Deriving Physical Laws with Neural Networks

Max Fusté Costa

February 2023

Abstract

The usage of neural networks to derive physical laws without any kind of pre-existing bias is a promising but relatively new field with the long-term goal to construct an artificial intelligence physicist that is able to derive physical laws from experimental data. In this project, a step is taken in the direction of solving complex problems by tackling the double pendulum, the simplest chaotic system. To do so, a neural network architecture, adapted from previous work, is used to find the relevant parameters of the system in multiple configurations of the pendulum. Afterwards, the possibility of a neural network derived general solution of the problem is discussed through the relevant aspects that increase its complexity.

Contents

1 Introduction

In physics, as in all other fields of human activity, new theories are built upon previously established knowledge, and are influenced by the schools of thought prevalent at the time of their development. Thus, all theories are somewhat biased and might not be the most natural and simple ones to describe experimental data, but rather the ones that easily follow from said previously established knowledge. This cognitive bias, tied to the nature of the human brain, risks the introduction of roadblocks in our way to scientific progress [1,2]. Quantum mechanics and general relativity, theories build from classical mechanics, are incredibly successful in their respective regimes. However, they are fundamentally incompatible, as gravity cannot be quantized, and are therefore non-unifiable under one sole theory.

One possible solution to know if the currently known physical theories are the most natural to explain experimental data is to try obtaining them again after removing the cognitive bias [1-4]. This is, unfortunately, not reliable from a human perspective, since said bias is tied to our way of thinking, but the recent advances in artificial intelligence allow for a first step down this path.

Early works in this field were focused on acquiring the mathematical expression describing a set of data [2]. They were successfully applied to complex physical systems, but required some sort of previous knowledge on the problem in question, such as the relevant variables of the system. Ideally, one does not want to impose any information so that the machine obtains a representation of the system that is free of bias.

The last years have seen an increase in the use of neural networks (NN) in machine learning. Physicists have used them to describe complex physical systems in both classical and quantum physics [3-5]. For example, in [3] the network finds the physically relevant parameters and predicts the future state of the system from conservation laws, helping to gain conceptual insight of problems such as the damped pendulum and quantum state tomography.

This project attempts to take the next step in this field by tackling the problem of the double pendulum, a notoriously complex system for its chaotic behaviour, while minimizing the amount of assumptions of the physical system imposed on the NN. Usually, it is desirable to sacrifice performance in order to gain generality, as it is necessary for the long-term goal of designing an artificial intelligence physicist that can work with any set of conditions [5]. However, chaotic systems have a big dependence on the initial conditions [6], meaning that a slight variation completely changes the outcome, making a general prediction much more computationally demanding than a linear problem. Therefore, the simulation must be limited to a small range of parameter values in order to ensure a certain degree of predictive power with a reasonable performance.

The double pendulum system is modelled in Python 3. The differential equations that describe its motion are solved for a fixed value of the initial angles and different sets of values for the lengths of the rods and the masses of the pendulums that are randomly generated in a given range. The NN used to train and run the model is a slight modification from the code used in [3].

1.1 Double Pendulum

The double pendulum is a dynamical system that consists of one pendulum, with length L_1 and mass m_1, with another pendulum attached to its end, with length L_2 and mass m_2. While fairly simple in structure, the double pendulum has a complex dynamic behaviour with a strong sensitivity to initial conditions [7]. The motion of the system is dictated by the following ordinary differential equations (ODEs).

$$(m_1+m_2)L_1\dot{\theta}_1 + m_2L_2\ddot{\theta}_2 cos(\theta_1-\theta_2) + m_2L_2\dot{\theta}_2 sin(\theta_1-\theta_2) + (m_1+m_2)gsin(\theta_1) = 0 \tag{1.a}$$

$$m_2L_2\ddot{\theta}_2 + m_2L_1\ddot{\theta}_1 cos(\theta_1-\theta_2) - m_2L_1\dot{\theta}_1 sin(\theta_1-\theta_2) + m_2gsin(\theta_2) = 0 \tag{1.b}$$

These ODEs can be solved numerically with initial values for the angles defined between each individual pendulum and the vertical axis, θ_1 and θ_2, their first time derivatives, $\dot{\theta}_1$ and $\dot{\theta}_2$, and with the values of the lengths of the rods and the masses of the blobs.

The motion of the double pendulum can be entirely described by the two angles, which are the two degrees of freedom of the system, together with the parameters and initial conditions. The angles θ_1 and θ_2 can be

used in a mathematical representation of the system in Cartesian coordinates, which allow for a simpler understanding of its state than the one that can be inferred from equation (1).

$$x_1 = L_1 sin(\theta_1) \tag{2.a}$$
$$y_1 = -L_1 cos(\theta_1) \tag{2.b}$$
$$x_2 = L_1 sin(\theta_1) + L_2 sin(\theta_2) \tag{2.c}$$
$$y_2 = -L_1 cos(\theta_1) - L_2 cos(\theta_2) \tag{2.d}$$

For small values of the initial angles and ratios between the parameters close to the unit (i.e., $m_1/m_2 \approx 1$), the system exhibits the beat phenomenon, and falls within a non-chaotic regime. The system is still sensitive to the initial conditions, but since the energy is low, the parameters of the system can be varied without significantly altering the outcome. Outside the regime of small energies, the system exhibits chaotic behaviour, and the available range of variation of both the parameters and the initial conditions is notably reduced.

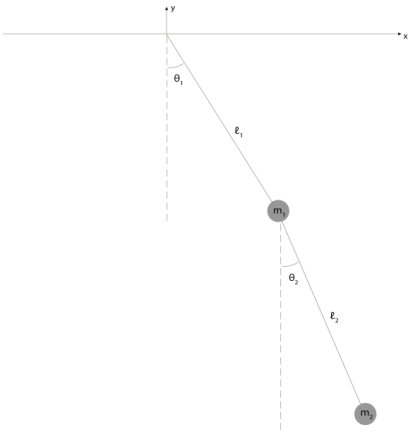

Figure 1: Double pendulum. The masses are labelled m_1 and m_2; the lengths of the rods are labelled L_1 and L_2; and the angles θ_1 and θ_2 [8].

2 Method

2.1 Modelling of the physical reasoning into a neural network architecture

The objective of this project is to progress in the direction of an artificial intelligence that is able to solve physics problems from experimental data without being biased by previous knowledge in physics. This is achieved by focusing on the human modelling process of a physics problem, rather than on a specific problem. The modelling process is later translated into the architecture of a NN. The specifics of this translation fall beyond the scope of this project, and so will not be discussed in depth, although a general understanding is imperative.

The first step is to consider a simplified version of the modelling process of a certain physical problem. In order to understand an event or a physical system, a physicist first needs to observe it. The physicist does not work with the information gathered from an observation directly, but with a mathematical representation of the considered system. The representation requires choosing adequate parameters in order for it to be both able to make predictions based on the acquired knowledge of the system and for doing so efficiently. Formally, the compression of the observation ($\mathcal{O}$) into a representation ($\mathcal{R}$) is called *encoding* (E : $\mathcal{O} \rightarrow \mathcal{R}$), while the process of interpreting the representation to answer a question ($\mathcal{Q}$) about the system is known as *decoding* (D : $\mathcal{R} \times \mathcal{Q} \rightarrow \mathcal{A}$) [3].

