

Department of Precision and Microsystems Engineering

Deep Symbolic Regression for Nonlinear Dynamical Systems

A. Thamban

Report no : 2023.041  
Coach : Farbod Alijani  
Professor : Alejandro Aragón  
Specialisation : Engineering Mechanics  
Type of report : Master Thesis  
Date : 28<sup>th</sup> June, 2023

# Deep Symbolic Regression for Nonlinear Dynamical Systems

by

Arun Thamban

Thesis report submitted in partial fulfillment of the requirements for the degree

*Master of Science, Mechanical Engineering*

at the Delft University of Technology,

to be defended publicly on

*Wednesday, June 28<sup>th</sup> 2023 at 9:30AM.*

Student number: 5445728

Project duration: August 22, 2022 – June 27, 2023

Thesis committee: Dr. A. M. Aragón, TU Delft, chair  
Dr. F. Alijani, TU Delft, supervisor  
Dr. D. M. J. Tax, TU Delft  
A. Keşkekler, TU Delft

# Preface

At the time of writing, the world is still adapting to the introduction of incredibly powerful Artificial Intelligence tools that blur the line between human and machine further than ever before. AI has revolutionised or replaced several fields of human endeavour. To enthusiasts such as myself, each development brings only anticipation for what is to come. Deep Learning-based techniques are also finding their way into Mechanical Engineering, and this thesis documents my efforts to facilitate that. I believe that the key difference between the Machine Learning-powered tools of yore and the current breed of virtual helpers is *explainability*. In other words, your chatbot of choice can typically back its responses up with credible reasoning, illuminating a hitherto opaque model. The reader (in the hope there are a few) will find explainability to be a key theme throughout my work as well.

I began this thesis armed with knowledge from a single introductory class in Dynamics and a couple of YouTube videos on Deep Learning. An interest in working on numerical models for origami structures brought me to the office of my supervisors, but I left having chosen a research direction that was somewhat divergent from my peers in the Hybrida group. I am convinced that there are significant potential benefits in the incorporation of AI-based tools in Mechanics and Dynamics research. There are, indeed, limitations to what is possible, but the first step in overcoming limitations is to find them. I hope that more researchers in this field deem it worthwhile to do so.

I would like to thank Alejandro Aragón and Farbod Alijani for their guidance and feedback throughout this project, and for setting targets that demanded more of me and my work. I am also grateful to Vincent Bos, Lucas Norder and Saransh Goel for valuable discussions on various aspects of this work. I had to step outside my comfort zone to complete this project, and I would not have been able to do it without the love and support of my family and friends. Thank you for helping me make my dreams come true.

*Arun Thamban  
Delft, June 2023*

# Abstract

From the motion of electrons in an atom to the orbits of celestial bodies in the cosmos, governing equations are essential to the characterisation of dynamical systems. They facilitate an understanding of the physics of a system, which enables the development of useful techniques such as predictive control. An increasingly popular method to obtain these equations is Symbolic Regression, where governing laws are discovered from observations of the system. In this work, we extend the Deep Symbolic Regression package to the identification of dynamical systems. Preliminary tests revealed the limits of the method as applied to dynamical systems, and new methods of incorporating domain knowledge to constrain the expression space are added. We test the extended package on 3 strongly nonlinear ODEs that exhibit different dynamics as their parameterisation varies. Finally, we demonstrate the working of this method in practice by discovering the governing equation of a pendulum, from a video of its oscillation. This method achieved a 100% equation recovery rate on our tests, and was able to consistently retrieve the correct equation from datasets representing a diverse range of dynamics.

# Contents

<b>Preface</b>	<b>i</b>
<b>Abstract</b>	<b>ii</b>
<b>I Introduction</b>	<b>1</b>
<b>II Paper: Deep Symbolic Regression for Nonlinear Dynamical Systems</b>	<b>4</b>
<b>III Reflection</b>	<b>20</b>
<b>IV Appendix</b>	<b>22</b>
A Learning in Neural Networks . . . . .	23
B Equation Represented by a Neural Network . . . . .	23
C Gradient Descent . . . . .	24
D LSTM Networks . . . . .	25
E Proof: Risk-Seeking Policy Gradient . . . . .	27
F Observations from Hyperparameter Tuning . . . . .	28
G Discovery of Modal Contributions in a Cantilever Beam using a Single-Point Measurement	30
<b>References</b>	<b>33</b>

## **Part I**

# **Introduction**

# Introduction

The characterisation of nonlinear dynamical systems often begin with their governing equations. Knowledge of the mathematics underlying an observed phenomenon is the first step to understanding the physics involved. This facilitates the development of useful engineering tools and techniques, such as predictive control [1] and uncertainty quantification [2]. In the past, scientists imposed mathematical models on empirical observations, informed by their own intuition and creativity. This approach yielded a number of foundational laws in physics, such as the Universal Law of Gravitation, Kepler's laws of planetary motion, and Schrödinger's equation. However, it is not always possible or practical to rely on intuition to discover a governing equation. Despite an abundance of data and ways to collect it, equation discovery is hindered by complex phenomena such as bifurcations, where the same equation exhibits qualitatively different dynamics with different parameter values. Furthermore, systems exhibiting chaotic dynamics are extremely sensitive to miniscule variations in their parameters, yielding dramatically different phase trajectories with parameter variations of the order  $10^{-5}$ . This necessitates the use of more advanced techniques. Black-box identification methods such as NARMAX [3] are a popular choice to model complex systems with unknown or poorly understood dynamics. However, they do not directly provide any insight into how the system dynamics is affected by variables of interest. Symbolic Regression (SR) methods prove useful in this regard.

