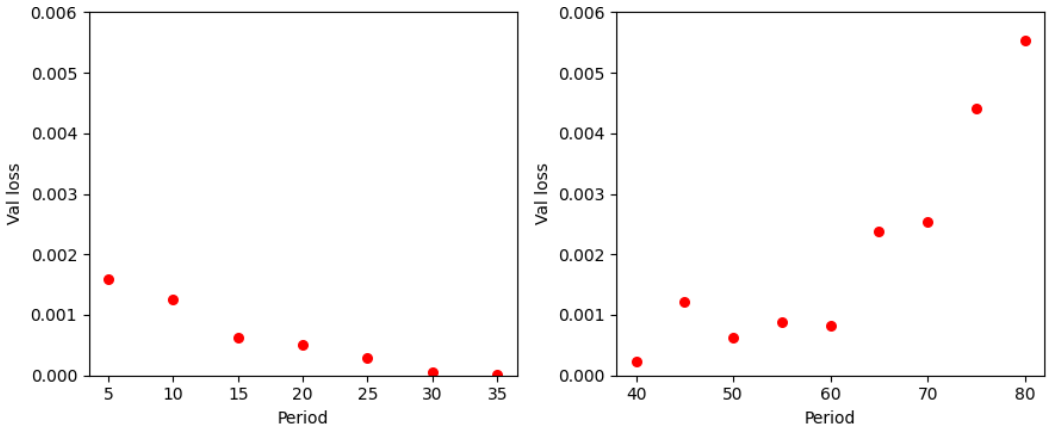

Figure 4.2: Validation loss with respect to minimum (left) and maximum (right) periods of the sinusoidal.

We see that periods ranging from 15 to 40 are the most appropriate for further analysis. Of course, working with all periods would be possible (all predictions have a validation loss of order 10^{-3}), but our goal is not working in the most general case but rather obtain more accurate predictions. From now on, we set the range of periods to be: $T \sim U(15, 40)$. Selection of periods represents an important part of the analysis, since one should consider the *Nyquist* frequency of the sampled data, which corresponds to one-half the sample rate. Frequencies higher than this will lead to the above-mentioned phenomenon of aliasing, which will reduce the accuracy of the results. In our case, the minimum period for which we could get reasonable results, is 2, since we have a sampling rate of 1, that is why we have chosen to do the selection of periods starting from 5 (see Appendix D to see examples of this phenomenon).

Now, the parameter we are going to study is the dimension of the latent space, that is, the number of nodes in the bottleneck layer.

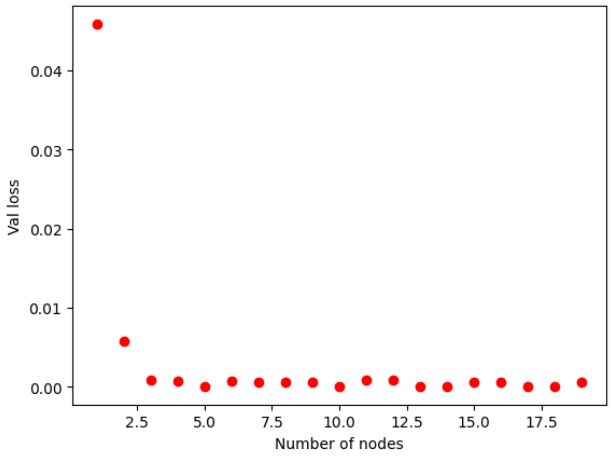

Figure 4.3: Validation loss with respect to number of nodes in the 'bottleneck' layer.

The considered number of nodes runs from 1 to 19 (see Figure 4.3). From the definition of the sinusoid (Equation 3.1), we know we that can characterise it by providing only 3 parameters: phase ϕ, period T, amplitude A, hence 2 nodes should be enough for the network to give satisfactory results. That is because one should consider that each connection is related to a weight and the node with a bias, so starting from only 2 nodes, the network should be able to produce accurate predictions. We train the autoencoder several times, optimising its hyper-parameters and we get a validation loss of order 10^{-2} for N=2 (see Figure C.2). However, we establish the dimension of the latent space to be 3 to obtain more precise results in further experiments, so the bottleneck layer will have 3 nodes; Figure 4.4 shows the level of precision obtained in predictions.

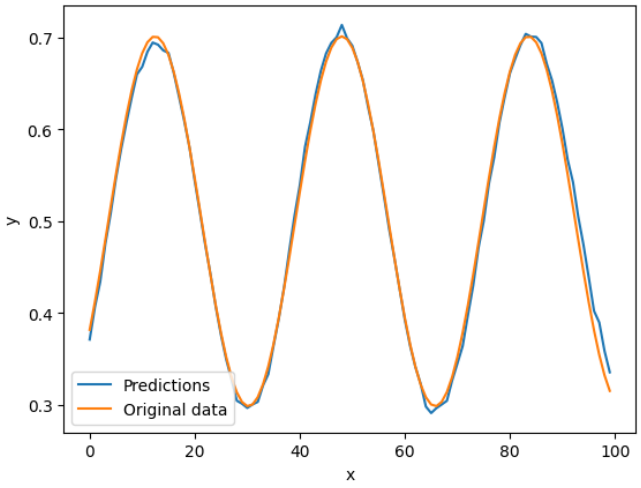

Figure 4.4: Original sinusoidal function and predictions from the autoencoder for $N = 3$.