The modelling process can be translated into a NN architecture. The main body of the NN is formed by the encoder (E) and the decoder (D), which are implemented as feed-forward NN, that is, the input and the output do not form a cycle. The encoder uses the observation as an input and outputs the latent representation, while the decoder uses the latter as an input together with the question and outputs an answer that can be interpreted [9-11].

During the training, we provide the observation to the encoder in the form (o, q, $a_{corr}(o,q)$), where o is the observation, q is the question and $a_{corr}(o,q)$ is the correct answer to the question given the observation. The encoder uses the information from the observation to parameterize the system into what is called a latent representation. The latent representation must not be imposed externally, and so the encoder must be completely free to choose whatever representation it wants, since one of the questions that this project attempts to answer is whether or not the NN is able to correctly parameterize the physical system.

A proper parameterization, i.e. a proper latent representation, of a physical system is considered to be the one that stores the minimal amount of information that allows to correctly answer the given questions $\mathcal{Q}$. In the case where the physically relevant parameters of the system change over time, an update rule is implemented. In this case, instead of a unique latent representation, many are generated from the initial representation, each having an identical decoder and time evolution steps as well as being trained simultaneously. Additionally, it is required that the latent neurons (LNs) are independent from each other, implying that the parameters that define the physical system are not correlated and independent from one another. This restriction motivates the NN to choose a representation that stores one parameter per LN, which coincide with the underlying degrees of freedom of the system.

The decoder uses the representation to answer the question and produces an answer that can be interpreted. Thus, the actual representation is irrelevant as long as it allows the decoder to answer the question.

2.2 Model of the double pendulum

SciNet is a NN architecture built to be able to solve classical- and quantum-mechanical physical problems with minimal previous information about the considered system. In this project, we model the double pendulum to try and see if *SciNet* is able to adequately parameterize the system and to gain insight in the nature of the problem.

As mentioned in the previous section, the NN requires an observation and a question as inputs, an output and a certain number of LNs to function. It is also required that we give a size to the encoder and the decoder, but their specifics are not relevant as long as they are big enough as to not constrain the network and small enough as to be efficiently trainable [3]. The motion of the double pendulum can be described exclusively with the angles that each individual pendulum forms with the vertical axis, θ_1 and θ_2. Thus, as an input, a time series of 50 values of each angle in an interval between 0 and t is given to the NN, which is then asked for a prediction of their value until $t' = 2t$ (or $t' = 4t$ in section 3.3). Similarly to the input size, the output size is also 2, since the an-

swer to the posed question is the value of each angle at a specific time. These values are summarized in Table I.

Number of free parameters	Observation input size	Question input size	Number of latent neurons	Output size	Encoder	Decoder
2	50	2	3	2	[500,100]	[100,100]
4	50	2	5	2	[500,100]	[100,100]

Table I. Network structure of the three different cases considered in the project. Both the encoder and the decoder contain two hidden layers, with n_1 and n_2 neurons, respectively. The notation used in this table depicts these values in the format $[n_1,n_2]$.

The parameters of the system are the initial angles and their first time derivatives, the lengths of the rods and the masses of the pendulums. Given the high sensibility of the double pendulum to the variation of these parameters, it is interesting to consider different network structures in order to see if it affects the generalization capabilities of the network outside of the non-chaotic regime. However, in hopes of slightly simplifying the calculations, the initial angles and their first time derivatives are kept fixed at all times, and are thus not considered when talking about parameters. Specifically, the first time derivatives are both initially set to 0 for every resolution of the problem.

First, a network structure consisting of only 3 LNs is chosen. In this case, the lengths are kept constant (this choice is arbitrary, any pair of parameters could've been chosen), leaving thus only two free parameters. It is then expected that *SciNet* assigns one neuron to each of these parameters and leaves the third neuron unused. For the second case, which has all four parameters be variable, a network structure consisting of 5 LNs is chosen. Similarly to the previous case, it is expected that *SciNet* assigns each of the first four neurons to each parameter, having the fifth one not provide any additional information about the system. Later, the time interval of the prediction is increased to test if the neuron usage in both cases remains the same or a new parameter becomes relevant.

The specifications of the training of the NN can be seen in Table III, present in Appendix A, along with a discussion on the convergence of the loss function, which is used to determine when the network is sufficiently trained.

3 Analysis of the results

3.1 Time evolution

The double pendulum is a dynamical system with a complex behaviour, caused by a high sensitivity to the initial conditions. The motion of the pendulum can be described by the angles that each rod forms with the vertical axis, θ_1 and θ_2, which follow equation (1). For small values of these angles, the system exhibits a non-chaotic behaviour. In this regime, the pendulum will hang almost vertically, describing a periodic motion that differs only slightly from that of a single pendulum. Outside of this regime, the system exhibits chaotic behaviour, which becomes less predictable the further we stray from the smaller angles.

The first relevant study that can be done is to check whether or not *SciNet* is able to correctly predict the motion of the double pendulum in both regimes for 10s (with training data going up to 5s). Of course, the predictions in the different regimes have varying ranges of generality, which will be addressed in a future section.

I. Non-chaotic regime

This regime considers both θ_1 and θ_2 to be small angles, i.e. under 15 degrees (or $\pi/12$), and the parameters to have ratios that do not differ much from the unit. For smaller angles, these differences are barely noticeable, but gain importance as the angle is increased. In this specific case, the angles are chosen to be:

$$\theta_1 = \frac{\pi}{30} \qquad \theta_2 = \frac{\pi}{17},$$

while the free parameters are always kept close to 1. Using these angles, *SciNet* was able to solve the equations and properly predict the time evolution of the double pendulum. Figure 2 shows the time evolution of a two different cases of the pendulum, with different amounts of free parameters. The first case, represented in Figure 2 (left), considers two free parameters and two fixed ones. In this case, the fixed parameters are arbitrarily chosen to be the lengths. The second one, represented in Figure 2 (right), considers all four parameters to be free within a determined range. The boundaries in the ranges of the free parameters are notably small which, as will be discussed in section 3.4, is a limitation derived from the nature of the problem.