By searching over the space of all mathematical functions that satisfy the imposed constraints, SR techniques can retrieve governing equations for dynamical systems, given data from them. However, the SR problem is NP-hard, i.e., no efficient algorithm that can solve an arbitrary SR problem exists. Hence, the SR task is typically tackled with heuristics that simplify the problem to a computationally tractable form. Sparse Identification of Nonlinear Dynamics (SINDy) [4] is one such heuristic. SINDy attempts to find an equation to fit a dataset by first building a set of mathematical terms from the state variables and their derivatives, followed by using sparse regression to identify the terms in that set which belong to the unknown equation. Several improvements to SINDy have been proposed since its introduction in 2016, such as robustness to noise [5], identification of rational nonlinearities [6], and adaptations to handle inaccurate derivative information [7]. However, a fundamental limitation of SINDy is that the discovered equation is strongly dependent on the chosen set of terms. Deciding which terms to use is a non-trivial task, since an increase in the order of the equation and/or the number of variables results in a combinatorial explosion of the number of possible entries. Furthermore, terms that result from the multiplication, division, or composition of other terms in the set cannot be discovered from their constituent components.

To circumvent the *a priori* selection of possible terms in the unknown equation, SR techniques based on Genetic Programming (GP) and Neural Networks (NNs) were developed. GP aims to discover an unknown equation by gradually evolving the best performing candidate equations through a series of operations inspired by biological processes, such as selection, mutation, and crossover. The terms within each equation are constructed from the ground up when using GP (as opposed to pre-defined terms in SINDy). Hence, they are far more flexible in the types of equations that can be discovered and are capable of discovering complex functional forms. Early breakthroughs in the use of GP to discover mathematical equations came in the early 2000s, where they were used to discover natural laws and coupled nonlinear systems [8, 9]. Since then, GP has been used successfully in the data-driven identification of Partial Differential Equations (PDEs) [10] and are the basis for commercial symbolic regression packages such as Eureqa. However, the flexibility afforded by GP comes with the associated drawback of poor scalability, since the search space is subject to the curse of dimensionality. Further, GP-based methods are known to exhibit high sensitivity to the applicable hyperparameters [11]. Some of the tunable hyperparameters in GP include the mutation rate, crossover rate, and the selection method.

As with GP, the use of NNs for SR is an active area of research. It is known that neural networks with just a single hidden layer are universal approximators, i.e., they can estimate any continuous function with an accuracy contingent on the number of neurons and the activation function [12]. Hence, NNs are *generalisable* in the sense that they can find a relationship between the input and output variables for any dataset, after training. The generalisation capabilities of NNs are typically exploited for the pre-processing or densification of data to support the functioning of another SR method. AI Feynman [13] is a physics-inspired SR method which uses an NN to learn the data distribution and discover relationships such as symmetry and separability of variables. This information is then used to simplify the symbolic regression task. Xu et al. presented a method called DL-PDE [14, 15]. This method leverages automatic differentiation in NNs to retrieve accurate derivative information from sparse and noisy data, which enabled the discovery of PDEs with time varying coefficients.

A caveat of the generalisation capabilities of NNs is that they are known to represent highly complex (often non-physical) equations. Their successful application to equation discovery is contingent on improving their *interpretability*, i.e., the output from the NN needs to be made understandable to the user. This may entail the use of another SR method in conjunction with the NN (as with DL-PDE), or modification of the NN to represent a more parsimonious expression. EQuation Learner (EQL) [16] is a SR method that employs a non-standard NN architecture. The activation functions in the hidden layers of the network are common functions that are often seen in physical equations, and sparsity is promoted in the weights of the network. After training, a concise, interpretable equation is obtained from the NN. This method was later extended by Sahoo et al. to facilitate the discovery of rational functions [17] and by Zhang et al. to accommodate time varying parameters [18]. Long et al. leveraged both the data pre-processing and function finding capabilities of NNs to create a hybrid numeric-symbolic network that can discover PDEs from sparse and noisy data [19]. While these methods shed light on the previously opaque internal working of NNs, the space of equations that can be searched by these NNs are restricted. Operations that are not defined on all real numbers (square root, logarithm, division) cannot be used as activation functions in the hidden layers, and these methods perform poorly with datasets that involve these operations.

When NNs are used to directly approximate the equation we are trying to find, the interpretability of the discovered equation comes at the cost of the generality of the approach. The successful application of NNs in the AI Feynman and DLGA-PDE methods is contingent upon the fact that the NNs were not required to be interpretable, and thus retained their generalisability while learning relations from the data. However, these methods require an additional procedure to find the symbolic expression that is represented by the NN. Deep Symbolic Regression (DSR) [11], developed by Petersen et al., is able to bypass this tradeoff entirely. In contrast to the other approaches, the NN of DSR outputs a probability distribution which is used to construct an equation that fits the data. No additional SR techniques are used and there is no requirement for the NN to be interpretable. An inbuilt system of user-defined constraints facilitates limiting the searched space of expressions to only contain those that may feasibly represent a physical system, thereby ameliorating the scalability issues associated with the curse of dimensionality. Through the use of a training scheme that maximises the best-case performance of the expressions generated by the network, DSR is able to outperform even commercial SR packages. Although the applicability of this method has been demonstrated on algebraic equations, it is yet to be extended to discover differential equations. This is the focus of the current work. Preliminary testing was conducted to understand the operational limits of the method, and new methods of constraining the expression space are introduced. The code extensions were then tested on three ODEs with strong nonlinearities- the van der Pol equation, the Selkov model and the Duffing equation. The dynamics represented by these equations changes qualitatively as their parameters are varied. It is demonstrated that DSR is robust to the type of dynamics underlying the dataset used for regression. Finally the method is validated with an empirical test, where the equation of motion of a pendulum is discovered from a video of its motion. In this test, equation recovery performance is evaluated with two trials; the first with moderate oscillations and the second with whirling motion of the pendulum bob. DSR is able to recover the correct expression in both cases, and achieved a 100% success rate in discovering the governing equations for all tested systems.