In order to understand which information is stored in the latent space, we use the encoder part of the autoencoder (which is already trained) using as input data sinusoids with 2 parameters fixed, thus studying how latent space changes with respect to one of the sinusoid's parameters: period, phase or amplitude.

For data with phase variable we have $\phi \sim U(0, 360)$; for the sinusoid with variable amplitude: $A \sim U(1, 10)$; and finally for data with variable period $T \in (25, 35)$.

We might have expected to have a one-to-one correspondence between parameters and latent dimensions, since both are 3. However, all latent dimensions change when changing any of the parameters, so we cannot associate any particular dimension to a specific parameter (see Figure 4.5 and Appendix C, Figure C.4).

In the case of (variable) phase, we see that the the values for the latent space data coincide for $\phi = 0$ and $\phi = 360$, hence we might conclude that the autoencoder has captured the periodicity in data. Amplitude is the parameter which makes the latent space data follow a steady variation, both phase and period are more complex parameters.

The `rainbow` function in Python starts plotting with colour red and grading the colour-scheme until purple. To be able to identify to which input data each point corresponds, we have sorted the random vectors of each parameter to assign the colour red to the smallest values and the purple to the largest. Therefore, we can divide the latent space into regions in which we can estimate the value of the parameters.

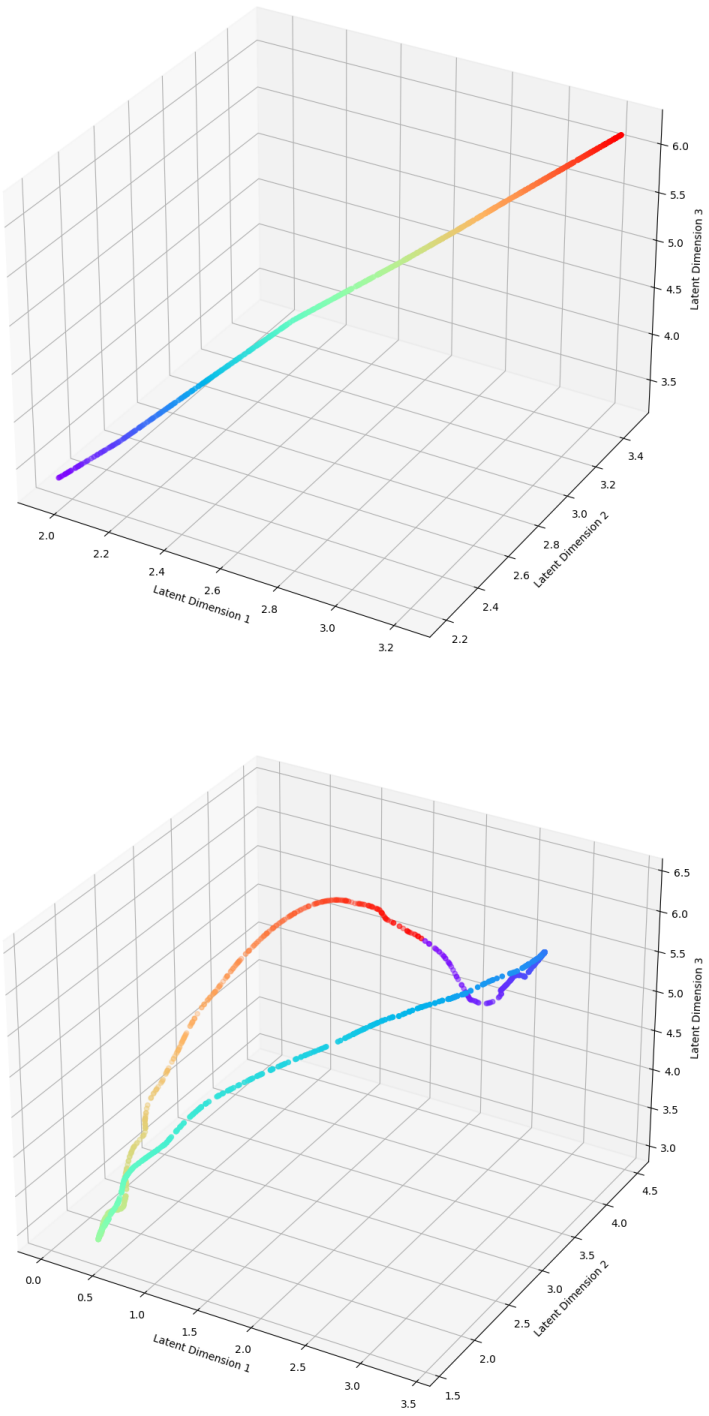

Figure 4.5: Latent space data for input data with variable amplitude (upper) and variable phase (lower).

In Appendix C, Figure C.6 each subplot of a row represents a dimension of the latent space and each complete row represents the latent space of the autoencoder, giving a specific visualisation for each latent dimension. Being able to interpret the latent space helps us to identify which patterns or features are the most valuable for the network. That way, in future predictions we will know which parameters the network is most sensitive to. As the latent space has dimension 3, we can visualise the data in a 3D plot without requiring dimension reduction techniques such as UMAP (Uniform Manifold Approximation and Projection) [17].