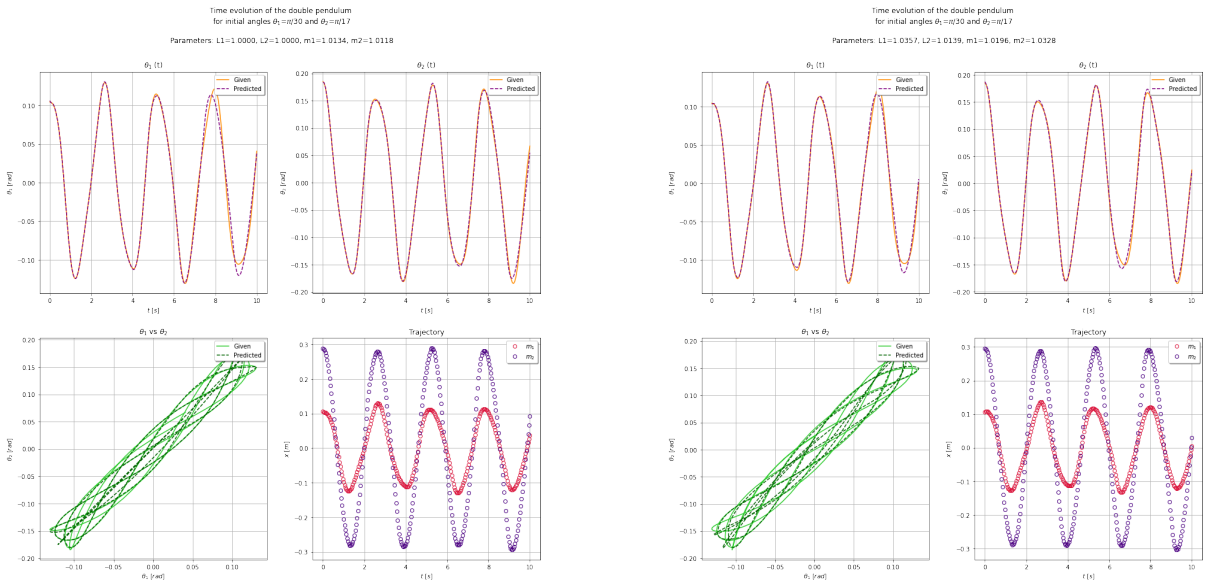

Figure 2: Time evolution of the double pendulum in the non-chaotic regime. The figure on the left (2.1) is the prediction with two free parameters (the masses) and two fixed parameters (the lengths), while the figure on the right (2.2) shows the prediction with four free parameters (the masses and the lengths). Inside every figure there's four subplots, that can be named using notation *a* to *d*. From top to bottom and left to right they are: time evolution of θ_1, time evolution of θ_2, θ_1 vs θ_2 and the trajectory in Cartesian coordinates.

The NN is able to predict the time evolution of the double pendulum with good accuracy for the full 10 seconds

it is asked to do so, as seen in Figures 2.1.a, 2.1.b, 2.2.a and 2.2.b. Nonetheless, the small deviations from the real solution are easily observable in plots 2.1.c and 2.2.c.

II. Chaotic regime

Outside of the range of small angles and similar valued parameters, the double pendulum exhibits its characteristic chaotic behaviour. In this regime, a small change in the initial conditions has a huge impact on the trajectory of the pendulum. This effect increases with the initial values of the angles, θ_1 and θ_2. In this specific example, the angles are chosen to be:

$$\theta_1 = \frac{\pi}{2} \qquad \theta_2 = \frac{8\pi}{15}$$

while the free parameters of the systems are kept close to the unit. However, it must be noted that the range of values that the free parameters of the system can take is more restricted than it is in the non-chaotic regime. Considering this restriction, *SciNet* was able to predict the trajectory of the double pendulum for 10s, achieving similar results to those shown in the previous section. Figure 3 plots the results in an analogous format to that of Figure 2.

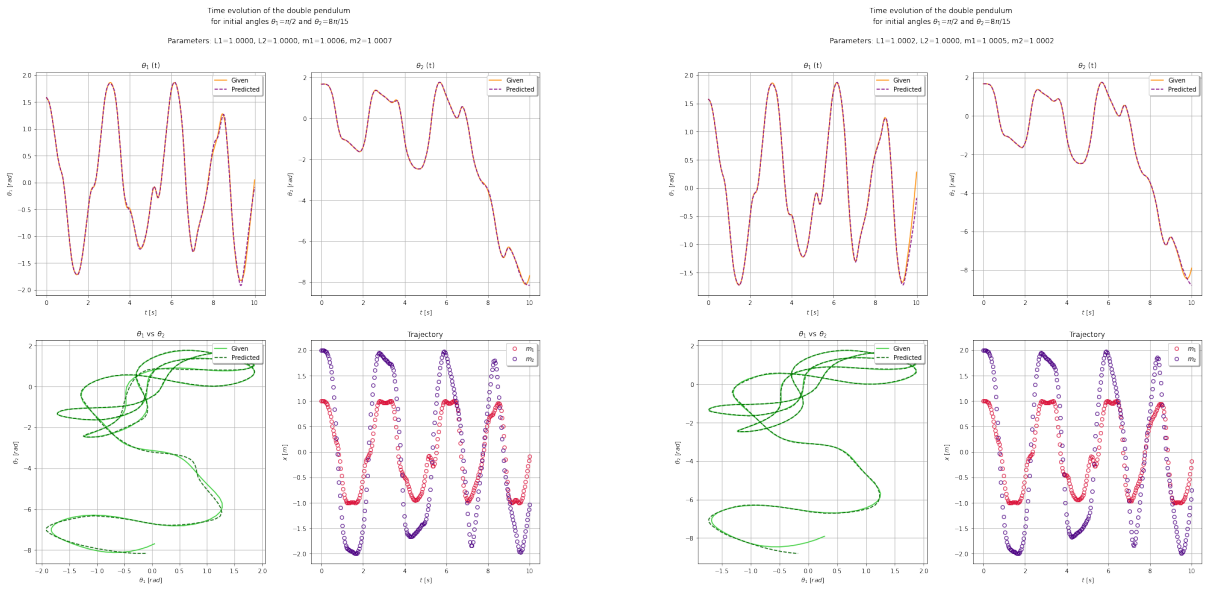

Figure 3: Time evolution of the double pendulum in the chaotic regime. The figure on the left (3.1) is the prediction with two free parameters (the masses) and two fixed parameters (the lengths), while the figure on the right (3.2) shows the prediction with four free parameters (the masses and the lengths). Inside every figure there's four subplots, that can be named using notation *a* to *d*. From top to bottom and left to right they are: time evolution of θ_1, time evolution of θ_2, θ_1 vs θ_2 and the trajectory in Cartesian coordinates.

One might notice that the NN seems to show better results for time evolution predictions on the chaotic regime than for those on the non-chaotic regime. This is a direct consequence of the aforementioned restriction on the range of values on the free parameters of the system. Specifically, this range is two orders of magnitude smaller in the chaotic regime, as can be seen in Table II, in section 3.4.

3.2 Latent neuron representation

The main goal of this project was not only to check whether or not *SciNet* is able to predict the time evolution of a chaotic system, but also if it is able to adequately parameterize the system with a minimal number of LNs. Drawing a parallel to the damped pendulum example developed in [3], *SciNet* should use a LN per free parameter of the system.

Figures 4 and 5 show the LN representation of the NN for both cases discussed in the previous section (two and four free parameters) for both the non-chaotic and chaotic regimes. In all cases, the NN is given one LN per free parameter, plus an additional one. According to the exposed hypothesis, based on an extrapolation from previous results, the NN should not use the additional LN. In case all given neurons are used, it might be that *SciNet* is not identifying the relevant parameters of the system, but rather fitting an n-order polynomial, in order to represent the given problem.