## **Part II**

# **Paper: Deep Symbolic Regression for Nonlinear Dynamical Systems**

# Deep Symbolic Regression in Nonlinear Dynamics

Arun Thamban, Alejandro Aragón and Farbod Alijani

Precision and Microsystems Engineering, Delft University of Technology,  
Mekelweg 2, Delft, 2628LX, The Netherlands.

\*Corresponding author(s). E-mail(s): [arunthamban98@gmail.com](mailto:arunthamban98@gmail.com);  
Contributing authors: [a.m.aragon@tudelft.nl](mailto:a.m.aragon@tudelft.nl); [f.alijani@tudelft.nl](mailto:f.alijani@tudelft.nl);

## Abstract

From the motion of electrons in an atom to the orbits of celestial bodies in the cosmos, governing equations are essential to the characterisation of dynamical systems. They facilitate an understanding of the physics of a system, which enables the development of useful techniques such as predictive control. An increasingly popular method to obtain these equations is Symbolic Regression, where governing laws are discovered from observations of the system. In this work, we extend the Deep Symbolic Regression package to the identification of dynamical systems. Preliminary tests were conducted to understand the limits of the method as applied to dynamical systems, and new methods of incorporating domain knowledge to constrain the expression space are added. We test the extended package on 3 parameterised nonlinear ODEs that exhibit different dynamics as the parameterisation varies. Finally, we demonstrate the working of this method in practice by discovering the governing equation of a pendulum, from a video of its oscillation. This method achieved a 100% equation recovery rate on our tests, and was able to consistently retrieve the correct equation from datasets representing a diverse range of dynamics.

**Keywords:** Nonlinear Dynamics, Symbolic Regression, Deep Learning

## 1 Introduction

The characterisation of nonlinear dynamical systems often begin with their governing equations. Knowledge of the mathematics underlying an observed phenomenon is the first step to understanding the physics involved. This facilitates the development of useful engineering tools and techniques, such as predictive control [1] and

uncertainty quantification [2]. In the past, scientists imposed mathematical models on empirical observations, informed by their own intuition and creativity. This approach yielded a number of foundational laws in physics, such as the Universal Law of Gravitation, Kepler’s laws of planetary motion, and Schrödinger’s equation. However, it is not always possible or practical to rely on intuition to discover a governing equation. Despite an abundance of data and ways to collect it, equation discovery is hindered by complex phenomena such as bifurcations, where the same equation exhibits qualitatively different dynamics with different parameter values. Furthermore, systems exhibiting chaotic dynamics are extremely sensitive to miniscule variations in their parameters, yielding dramatically different phase trajectories with parameter variations of the order  $10^{-5}$ . This necessitates the use of more advanced techniques. Black-box identification methods such as NARMAX [3] are a popular choice to model complex systems with unknown or poorly understood dynamics. However, they do not directly provide any insight into how the system dynamics is affected by variables of interest. Symbolic Regression (SR) methods prove useful in this regard.

By searching over the space of all mathematical functions that satisfy the imposed constraints, SR techniques can retrieve governing equations for dynamical systems, given data from them. However, the SR problem is NP-hard, i.e., no efficient algorithm that can solve an arbitrary SR problem exists. Hence, the SR task is typically tackled with heuristics that simplify the problem to a computationally tractable form. Sparse Identification of Nonlinear Dynamics (SINDy) [4] is one such heuristic. SINDy attempts to find an equation to fit a dataset by first building a set of mathematical terms from the state variables and their derivatives, followed by using sparse regression to identify the terms in that set which belong to the unknown equation. Several improvements to SINDy have been proposed since its introduction in 2016, such as robustness to noise [5], identification of rational nonlinearities [6], and adaptations to handle inaccurate derivative information [7]. However, a fundamental limitation of SINDy is that the discovered equation is strongly dependent on the chosen set of terms. Deciding which terms to use is a non-trivial task, since an increase in the order of the equation and/or the number of variables results in a combinatorial explosion of the number of possible entries. Furthermore, terms that result from the multiplication, division, or composition of other terms in the set cannot be discovered from their constituent components.

To circumvent the *a priori* selection of possible terms in the unknown equation, SR techniques based on Genetic Programming (GP) and Neural Networks (NNs) were developed. GP aims to discover an unknown equation by gradually evolving the best performing candidate equations through a series of operations inspired by biological processes, such as selection, mutation, and crossover. The terms within each equation are constructed from the ground up when using GP (as opposed to pre-defined terms in SINDy). Hence, they are far more flexible in the types of equations that can be discovered and are capable of discovering complex functional forms. Early breakthroughs in the use of GP to discover mathematical equations came in the early 2000s, where they were used to discover natural laws and coupled nonlinear systems [8, 9]. Since then, GP has been used successfully in the data-driven identification of Partial Differential Equations (PDEs) [10] and are the basis for commercial symbolic regression

packages such as Eureqa. However, the flexibility afforded by GP comes with the associated drawback of poor scalability, since the search space is subject to the curse of dimensionality. Further, GP-based methods are known to exhibit high sensitivity to the applicable hyperparameters [11]. Some of the tunable hyperparameters in GP include the mutation rate, crossover rate, and the selection method.