4.2 Starsim one-spot models

4.2.1 1 CCF

After doing experiments with the autoencoder architecture (changing the number of nodes of the bottleneck-layer and optimising its hyper-parameters), we obtain that the predictions of highest faithfulness are achieved when we use `Conv2D` layers. Although we have only plotted 2 random CCFs from the test set (see Figure 4.6), we state a validation loss of order 10^{-6}, which compared to the values of the differential CCF, represents a more than adequate order.

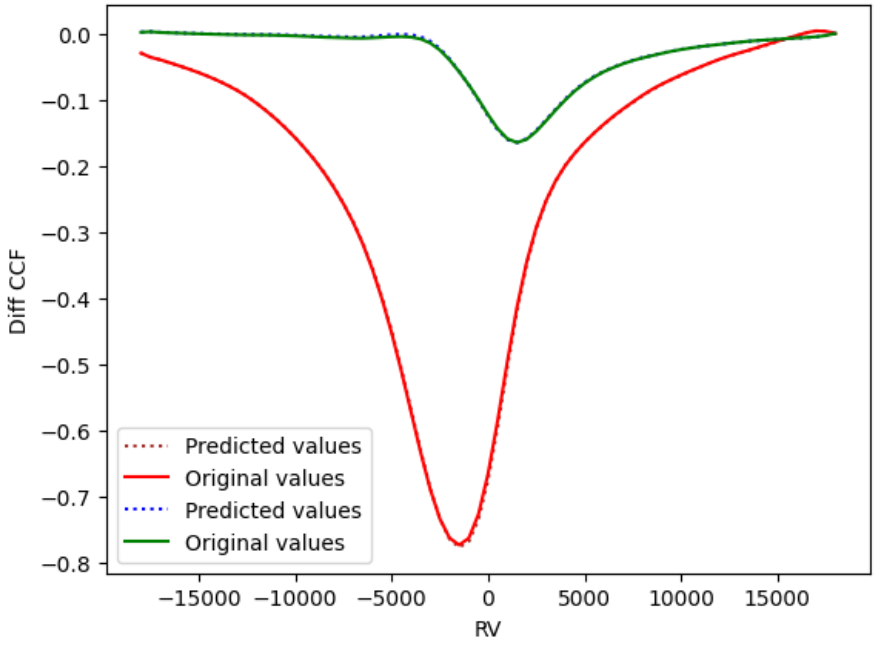

Figure 4.6: Some test data with their corresponding predictions.

4.2.2 Evolution of CCFs

Now that we have an idea of what kind of architecture is valid to work with the differential CCFs, we are able to use as input data the complete set of CCFs.

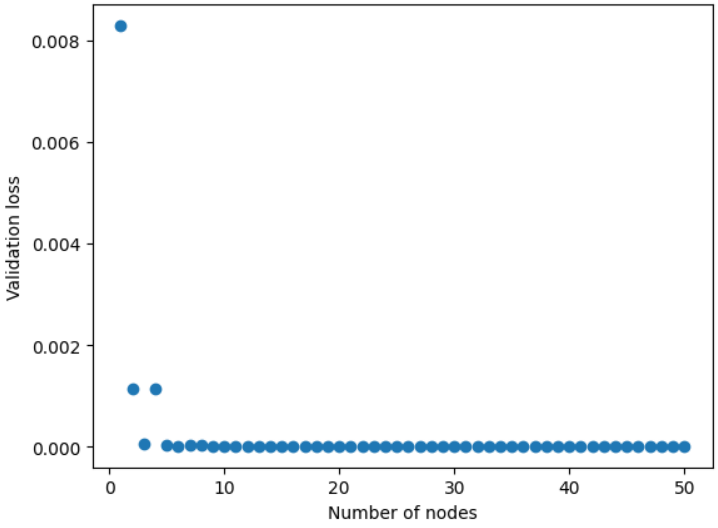

Figure 4.7: Validation loss with respect to the number of nodes in the bottleneck layer.

In Figure 4.7, we see that beyond 3 neurons the autoencoder achieves validation losses of order 10^{-6} (relative error 10^{-6}) and remains considerably constant (Appendix C, Figure C.5 to a zoomed version of Figure 4.7), thus we consider that the network has converged. This tendency reminds us to the results obtained for the sinusoids (Figure 4.3). Although the more neurons the better the predictions, the computational cost is higher and might not be worth the time. Thus, we continue by setting the dimension of the latent space to be 3, which will also let us visualise it without having to use dimension reduction techniques. Taking into account that we are working with data from a single spot from which we have only changed the radius and the appearing longitude, it is not surprising that the autoencoder needs this little quantity of nodes to reconstruct the information. In future works, we will work with more surface spots and we will change more of their parameters such as the co-latitude or the contrast temperature with the star surface.