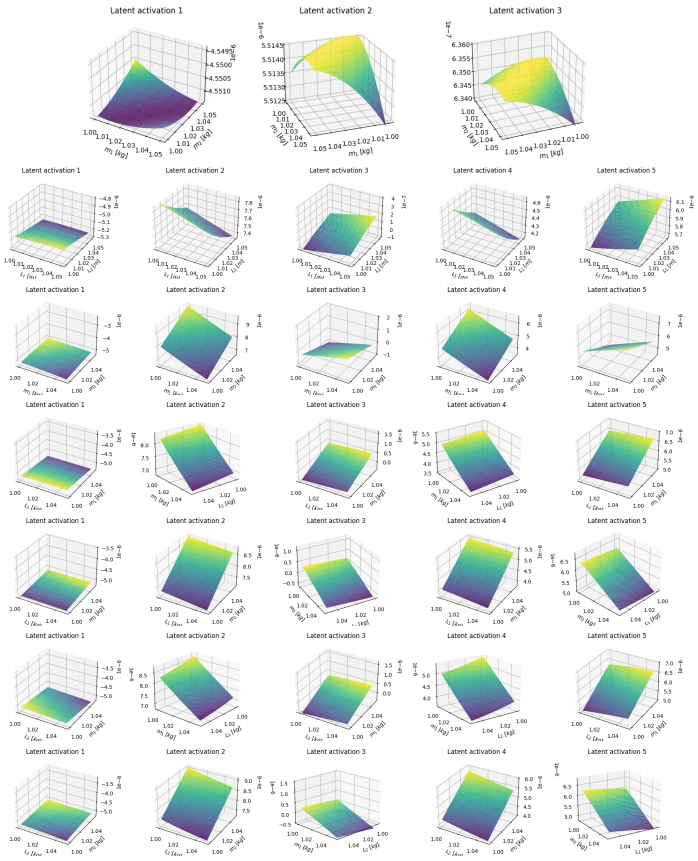

Figure 4: LN representation for the two cases in the non-chaotic regime at t=7.5s. The top figure (4.1) is the representation for only two free parameters (the masses), while the bottom ones (4.2-4.7) show the representation for the case of four free parameters. Specifically, Figure 4.2 plots the LN representation of L_1 and L_2, and 4.3 does the same with the masses. Figures 4.4-4.7 plot the representations of L_1 with m_1, L_1 with m_2, L_2 with m_1 and L_2 with m_2, respectively. The formatting of these plots is discussed in Appendix B.

As seen in Figure 4, the NN is able to properly parameterize the system for both cases in the non-chaotic regime. It is to be noted that the unused LN is not necessarily the third, as the NN is, by design, free to choose them as it sees fit. This can be seen in both cases represented, where *SciNet* leaves the first neuron unused and uses the other ones to parameterize the system. For the case with only two free parameters, the plots show that the network stores parameters that are a linear combination of the masses of the pendulums. With four free parameters, the parameterization is analogous, but with both masses and lengths.

SciNet is also able to parameterize the double pendulum in the chaotic regime, and does so with linear combinations of the free parameters. In both cases shown, however, there is one neuron with much higher activation values than the other ones. For the case of two free parameters, this neuron is the second one, while for the case with four free parameters it happens with the fourth one. Thus, in order to be able to distinguish if the NN uses only as many LNs as free parameters are in the system, this LN was not considered when formatting according to what is explained in Appendix B. With this choice, it is clear in Figure 5.1 that for the case with only two free parameters the first neuron is unused. This is not the case with the case with four free parameters.

Using the actual values of the activation, the results suggest that the second LN from the left is the unused

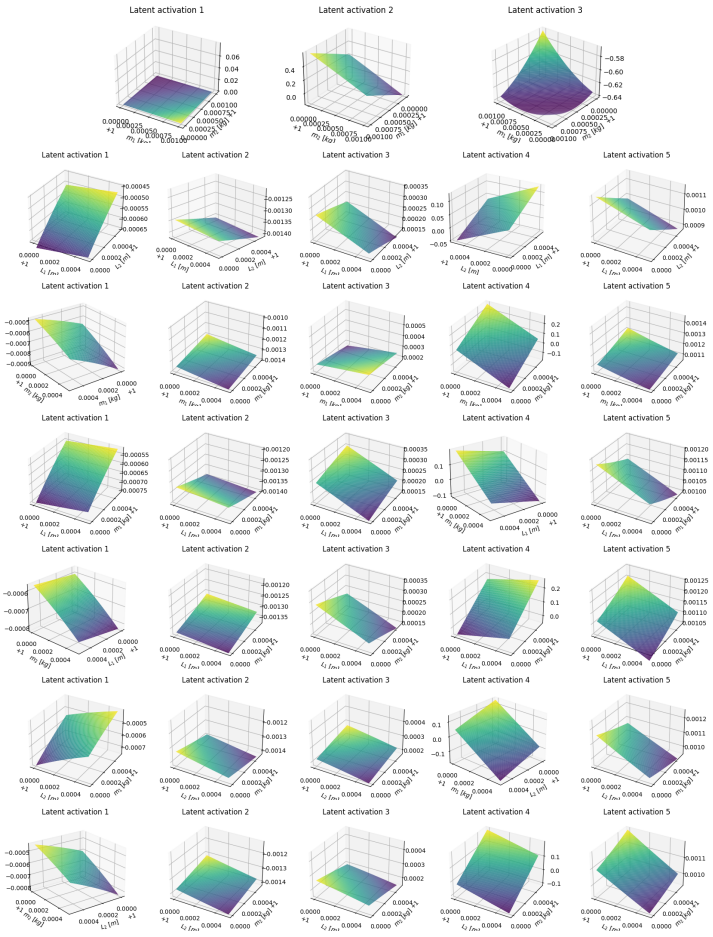

Figure 5: LN representation for the two cases in the chaotic regime at t=7.5s. The top figure (5.1) is the representation for only two free parameters (the masses), while the bottom ones (5.2-5.7) show the representation for the case of four free parameters. Specifically, Figure 5.2 plots the LN representation of L_1 and L_2, and 5.3 does the same with the masses. Figures 5.4-5.7 plot the representations of L_1 with m_1, L_1 with m_2, L_2 with m_1 and L_2 with m_2, respectively. The formatting of these plots is discussed in Appendix B.

neuron, but there is not much difference with the middle one (the range of activation values second LN from the left is either half an order of magnitude smaller or equal to that of the middle one). This difference can be seen clearly in Figures 5.2, 5.4 and 5.6.

3.3 Long exposure

Given a certain amount of free parameters, *SciNet* requires a network structure consisting only of that many LNs in order to correctly parameterize the system and predict its time evolution. The NN assigns each free parameter to a LN, leaving unused any extra neurons. The chaotic nature of the double pendulum might, however, cause this not to be true over larger periods of time. It is possible that, if asked to predict the time evolution of the system for a longer span of time, the NN requires an additional LN (or maybe more) in order to properly parameterize the system.

Using the same cases as in section 3.1, a longer time evolution prediction is asked to *SciNet*, and the LN usage is then checked. Specifically, the NN is asked to predict the time evolution of the double pendulum for 20s with training data up to 5s.

I. Non-chaotic regime

In this regime, the NN is able to predict the time evolution of the system for up to 19s, approximately. This is the case for both two and four free parameters, as shown in Figure 6. Figure 7, on the other hand, shows the

LN representation for these two cases.