As with GP, the use of NNs for SR is an active area of research. It is known that neural networks with just a single hidden layer are universal approximators, i.e., they can estimate any continuous function with an accuracy contingent on the number of neurons and the activation function [12]. Hence, NNs are *generalisable* in the sense that they can find a relationship between the input and output variables for any dataset, after training. The generalisation capabilities of NNs are typically exploited for the pre-processing or densification of data to support the functioning of another SR method. AI Feynman [13] is an physics-inspired SR method which uses an NN to learn the data distribution and discover relationships such as symmetry and separability of variables. This information is then used to simplify the symbolic regression task. Xu et al. presented a method called DL-PDE [14, 15]. This method leverages automatic differentiation in NNs to retrieve accurate derivative information from sparse and noisy data, which enabled the discovery of PDEs with time varying coefficients.

A caveat of the generalisation capabilities of NNs is that they are known to represent highly complex (often non-physical) equations. Their successful application to equation discovery is contingent on improving their *interpretability*, i.e, the output from the NN needs to be made understandable to the user. This may entail the use of another SR method in conjunction with the NN (as with DL-PDE), or modification of the NN to represent a more parsimonious expression. EQuation Learner (EQL) [16] is a SR method that employs a non-standard NN architecture. The activation functions in the hidden layers of the network are common functions that are often seen in physical equations, and sparsity is promoted in the weights of the network. After training, a concise, interpretable equation is obtained from the NN. This method was later extended by Sahoo et al. to facilitate the discovery of rational functions [17] and by Zhang et al. to accommodate time varying parameters [18]. Long et al. leveraged both the data pre-processing and function finding capabilities of NNs to create a hybrid numeric-symbolic network that can discover PDEs from sparse and noisy data [19]. While these methods shed light on the previously opaque internal working of NNs, the space of equations that can be searched by these NNs are restricted. Operations that are not defined on all real numbers (square root, logarithm, division) cannot be used as activation functions in the hidden layers, and these methods perform poorly with datasets that involve these operations.

When NNs are used to directly approximate the equation we are trying to find, the interpretability of the discovered equation comes at the cost of the generality of the approach. The successful application of NNs in the AI Feynman and DLGA-PDE methods is contingent upon the fact that the NNs were not required to be interpretable, and thus retained their generalisability while learning relations from the data. However, these methods require an additional procedure to find the symbolic expression that is represented by the NN. Deep Symbolic Regression (DSR) [11], developed by Petersen et al., is able to bypass this tradeoff entirely. In contrast to the other

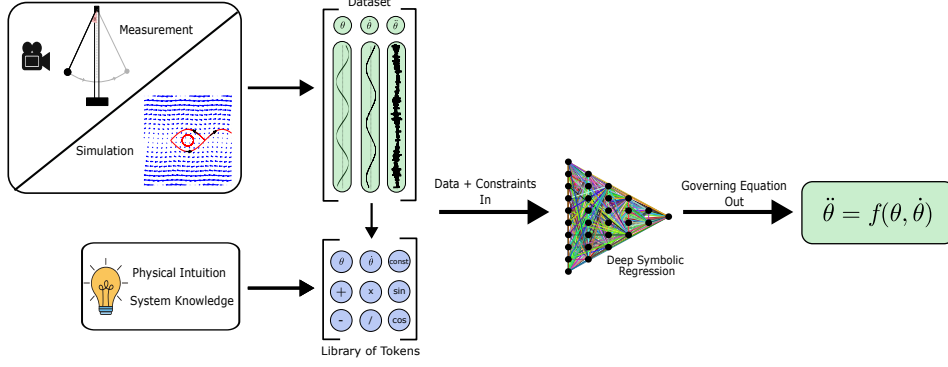
approaches, the NN of DSR outputs a probability distribution which is used to construct an equation that fits the data. No additional SR techniques are used and there is no requirement for the NN to be interpretable. An inbuilt system of user-defined constraints facilitates limiting the searched space of expressions to only contain those that may feasibly represent a physical system, thereby ameliorating the scalability issues associated with the curse of dimensionality. Through the use of a training scheme that maximises the best-case performance of the expressions generated by the network, DSR is able to outperform even commercial SR packages. Although the applicability of this method has been demonstrated on algebraic equations, it is yet to be extended to discover differential equations. This is the focus of the current work. Preliminary testing was conducted to understand the operational limits of the method, and new methods of constraining the expression space are introduced. The code extensions were then tested on three ODEs with strong nonlinearities- the van der Pol equation, the Selkov model and the Duffing equation. The dynamics represented by these equations changes qualitatively as their parameters are varied. It is demonstrated that DSR is robust to the type of dynamics underlying the dataset used for regression. Finally the method is validated with an empirical test, where the equation of motion of a pendulum is discovered from a video of its motion. In this test, equation recovery performance is evaluated with two trials; the first with moderate oscillations and the second with whirling motion of the pendulum bob. DSR is able to recover the correct expression in both cases, and achieved a 100% success rate in discovering the governing equations for all tested systems.

## 2 Methodology

Deep symbolic regression [11] is an SR method that uses a recurrent neural network to generate a probability distribution over mathematical expressions, from which candidate solutions are sampled. Through the use of a modified reinforcement learning technique, described below, the NN adjusts the probability distribution such that better fitting solutions have a higher probability of being sampled. By using the NN to emit a probability distribution over expressions (as opposed to an expression itself), the task of interpreting the NN is rendered obsolete. This is because the expressions sampled from the distribution are obtained as expression trees, which are already interpretable.