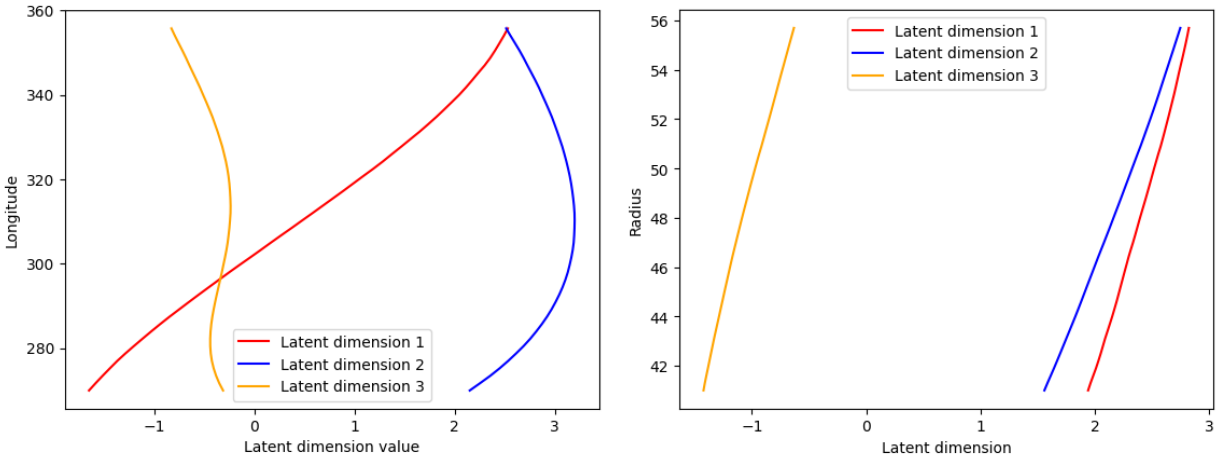

Figure 4.8: Visualisation of the latent space for the set of CCFs. Left: input data with constant radius; right: input data with constant longitude.

In Figure 4.8 we see the evolution of the different latent space dimension when having used the autoencoder with data of constant radius or longitude (Appendix C, Figure C.7 for a 3D visualisation).

The case of the radius reminds us to the linear relationship between amplitude and the latent space of sinusoid autoencoder. As in the bottleneck layer we have used the activation function `'linear'`, we do not have values restricted to -1 and 1. One intriguing characteristic of the values of the latent space is that the values of a fixed dimension do not change sign.

On the other hand, the longitude plots are quite similar to the period ones of the sinusoids. This let us see the relation between time and longitude that we already anticipated when discussing the selection of CCFs for the 1CCF experiment.

5 Conclusion

We have used neural networks (NN) approach to work with the cross-correlation function (CCF) of models of the stellar spectrum of the Sun to mitigate the stellar effects on it. We have analysed the latent space of the autoencoders trained with the sets of CCFs to assess whether the information stored in it is accurate and faithful for further analysis, so we can replace the current activity indicators used. Of course, all these analyses can be generalised to any star simply by changing the initial parameters on the codes.

We performed the initial experiments on sinusoids as input data, to become familiar with NN architectures. We identified which of the proposed architectures produced more accurate predictions by optimising their hyper-parameters. We study the phenomenon of aliasing and improve the input data by setting a higher minimum period. The next step was to examine the latent space to see how the variations of the parameters that describe a sinusoid (amplitude, phase and period) affect its values. As the validation loss stabilised beyond 3 neurons, we set the latent space to have dimension 3. Hence, we analysed the latent space in 3D plots. Although we expected the autoencoder to store the information of one parameter in a single dimension, we found that the 3 of them depended on all the parameters. Thus, we could not associate each latent dimension to a feature of the input data.

In the first experiment with the simulated data of the Sun, we study one CCF as input data. We train an autoencoder and found that a few convolutional layers followed by a few dense layers was the most effective architecture. As first attempt, we applied this architecture to the complete set of CCFs and, by optimising the architecture we found the optimal one while keeping a modest computational work. Actually, producing simulation in *starsim* was the most computer intensive part of the process.

To study the latent space, we tracked the value of the validation loss to see which number of nodes in the bottleneck layer performed better. Coincidentally, the optimal number of nodes is 3, as it happened with sinusoids. The fact that this number is so small is not surprising given that, although we were not predicting periodic functions like sinusoids, we have only changed 2 parameters of the surface spot (its longitude and radius) keeping the functions that we are studying also relatively simple. With this architecture, we obtained a relative error of 10^{-6} which let the autoencoder reconstruct the input differential CCFs with really high accuracy. Thus, from only 3 points, it is able to reconstruct a set of $27 * 73 = 1971$ input points, implying a reduction of the necessary input information. The compressed data might contain more information than the activity indicators, becoming a potential substitute for them.

As a future step, we plan to test the autoencoder approach with more complex data, such as studying in more detail the one spot case as well as simulating data of stars with a random number of spots. For the first experiment proposed, we want to vary all the parameters at once and compare the latent space parameters with Starsim parameters and with CCF indices (FWHM, CON, BIS). On the other hand, we expect the second experiment to imply an increase of the latent dimension, which will make us investigate more in the UMAP method. Recall that our main objective is to understand the compressed version of the data to know which features of the CCFs are more relevant so we can improve the precision on the radial velocity method for the detection of exoplanets.

Bibliography

[1] Erik A Petigura, Andrew W Howard, and Geoffrey W Marcy. "Prevalence of Earth-size planets orbiting Sun-like stars". In: *Proceedings of the National Academy of Sciences* 110.48 (2013), pp. 19273–19278 (cit. on p. 1).

[2] A. Wolszczan and D.A. Frail. "A planetary system around the millisecond pulsar PSR1257 + 12". In: *Nature* 355.6356 (Jan. 1992), pp. 145–147 (cit. on p. 1).