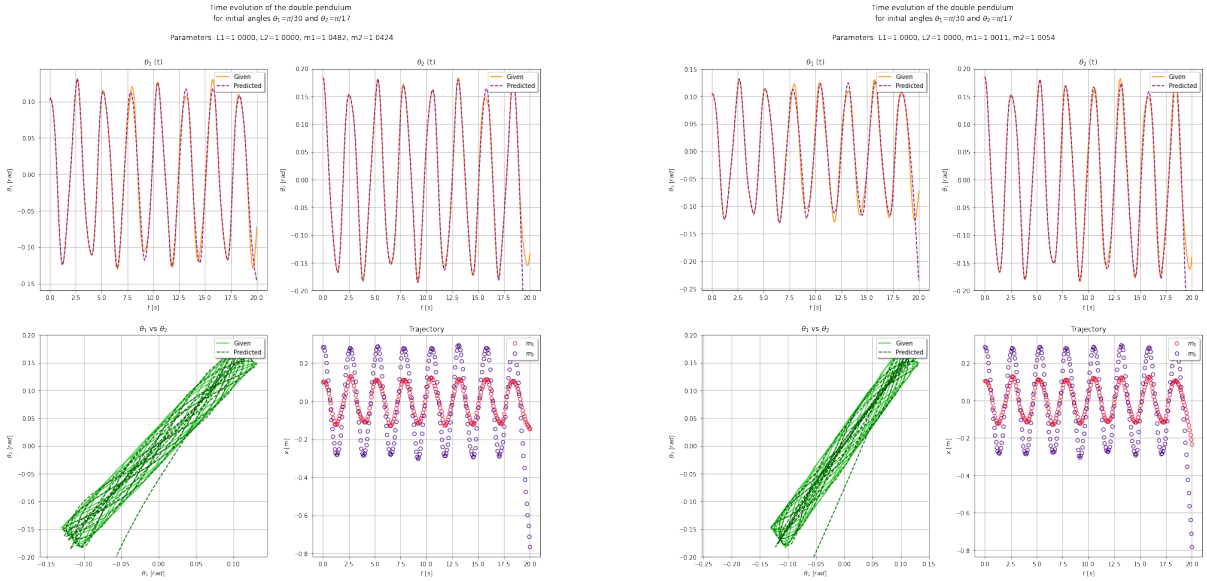

Figure 6: Time evolution of the double pendulum in the non-chaotic regime for 20s. The figure on the left (6.1) is the prediction with two free parameters (the masses) and two fixed parameters (the lengths), while the figure on the right (6.2) shows the prediction with four free parameters (the masses and the lengths).

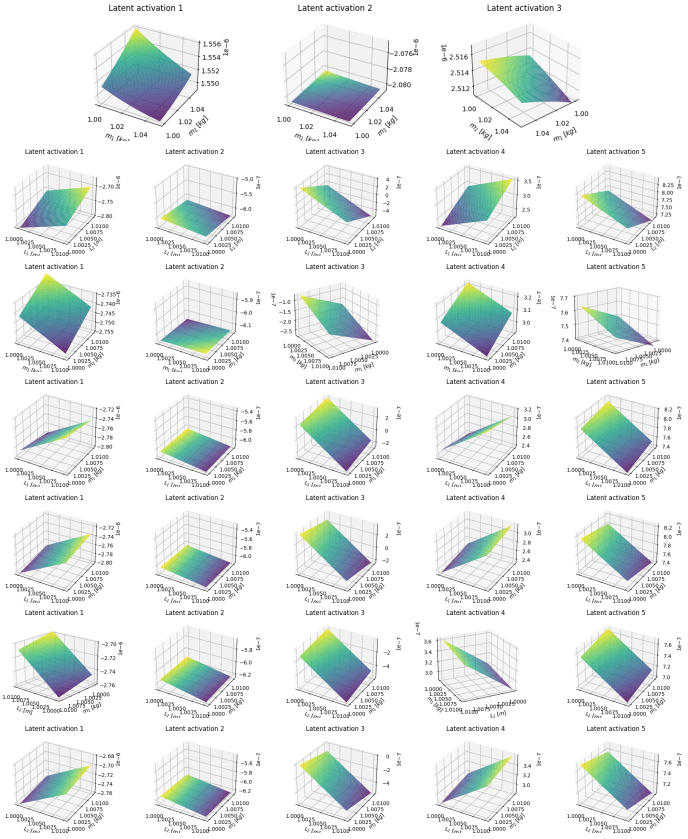

Figure 7: LN representation for the two cases in the non-chaotic regime at t=7.5s. The top figure (7.1) is the representation for only two free parameters (the masses), while the bottom ones (7.2-7.7) show the representation for the case of four free parameters.

The NN parameterizes the system using a linear combination of the free parameters, leaving one unused neu-

ron in both cases. Similarly to the chaotic regime of the previous case (time evolution prediction to 10s), however, the LN activation is not of the same order of magnitude in all the used LNs. For the case with only two free parameters, this difference is easily noticeable, as it is of two orders of magnitude. For both cases, the first neuron (in the two free parameter case) and the third neuron (in the four free parameter case), were not considered when applying the formatting process described in Appendix B.

II. Chaotic regime

In the chaotic regime, the complexity of the problem scales significantly. The NN requires a much longer training in order to solve the problem, as well as an even smaller range of values of the free parameters (which is so small that the problem is close to being a single case, as seen in Table II). With these conditions, *SciNet* was both unable to converge and correctly predict the time evolution of the pendulum for the case with two free parameters in a reasonable time. The case with four free parameters was not computed as a result of this.

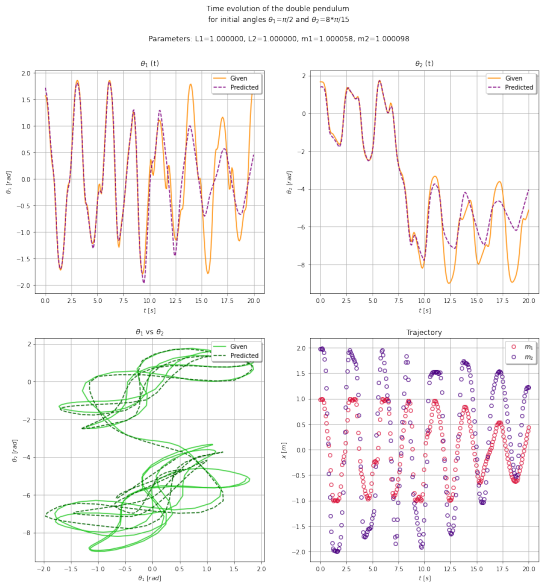

Figure 8: Time evolution of the double pendulum in the chaotic regime for 20s for the case with two free parameters.

Figure 8 shows the best prediction obtained in this case, which goes until 12.5s approximately. Both more computational power and a smaller range of values for the free parameters are necessary to obtain a full solution.

3.4 Generality

The long-term goal of designing an artificial intelligence physicist requires the ability to solve a problem given any set of initial and boundary conditions. In order to achieve a general solution, it is usually necessary to sacrifice performance. However, the problem at hand is very susceptible to changes to the initial conditions. This forces a certain sacrifice of generality in pursuit of acceptable computation times and data volumes.

In this section, the loss of generality caused by varying the number of free parameters, the range of values for said parameters, the initial angles and the prediction time is studied in order to achieve an adequate level of generality with acceptable computation times (five training phases, as described in Table III) and data volumes (100.000 data samples with 50 equally spaced values each).