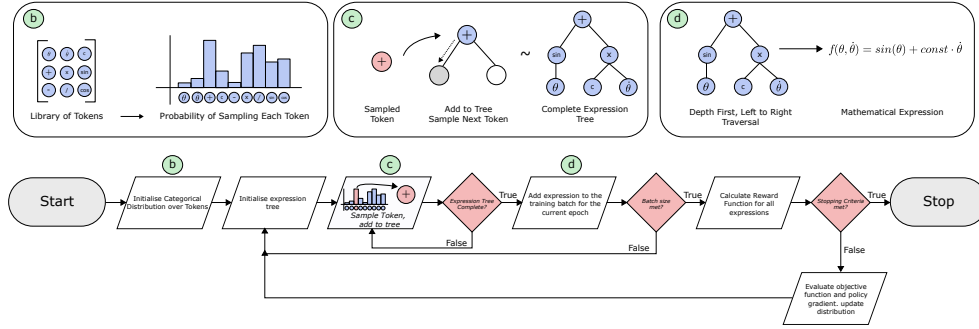
### 2.1 Equation Sampling and Constraint Handling

The expressions are generated sequentially and in an autoregressive manner, i.e, the probability distribution for the current selection depends on the operators and variables that have been selected previously. The expression is generated from a library of tokens, as shown in figure 1. The tokens may be unary operators (accepting one argument), binary operators, or variables. Constants may also be included via a token. If constants are used, then the algorithm also includes a constant optimisation step to discover the values of the constants that provide the best fit. Any optimisation algorithm may be used for this purpose (by default, DSR uses the L-BFGS optimisation algorithm). In order to prevent the generation of low-quality expressions, such as



**Fig. 1** Workflow: Discovering the governing laws of dynamical systems with DSR

ones which are too short, include unphysical terms such as  $\sin(\cos(x))$ , or inversions such as  $\log(e^x)$ , constraints are applied during the expression generation stage. Given the sequence of tokens that have already been sampled, the probability distribution for the following selection is adjusted such that tokens which yield expressions that are unlikely to fit the dataset have zero probability associated with them. Figure 2 depicts the equation sampling process and how the weights of the Neural Network are evolved with training.



**Fig. 2** Equation sampling and training of the NN in DSR. Expression trees are dynamically generated from the sampled tokens. The equation structure is dependent on the arity of the sampled token. The tree is then interpreted as an expression by reading the tree depth-first and from left to right.

## 2.2 Reward Function and Risk-Seeking Objective

The reward function used to evaluate the expressions generated by the NN in DSR is not differentiable with respect to the weights of the NN. Hence, the reinforcement learning problem is formulated in terms of maximising the *expectation* of a reward on expressions generated by the NN, which is a function of the weights. The gradient of the

expectation may then be used to update the weights of the network. Mathematically, the expectation and its gradient are given by [11]

$$J_{std}(\mathcal{W}) = \mathbb{E}_{\tau \sim p(\tau|\mathcal{W})}[R(\tau)] \quad (1)$$

$$\nabla_{\mathcal{W}} J_{std}(\mathcal{W}) = \mathbb{E}_{\tau \sim p(\tau|\mathcal{W})}[R(\tau) \nabla_{\mathcal{W}} \log(p(\tau|\mathcal{W}))] \quad (2)$$

Here,  $\mathcal{W}$  are the weights of the NN,  $p(\tau|\mathcal{W})$  is the probability of sampling an expression  $\tau$  given the weights  $\mathcal{W}$  of the NN. The reward function,  $R$  is defined for a candidate expression  $f$  and a dataset  $(X, y)$  containing  $N$  data points as

$$R(\tau) = \frac{1}{1 + \delta} \quad (3)$$

$$\delta = \frac{1}{\sigma_y} \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - f(X_i))^2} \quad (4)$$

Here,  $\sigma_y$  is the standard deviation of the output values,  $y$ . For a batch of  $N$  sampled expressions, an estimate of the gradient of  $J_{std}(\mathcal{W})$  is obtained as

$$\nabla_{\mathcal{W}} J_{std}(\mathcal{W}) \approx \frac{1}{N} \sum_{i=1}^N [R(\tau^i) \nabla_{\mathcal{W}} \log(p(\tau^i|\mathcal{W}))] \quad (5)$$

The above formulation maximises the expectation of a reward *on average* across the sampled expressions. While improving the average performance is a sensible choice for applications such as autonomous driving, the SR task may be considered complete if a single expression that exactly fits the dataset is sampled. Hence, it is desirable to train the NN in a manner that maximises the accuracy of the best-performing expressions, at the expense of average performance. An alternative objective function was proposed to this end [11]

$$J_{risk}(\mathcal{W}, \epsilon) = \mathbb{E}_{\tau \sim p(\tau|\mathcal{W})}[R(\tau) | R(\tau) \geq R_{\epsilon}(\mathcal{W})] \quad (6)$$