[3] Ziqi Dai et al. "Five methods of exoplanet detection". In: *Journal of Physics: Conference Series*. Vol. 2012. 1. IOP Publishing. 2021, p. 012135 (cit. on p. 1).

[4] Hans J Deeg and Roi Alonso. "Transit photometry as an exoplanet discovery method". In: *arXiv preprint arXiv:1803.07867* (2018) (cit. on p. 1).

[5] Enrique Herrero et al. "Modelling the photosphere of active stars for planet detection and characterization". In: *Astronomy & Astrophysics* 586 (2016), A131 (cit. on p. 1).

[6] M Perger et al. "A machine learning approach for correcting radial velocities using physical observables". In: *Astronomy & Astrophysics* 672 (2023), A118 (cit. on p. 1).

[7] ARG Santos et al. "Starspot signature on the light curve-learning about the latitudinal distribution of spots". In: *Astronomy & Astrophysics* 599 (2017), A1 (cit. on p. 3).

[8] Lennart Lindegren and Dainis Dravins. "The fundamental definition of "radial velocity"". In: *Astronomy & Astrophysics* 401.3 (2003), pp. 1185–1201 (cit. on p. 3).

[9] Michel Mayor, Christophe Lovis, and Nuno C Santos. "Doppler spectroscopy as a path to the detection of Earth-like planets". In: *Nature* 513.7518 (2014), pp. 328–335 (cit. on p. 4).

[10] M Lafarga et al. "The CARMENES search for exoplanets around M dwarfs-Radial velocities and activity indicators from cross-correlation functions with weighted binary masks". In: *Astronomy & Astrophysics* 636 (2020), A36 (cit. on p. 6).

[11] Alexander G de G Matthews et al. "Gaussian process behaviour in wide deep neural networks". In: *arXiv preprint arXiv:1804.11271* (2018) (cit. on p. 7).

[12] Andrea M Tonello et al. "Machine learning tips and tricks for power line communications". In: *IEEE Access* 7 (2019), pp. 82434–82452 (cit. on p. 7).

[13] Liam A Kruse et al. "Dyadic sex composition and task classification using fNIRS hyperscanning data". In: *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE. 2021, pp. 582–588 (cit. on p. 8).

[14] Scott M Lundberg and Su-In Lee. "A unified approach to interpreting model predictions". In: *Advances in neural information processing systems* 30 (2017) (cit. on p. 8).

[15] Francois Chollet. *Deep learning with Python*. Simon and Schuster, 2021 (cit. on p. 11).

[16] David Baroch López. *Starsim code*. https://github.com/dbarochlopez/starsim. Accessed: Jan 2024 (cit. on p. 12).

[17] Leland McInnes, John Healy, and James Melville. “Umap: Uniform manifold approximation and projection for dimension reduction”. In: *arXiv preprint arXiv:1802.03426* (2018) (cit. on p. 16).

A Convolutional Layers's Kernels

To examine a bit more in the convolutional architecture, we visualise the filters of the first convolutional layer of the network with output labels amplitude, phase and period.

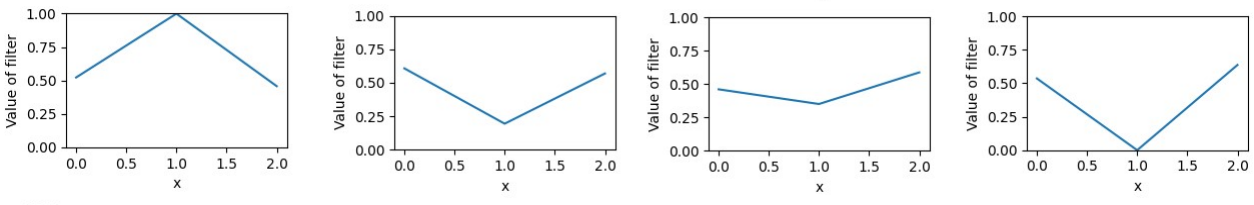

Figure A.1: Filters of the first layer of convolutional network.

Let us first describe what we see in the different plots. The $y-$axis represents the value of the filter. The $x-$axis refers to a node in the layer. Recall the fact that each filter represents a feature to be learned. As the dimension of the filter is 3, we have that each filter covers 3 neurons, so the value $x = 1$ corresponds to the neuron where the filter is centred on. For example, the 4-th filter corresponds to the difference symmetric quotient $\frac{f(x+h)-f(x-h)}{2h}$ a discrete version of the derivative function. In our case, $h = 1$, since it is the separation between 2 data points. We see that the other filters are combinations of the 3 points involved, but we are not able to detect any particular feature in them.

B Tables

Rotation period (at equator)	24.47 days
Convective shift	1
Differential rotation	-2.66 deg/day
Axis inclination	7.25 deg
Photosphere temperature	5772 K
Mass (over M_{sun})	1
Radius (over R_{sun})	1
Logg (stellar surface gravity)	5.61

Table B.1: Characteristics of the Sun.