I. Number of free parameters

The double pendulum has a total of four initial conditions: the initial angles, θ_1 and θ_2, their first time derivatives, $\dot{\theta}_1$ and $\dot{\theta}_2$; and four parameters: the masses, m_1 and m_2, and the lengths, L_1 and L_2. In this project, the initial angles and their first time derivatives are kept constant at all times in order to simplify the resolution. Thus, only the parameters are considered for this section.

By taking a look at Table III, displayed in Appendix A, it can be seen that the required training for an acceptable prediction is, in the non-chaotic regime, longer for the case with four free parameters. In the chaotic regime, however, the situation is the opposite. The NN requires longer training times to adequately converge in this case. There is, however, something to be taken into account. In all cases but for the non-chaotic regime with a prediction up to 10s, the range of values for the free parameters is reduced when the number of parameters is increased, otherwise the network is unable to converge while giving an adequate prediction for the time evolution. Therefore, an increase of the number of free parameters results in either longer training times or a sacrifice in the range of values that the parameters can take. With the discussed exception, the second option was chosen in this project. The mentioned reduction can be seen in Table II.

Regime	Time evolution prediction (s)	Range of values	
		2 free par.	4 free par.
Non-chaotic (NC)	10	[1,1.05]	[1,1.05]
	20	[1,1.05]	[1,1.01]
Chaotic (C)	10	[1,1.001]	[1,1.0005]
	20	[1,1.0001]	-

Table II. Loss of generality, seen as a decrease in the range of values that the free parameters can take, caused by an increase in the complexity of the problem.

II. Range of values for the free parameters

The results displayed in sections 3.1-3.3 were obtained with the range of values for the free parameters displayed in Table II. Using these ranges, the network was able to obtain a converged solution in a reasonable time. In the chaotic regime, these ranges cannot be increased, otherwise a time evolution prediction is not possible. Using small initial values for the angles, however, the range can be increased and a reasonable prediction can be obtained. It must be noted that these predictions do not meet the convergence criteria discussed in Appendix A. This is intentional, as the objective of this section is to illustrate the difference in the accuracy of the predictions made by the NN with different ranges of values for the free parameters while using the exact same training. Figure 9 shows these differences for $\theta_1 = \pi/30$ and $\theta_2 = \pi/17$ with two free parameters and a time evolution prediction of 10 seconds.

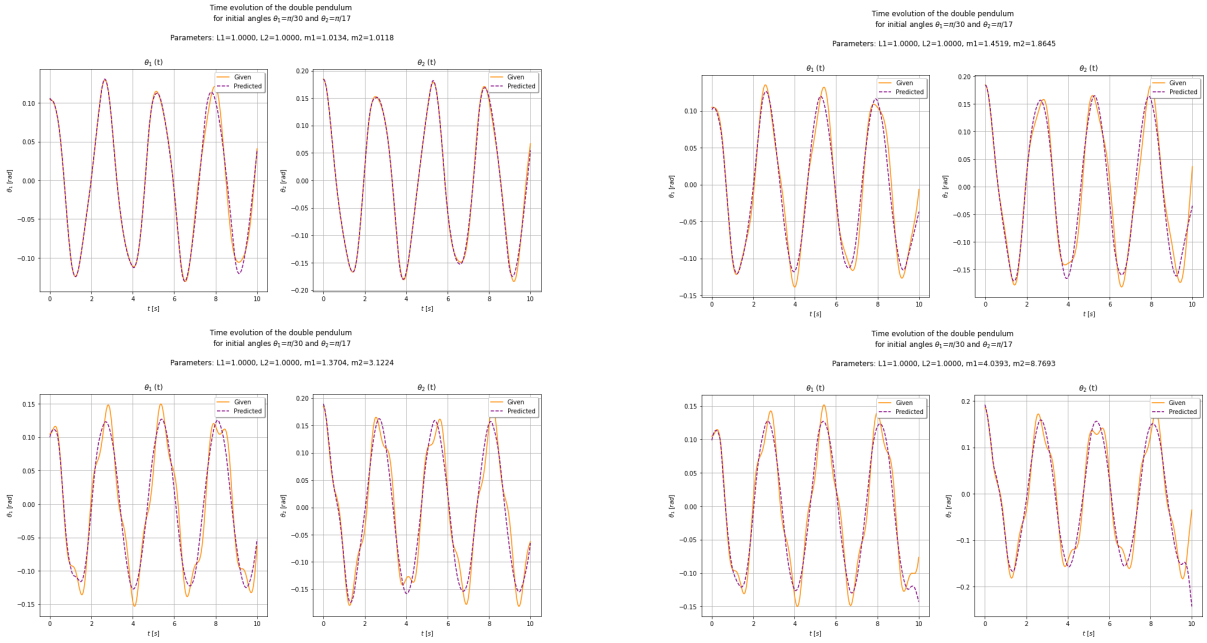

Figure 9: Time evolution prediction for different ranges of values of the free parameters (in this case, the masses) with initial angles $\theta_1 = \pi/30$ and $\theta_2 = \pi/17$. The top left figure (9.1) is the same one as the one plotted in Figure 4.1, but only 4.1.a and 4.1.b are shown. The ranges of the parameters are, from left to right and from top to bottom, [1,1.05], [1,2], [1,4], [1,10].

At first glance, it seems as that the predictive power decreases as the range of the parameters increases. Figure 9.2 shows that the case with range [1,2] requires more training to get an accurate prediction, but the given solution is overall correct. This can also be the case for the cases with ranges [1,4] and [1,10], but the purple curves in Figures 9.3 and 9.4 suggest that the NN might be fitting a polynomial to give an acceptable solution. To determine whether this is the case or not, it is necessary to check the LN representation.

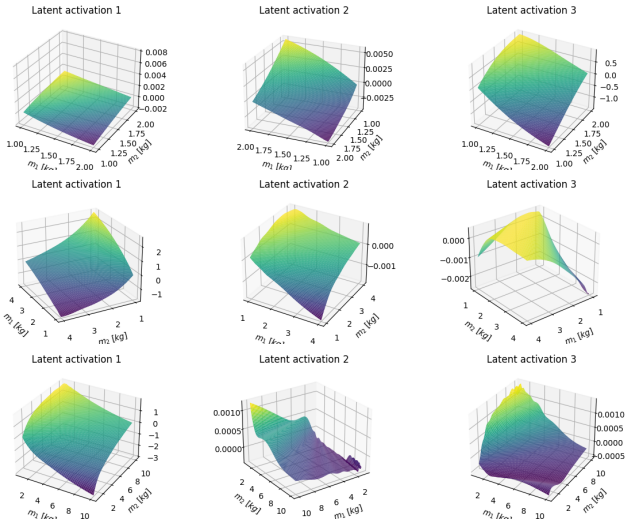

Figure 10: LN representation for the case with two free parameters in the non-chaotic regime at t=7.5s, with varying ranges of the free parameters. From top to bottom (10.1-10.3), the figures represent the cases with ranges [1,2], [1,4] and [1,10], respectively.

Similarly to the cases displayed in Figures 5 and 7, Figure 10.1 shows that there is one LN with a much higher activation than the other two (two orders of magnitude bigger than the biggest of the other two). Of the remaining ones, the middle one has an activation one order of magnitude bigger than the first one. Thus, it can be said that the NN adequately parameterizes the system using linear combinations of the free parameters. For the cases in Figures 10.2 and 10.3, even though there is still one LN with a bigger activation, there is next to no difference in the activation values of the remaining two LN. This suggests that the NN is indeed using a polynomial fit to represent the system. Figure 11 shows the loss functions of all four cases, showing that all had converged according to the convention proposed in Appendix A.