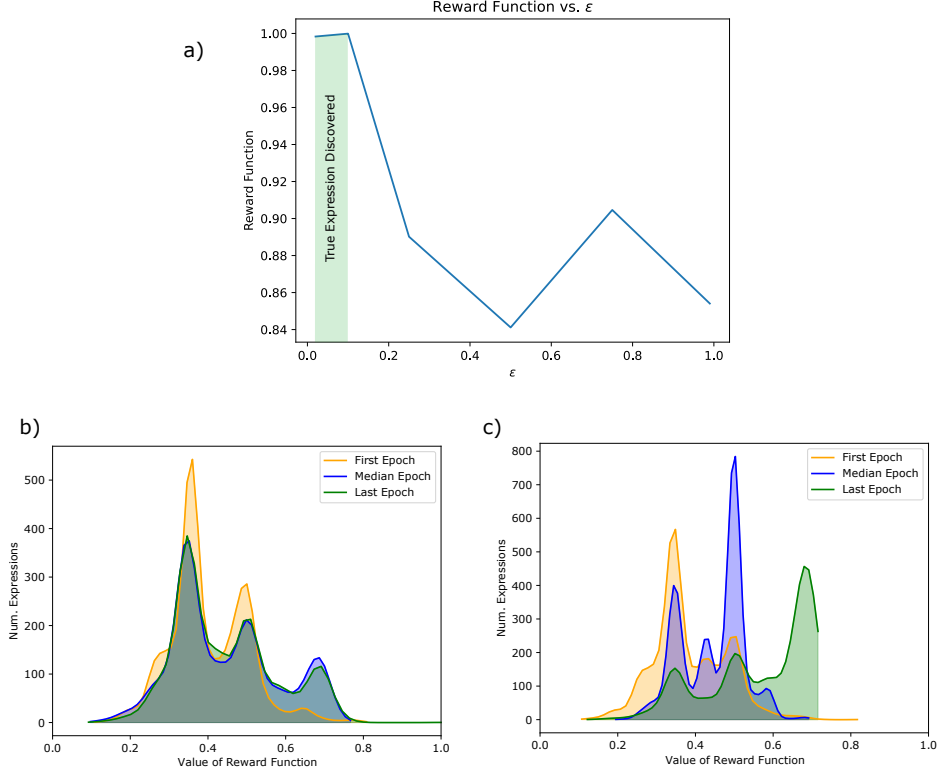
Here,  $R_{\epsilon}(\mathcal{W})$  is the  $(1 - \epsilon)$  quantile of the distribution of rewards. This objective only considers the reward of the top  $\epsilon$  fraction of sampled expressions, which increases best case performance at the expense of average performance. It was shown that the gradient of this objective is given by [20]

$$\nabla_{\mathcal{W}} J_{risk}(\mathcal{W}, \epsilon) = \mathbb{E}_{\tau \sim p(\tau|\mathcal{W})}[(R(\tau) - R_{\epsilon}(\mathcal{W})) \cdot \nabla_{\mathcal{W}} \log(p(\tau|\mathcal{W})) | R(\tau) \geq R_{\epsilon}(\mathcal{W})] \quad (7)$$

From a batch of  $N$  sampled expressions, the gradient can be estimated by

$$\nabla_{\mathcal{W}} J_{risk}(\mathcal{W}, \epsilon) \approx \frac{1}{\epsilon N} \sum_{i=1}^N [(R(\tau^i) - \tilde{R}_\epsilon(\mathcal{W})) \cdot (\mathbf{1}_{R(\tau^i) \geq \tilde{R}_\epsilon(\mathcal{W})}) \nabla_{\mathcal{W}} \log(p(\tau^i | \mathcal{W}))] \quad (8)$$

Where  $\tilde{R}_\epsilon(\mathcal{W})$  is the empirical  $(1-\epsilon)$  quantile of rewards, and  $\mathbf{1}_x$  returns 1 if  $x$  is true and 0 otherwise. Figure 3 shows the effect of the risk-seeking objective in training.



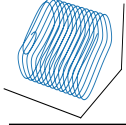
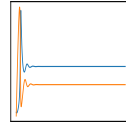
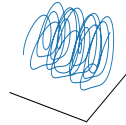
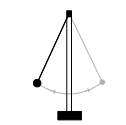
**Fig. 3** a.) Value of Reward Function at the end of training vs. the parameter  $\epsilon$ , as tested with the Duffing equation. Higher values of  $\epsilon$  lead to a higher fraction of the sampled batch being considered for training. This is associated with a drop in equation recovery performance. b) Distribution of Rewards across batches of sampled expressions for  $\epsilon = 0.02$  and c)  $\epsilon = 0.99$ . The risk-seeking objective creates longer tails in the distribution, which is beneficial in this context. The distribution of rewards in the first training epoch is the same in both cases.

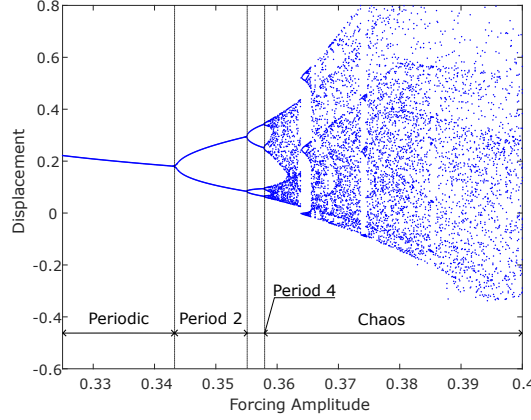


### 3 Results

#### 3.1 Testing with Computer-Generated Data

The efficacy of DSR in discovering ODEs with strong nonlinearities was first tested with simulated data. As illustrated in Figure 5, the Selkov equation exhibits a Hopf bifurcation depending on the values of parameters  $a$  and  $b$ , transitioning from a stable equilibrium to a limit cycle. The Duffing equation exhibits a period-doubling cascade as the value of the forcing amplitude increases. Chaotic dynamics are observed with higher forcing amplitudes, as seen in Figure 4. For both sets of equations, DSR is able to recover the true expression irrespective of the dynamics contained in the dataset. In some cases, a negligibly small extraneous term was also present in the discovered equation. The presence of the extraneous term did not affect the dynamics of the system, when the discovered equations were simulated. The coefficients of the terms in the equations were estimated with an error of  $10^{-5}$ . This is the cause of the divergent phase-space trajectory when the dynamics are chaotic, as observed in Figure 6. Chaotic dynamics are highly sensitive to miniscule variations in parameter values, and even marginal errors in the discovered parameter values can result in the predictive power of the equation being restricted to a certain interval of time.