C Figures

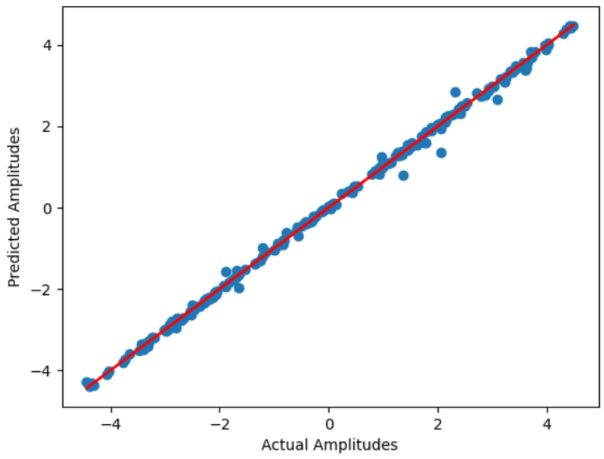

Figure C.1: Predicted values for amplitude with respect to the expected (original) labels with amplitud as the only output label.

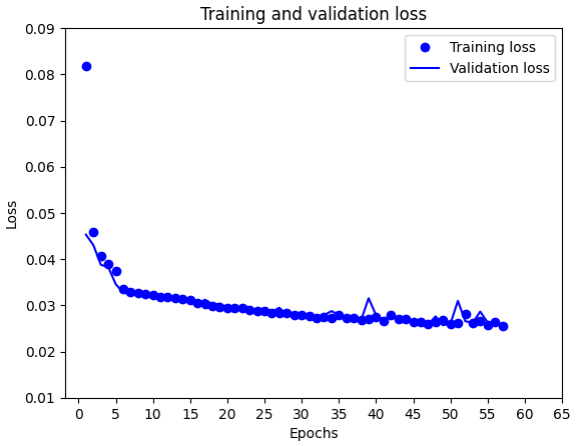

Figure C.2: Training and validation loss for N=2

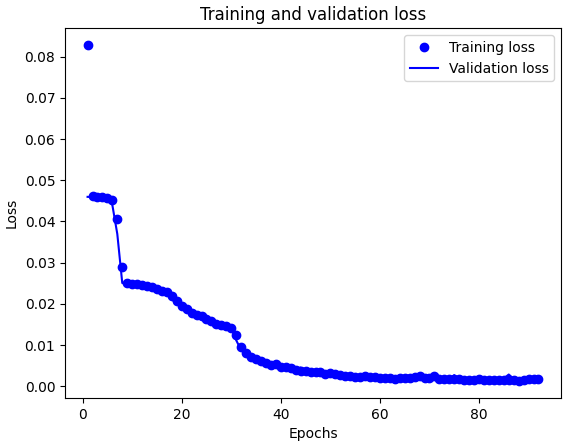

Figure C.3: Training and validation loss for N=3

N stands for number of layers in the “bottleneck” layer as used in text.

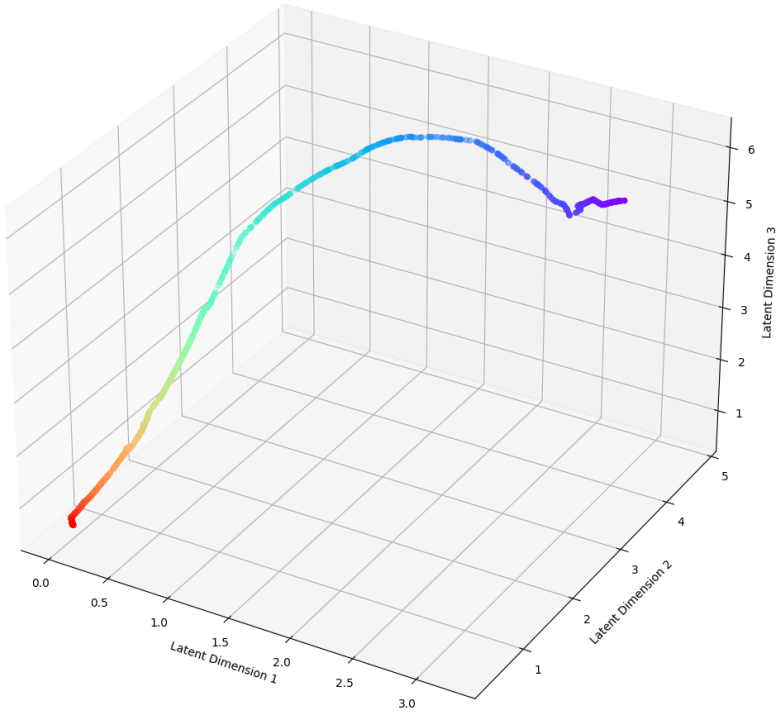

Figure C.4: Latent space data for input data with variable period.

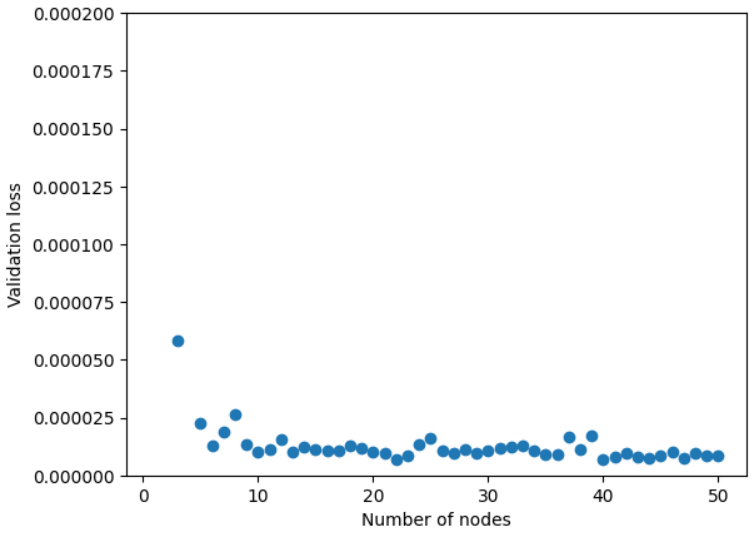

Figure C.5: Zoomed version of the validation loss graphic with respect to the number of nodes in the bottleneck layer for complete set of CCFs.