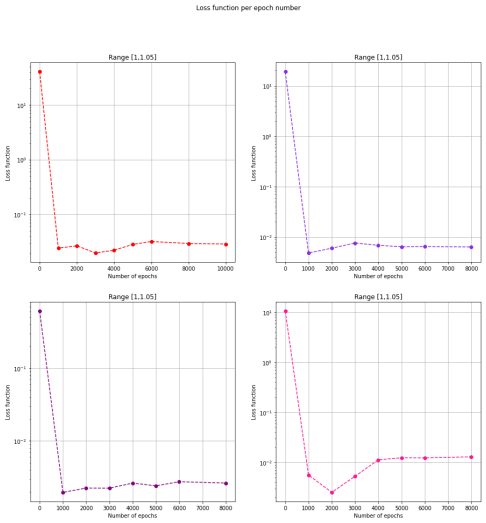

Figure 11: Loss function as a function of the number of epochs for the four cases shown in Figure 9, in the same order. Each dot in the plot represents a training phase.

III. Initial angles

Both the initial values of the angles and the ratios between the parameters of the system have a substantial impact on the trajectory of the double pendulum. In this project, the ratios between the parameters were kept close to 1 in order to ensure the convergence of the calculations for bigger values of the angles and keep the computation times as small as possible for the case of small angles, as seen in the previous section.

However, keeping these ratios close to the unit is not sufficient in order to ensure convergence. As seen in Table II, increasing the initial values of the angles requires a reduction of the ranges of values for the free parameters up to a point were nearly all generality is lost. For the angles used in the chaotic regime, these range is reduced to a variation of the order of 1e-3 and 5e-4.

4 Conclusions

The objective of this project was to use a NN to find the relevant parameters and predict the future state of a double pendulum with a minimal amount of assumptions imposed on the network. Additionally, the aim was to generalize the problem as much as possible. Given the inherent complexity of the problem, a full generalization was not possible, and so different cases were studied separately in order to understand the capabilities of the network in different regimes.

Taking fixed initial values for the angles, the NN was able to predict the time evolution of the system for 10s with a training lasting only up to 5s, for both small ($\theta_1 = \pi/30$ and $\theta_2 = \pi/17$) and big angles ($\theta_1 = \pi/2$ and $\theta_2 = 8\pi/15$) with two and four free parameters. The predictions of the cases in the chaotic regime required longer training times, as well as smaller ranges of values of the free parameters, in order to converge in an adequate computation time. Additionally, the network was able to predict the time evolution of the system for 20s with the training also lasting up to 5s for the non-chaotic regime, but was unable to do so within the chaotic regime.

The prediction of the trajectory of a double pendulum by a NN in and of itself is not of much use. As mentioned earlier, the correct parameterization of the system via identification of the relevant parameters of the double pendulum is what lets this project be a step in the direction of an artificial intelligence that is able to solve complex physics problems. For all the cases described, *SciNet* was able to properly parameterize the double pendulum, using only one LN per free parameter of the system. The parameterization itself consists of a linear combination of said parameters.

The complexity of the pendulum arises mainly from its chaotic nature. The major complications faced during this project were a manifestation of this. First of all, the time evolution prediction of the system for 20s in the chaotic regime was impossible within a reasonable time. Additionally, the NN experienced a loss of generality in this regime, seen mainly as a decrease in the available range of values of the free parameters.

In the non-chaotic regime, this nature also posed problems. When attempting to increase the range of values of the free parameters, the network was only able to make accurate predictions for ranges that allowed the ratios of the free parameters to be of a maximum of 2 to 1. For bigger ranges, the NN tried to fit a polynomial in order to solve the problem. Therefore, in order to gain generality, longer computation times and bigger amounts of data are necessary, which translates into more computational power.

Acknowledgements

This project would not have been possible without the efforts of Rikard Enberg, Eliel Camargo-Molina and Harri Mikael Waltari. All three of them set aside huge amounts of time from their busy schedules to help in both the computational and the physical parts of this problem. Additionally, I would like to thank R. Iten et al. both for their work with *SciNet* and the access to the source code.

Appendix A: Training of the neural network and convergence of results

As discussed in section 3.4, the variation of certain aspects of the problem (namely the number of free parameters and their range of values, and the initial values of the angles) increases the complexity of the problem. This in turn increases the computation time that is required for the NN to properly train. Therefore, the different cases discussed in this project required different specifications for the training. Usually, this meant longer training times, but in some cases other parameters needed to be adjusted. Table III shows the specific values of each of the relevant parameters for the training (batch size, learning rate and number of epochs) for each of the considered cases, along with the number of training phases required.

Regime	Number of free parameters	Time evolution prediction (s)	Training phases	Batch size	Learning rate	Number of epochs
Non-chaotic (NC)	2	10	7	512 (3) 1024 (3) 2048	1e-3 (3) 1e-4 (3) 1e-5	1000 (6) 2000
		20	8	512 (3) 1024 (4) 2048	1e-3 (3) 1e-4 (4) 1e-5	1000 (7) 2000
	4	10	10	512 (3) 1024 (5) 2048 (2)	1e-3 (3) 1e-4 (5) 1e-5 (2)	1000 (8) 2000 (2)
		20	9	512 (3) 1024 (4) 2048 (2)	1e-3 (3) 1e-4 (4) 1e-5 (2)	1000 (7) 2000 (2)
Chaotic (C)	2	10	13	512 (2) 1024 (4) 2048 (7)	1e-3 (2) 1e-4 (4) 1e-5 (7)	1000 (6) 2000 (7)
		20	17	512 (4) 1024 (6) 2048 (7)	1e-3 (4) 1e-4 (6) 1e-5 (7)	1000 (10) 2000 (7)
	4	10	10	512 (3) 1024 (4) 2048 (3)	1e-3 (3) 1e-4 (4) 1e-5 (3)	1000 (7) 2000 (3)

Table III. Specific parameters used during the training for every specific case discussed in the paper. The batch size increased progressively, while the learning rate decreased progressively, this meaning that neither of them took previous values again once changed. The numbers inside the parentheses indicate the amount of training phases these values were used for.

It must be noted that the case in the chaotic regime with four free parameters with a prediction up to 20s was not included because a satisfying solution was not achieved in an acceptable time for the case with only two free parameters (which suggested that this case, much more complicated, would also be impossible within reasonable computation times).
In order to determine whether or not the NN was sufficiently trained, the loss function was evaluated. In this project, the NN was trained until the loss function showed a variation smaller than 0.001 between training phases. Given the way neural networks function, it is possible for them to reach a local minima during the training, in which the loss function does not vary significantly (thus satisfying the imposed condition for a successful training), but the prediction of the time evolution (namely, the answer to the question) is not accurate. In order to avoid accepting such cases as valid, time evolution prediction plots must be checked. If the NN is unable to predict the evolution of the system for the asked time, additional training is required.