Equation Name	True Form / Discovered Form	Epochs to Convergence	RMSE on Dataset
 <div>van der Pol</div>	$\ddot{x} = (1 - x^2)\dot{x} - x$ $\ddot{x} = (1 - x^2)\dot{x} - x$	2	$8.795 \cdot 10^{-7}$
 <div>Selkov</div>	$\dot{y} = 0.6 - 0.2y - x^2y$ $\dot{y} = 0.6 - 0.2y - x^2y$	2	$1.364 \cdot 10^{-4}$
 <div>Duffing</div>	$\ddot{x} = 0.39\cos(t) - 0.5\dot{x} + x - x^3$ $\ddot{x} = 0.3899\cos(t) - 0.5\dot{x} + x - x^3$	92	$9.733 \cdot 10^{-4}$
 <div>Pendulum</div>	$\ddot{\theta} = -0.0921\dot{\theta} - 80.3246\sin(\theta)$ $\ddot{\theta} = -0.1093\dot{\theta} - 80.6807\sin(\theta)$	10	$1.074 \cdot 10^{-1}$

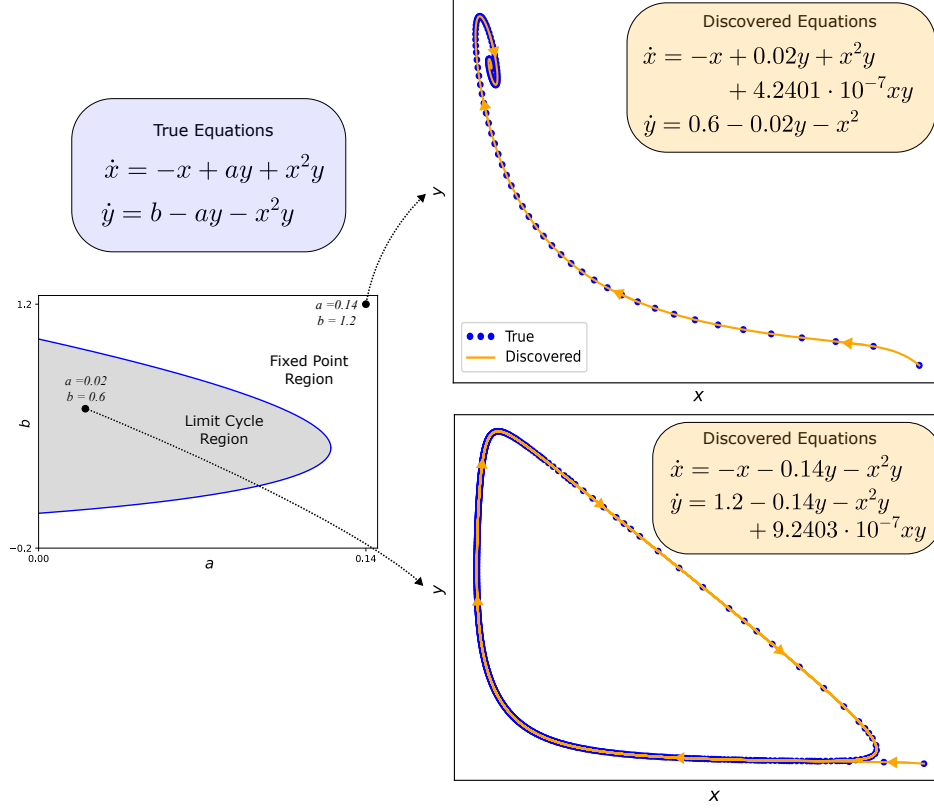


**Fig. 4** Bifurcation Diagram of the Duffing equation. A period-doubling cascade is observed as the value of the forcing amplitude increases, with bifurcations at  $f = 0.343$ ,  $f = 0.355$ , and  $f = 0.358$ . The dynamics exhibited by the equation in each of these intervals is distinct, as seen in Figure 6

### 3.2 Validation with Empirical Data

DSR was used to discover the governing equation for a pendulum from a video of its motion. Given the relative simplicity of the pendulum equation as compared to the test equations used in section 3.1, it is clear that the discovery of this equation does not provide any notable additional insight into the capabilities of this method. However, it does illustrate a potential use-case for this method in practice. The ability to recover a governing equation from empirical data would be a precious tool to understand the behaviour of the system in an intuitive manner. Additionally, the video of the pendulum’s motion is shot using a low-precision, low-cost setup, which introduces a large amount of noise in the dataset. Thus, this test also demonstrates how DSR (combined with denoising techniques) is able to cope with poor-quality measurement data.

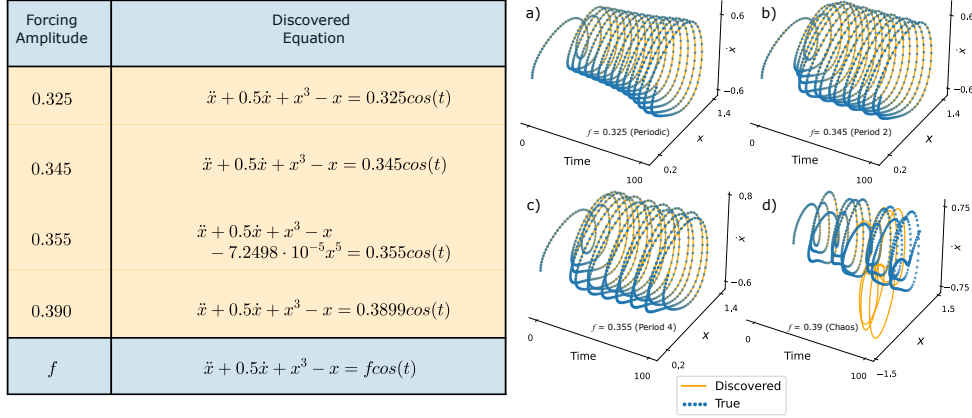
The video was shot with a mobile phone camera. Aside from a camera tripod, no special equipment was used. Positional data was extracted from the video using Tracker video analysis software. To eliminate some of the high-frequency noise introduced during measurement, we considered the moving average of angular position, amalgamating a number of frames into a single data point. The angular velocity and acceleration were computed numerically from the averaged position data. From Figure 7, we see that DSR is able to successfully identify the structure of the differential equation (harmonic function of the angular position, damping term). The damping coefficient is estimated with a maximum error of 21% and the natural frequency is estimated with a maximum error of 0.44%.