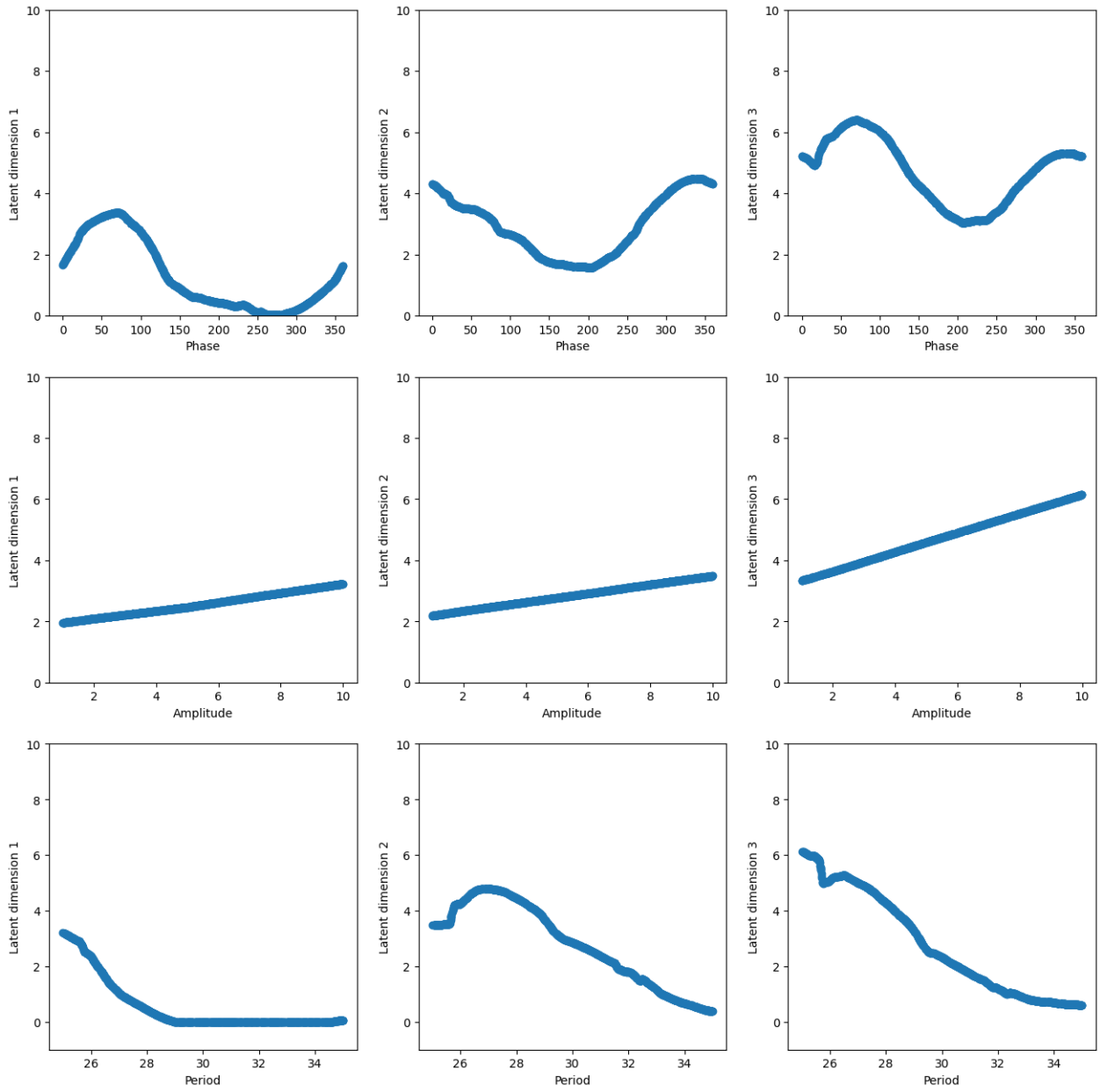

Figure C.6: Upper: variation of the latent space data for every dimension with respect to phase. Mid: variation of the latent space data for every dimension with respect to amplitude. Lower: variation of the latent space data for every dimension with respect to period.