The different plots in Figure 12 show the loss function for each of the studied cases. In all of the plots, it can be noted that the number of epochs does not coincide with the one displayed in Table III. This is because Figure 12 plots the loss function of an additional training phase, which is the one that does not differ significantly from the previous one, and is therefore unnecessary. For example, for the case of non-chaotic with two free parameters with a time evolution prediction of 10s, Table III mentions only 8000 epochs, but Figure 12 shows 10000 epochs. The additional epochs correspond to the 8th training phase, which has a loss function that differs less than 0.001 with respect to that of the 7th one.

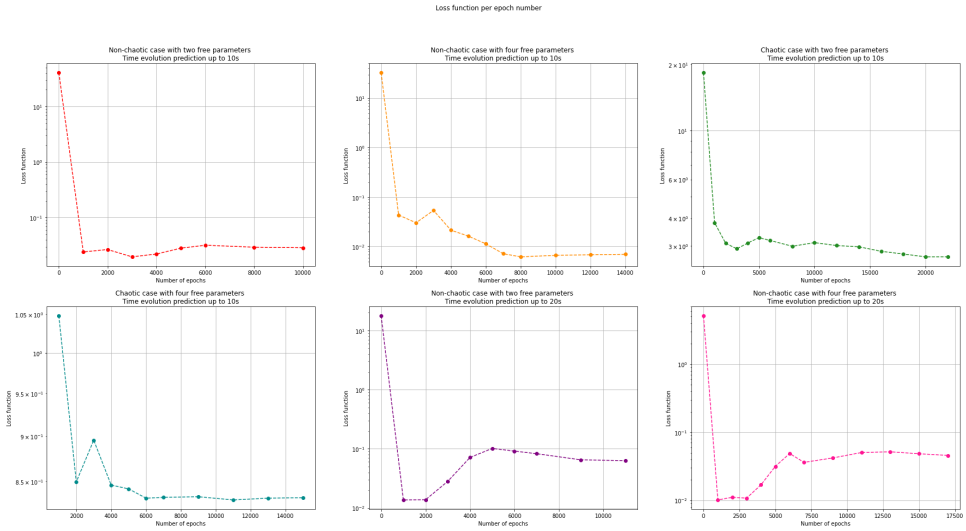

Figure 12: Loss function as a function of the number of epochs for all cases in section 3.1 and the ones in the non-chaotic regime of section 3.3. From left to right and top to bottom, the figures are labelled 12.1-12.6. In this order, they plot the cases non-chaotic with two free parameters for 10s, non-chaotic with four free parameters for 10s, chaotic with two free parameters for 10s, chaotic with four free parameters for 10s, non-chaotic with two free parameters for 20s and non-chaotic with four free parameters for 20s.

Appendix B: Latent neuron representation format

The LN representation plots allow us to get insight into how the NN is parameterizing the system. The raw data is, however, not very illustrative in this regard. Therefore, some formatting is necessary. First, the LN representation is plotted in order to check the range of activation values of each LN. The LN with the maximum range is identified (x_{max}), and the representation is plotted again, now inside the range of activation values determined by this LN. Although the used LNs have similar ranges of activation values, their actual values are not necessarily close to each other. Thus, a LN that takes values within the range (a_0,a_1) is represented in the range (a_0,a_0+x_{max}) [10,11]. This format allows for a better visualization of the representation, simplifying the task of identifying the unused LN. Figure 11 shows the difference between the raw (top) and the formatted (bottom) representation.

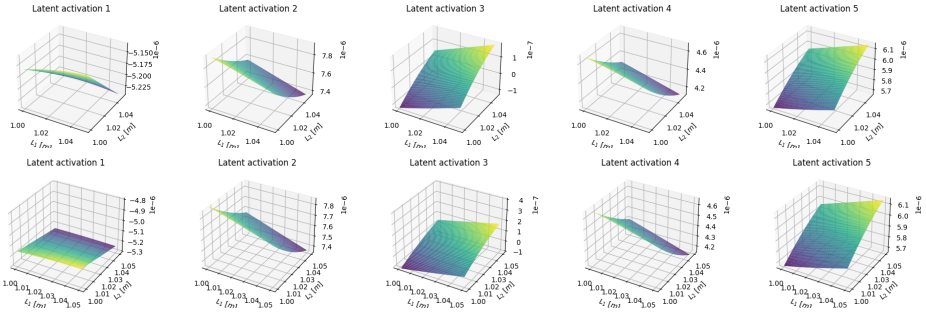

Figure 13: LN representation of L_1 and L_2 for the non-chaotic regime with four free parameters at t=7.5s. The top plot displays the raw data, where identification of the unused neuron is complicated at first glance. After the format, the bottom plot is obtained, where the identification is simplified.

Appendix C: Source code

The source code for this project was written in Python 3, adapting the model for the damped pendulum used in [3] for the double pendulum.

The code is divided in two main parts: the data generation and the model. The data generation file solves equation (1) N (= 100000) times for different initial values of the parameters. The equation is solved between $t_0 = 0s$ and t_{max} (= $5s$), taking t_{int} (= 50) steps. Two sets of data are generated with this calculation, one that goes up to t_{max}, which will be used for the training, and another that goes up to the predicted time, which will be used for testing the NN. The data is then saved in a format akin to the one seen in the heliocentric model of the Solar System from [3] in order for it to fit structure of the model for the damped pendulum. This model is used for the training and testing of the NN.

The link below gives access to the GitHub repository were the source code can be found, along with the interactive program (written in Jupyter) that allows for a training of the NN and a further analysis of the time prediction capabilities of the network and the latent neuron representation.

[**GitHub repository with all the computational details**]

The data generation and training processes are quite demanding in terms of time; thus, pre-trained data in all the cases discussed in this project can also be found in the repository.

References

[1] C. Wood *Powerful 'Machine Scientists' Distill the Laws of Physics From Raw Data.* Quanta Magazine, 2022.

[2] M. Schmidt and H. Lipson, *Distilling Free-Form Natural Laws from Experimental Data.* Science, **324**, 5923, 2009.

[3] R. Iten et al. *Discovering physical concepts with neural networks.* Phys. Rev. Letter, **124**, 010508, 2020.

[4] R. Tito D'Agnolo and A. Wulzer. *Learning New Physics from a Machine,* Phys. Rev., **D 99**, 015014, 2019.

[5] T. Wu and M. Tegmark. *Toward an AI Physicist for Unsupervised Learning*, Phys. Rev., **E 100**, 033311, 2019.

[6] S. H. Kellert, *In the Wake of Chaos: Unpredictable Order in Dynamical Systems.* University of Chicago Press, p. 32, 1993.

[7] R. B. Levien and S. M. Tan. *Double Pendulum: An experiment in chaos.* American Journal of Physics, **61** (11), 1038, 1993.

[8] B. Yeşilyurt. *Equations of Motion Formulation of a Pendulum Containing N-point Masses.* 2019.

[9] S. M. A. Eslami et al. *Neural scene representation and rendering.* Science, **360**, 1204, 2018.

[10] Y. Bengio et al. *Representation Learning: A Review and New Perspectives.* IEEE Transactions on Pattern Analysis and Machine Intelligence, **35**, 1798, 2013.

[11] G. E. Hinton and R. R. Salakhutdinov, *Reducing the Dimensionality of Data with Neural Networks*, Science **313**, 504, 2006.