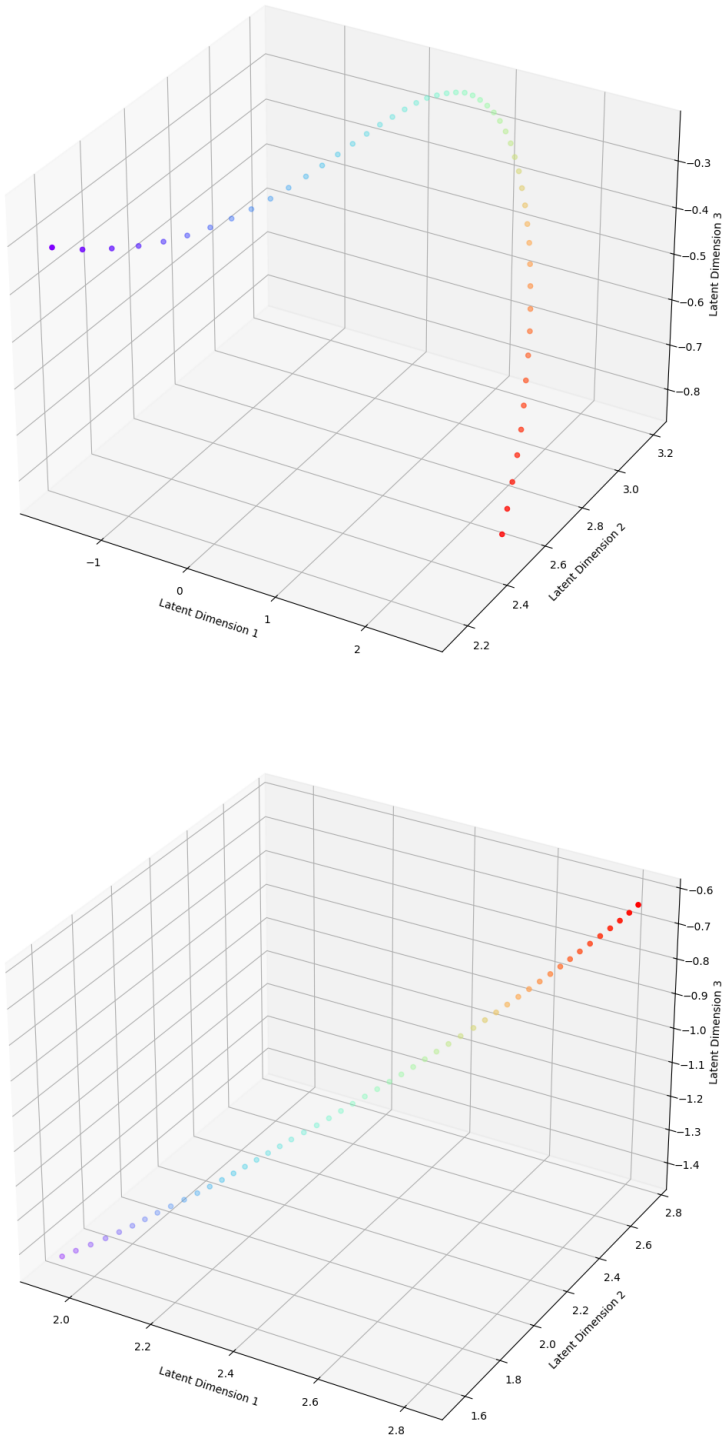

Figure C.7: Visualisation of the latent space for the set of CCFs in 3D. Upper: input data with constant radius; lower: input data with constant longitude.

D Aliasing

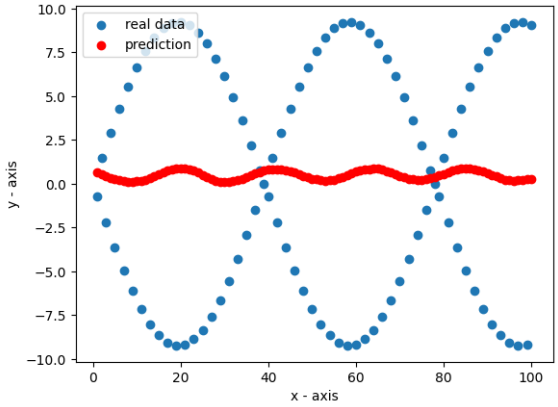

Figure D.1: Sinusoidal function with $T = 1.95$ (blue) and autoencoder's prediction (red).

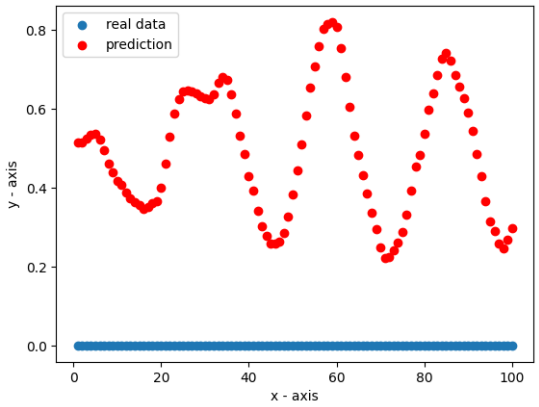

Figure D.2: Sinusoidal function with $T = 1$ (blue) and autoencoder's prediction (red).

Regarding the first example of the aliasing phenomenom (Figure D.1)[3], we see that when studying sinusoids whose periods are close to 2, the computer misunderstands the input function. On the other example (Figure D.2), we see that as the sinusoid period is 1, and the sampling rate is also 1, we obtain a constant function, since we are only sampling points of the function whose value is 0. We see that by incorporating these samples on the training/validation set, we would have obtained larger validation losses. Thus, from now on, we use the periods selected from this experiment.

[3]Both example sinusoids have the same amplitude and phase.