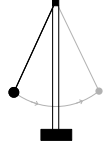

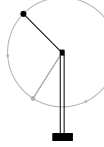

**Fig. 5** DSR as applied to the Selkov model. This system exhibits different dynamics based on the values of  $a$  and  $b$ , with the dynamics either converging to a stable limit cycle or a fixed point. In both cases, DSR is able to recover the governing equations, estimating the coefficients up to the fifth decimal place.

## 4 Discussion

Sections 3.1 and 3.2 show that Deep Symbolic regression is a highly capable method to understand the physics driving a system, using observations of it. As outlined in section 2.2, the reward and objective function seek to generate expressions that minimise the NRMSE on the output variable of the dataset (i.e., data of the highest order derivative). The sampled equations are thus treated as algebraic expressions, and certain subtleties of differential equations (such as how derivatives evolve with the value of the base variable) are ignored. While this does not present an issue in the majority of considered test cases, disparate phase trajectories are observed when the dynamics are chaotic (Figure 6). Figure 8 indicates that the value of the forcing amplitude would need to be resolved up to the eighth decimal place for the phase trajectory to match the true dynamics over the complete dataset. It is clear that although the algorithm achieves an excellent fit on the dataset when all derivatives are independent



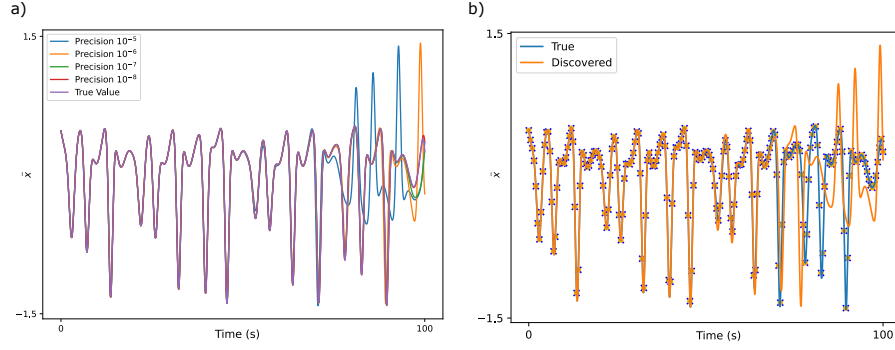
**Fig. 6** Results from testing on the Duffing equation with harmonic forcing. The dynamics arising from this equation evolve as the forcing amplitude changes, as described in Figure 4. For periodic and subharmonic dynamics (subfigures a, b, c), the true dynamics are captured exactly by simulating the equation discovered by DSR. For chaotic motion (subfigure d)), a significant deviation is observed for time  $> 50$ s. This is due to the error (of the order  $1e-5$ ) in estimating the coefficients in the equation. Small differences in parameter values causing large differences in the discovered phase portraits is a typical characteristic of chaotic dynamics.

Type of Motion	 Moderate Oscillation 	 Whirling Motion 
Discovered Equation	$\ddot{\theta} + 0.1093\dot{\theta} + 80.6807\sin(\theta) = 0$	$\ddot{\theta} + 0.0727\dot{\theta} + 80.6955\sin(\theta) = 0$
True Equation	$\ddot{\theta} + 0.0921\dot{\theta} + 80.326\sin(\theta) = 0$	

**Fig. 7** Summary of empirical testing with a simple pendulum. Two different initial conditions were considered; moderate oscillations ( $< 0.35$  rad) and whirling motion, with the bob tracing a complete circle about the fixed point. The data collection workflow introduces significant amounts of noise to the dataset. Although the measurement noise is ameliorated somewhat by the frame-averaging process, numerical estimation of the gradients also increases the noise level.

variables, the result from simulating the expression using a numerical solver differs significantly. This can be attributed to the error in estimating the coefficients of the terms in the equation. Small differences in parameter values causing large differences in the discovered phase portraits is a typical characteristic of chaotic dynamics.

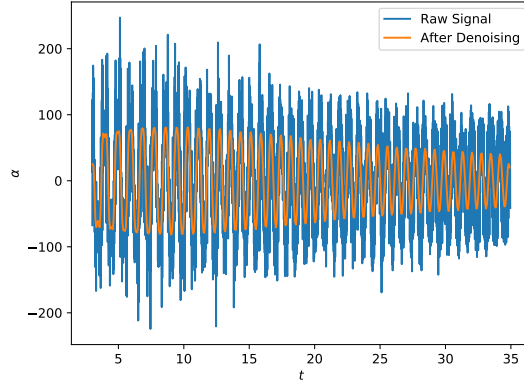
The tests also revealed that DSR is capable of identifying the true equation underlying a dataset even in the presence of low noise. However, higher noise levels causes a significant drop-off in the value of the reward function. The NN attempts to achieve



**Fig. 8** a.) Variation in the phase trajectories with the precision of the resolved forcing amplitude with chaotic dynamics. The true forcing amplitude is  $f = 0.39$ . b.) Acceleration vs. Time for  $f = 0.39$  in the Duffing equation. The marked points represent the acceleration values when the both expressions (true and discovered) are treated as algebraic expressions, and the lines represent the acceleration values when the expressions are simulated.

an exact fit against noisy data, promoting overfitting. In the context of dynamical systems, the problem is further exacerbated by the fact that high variable powers can make an equation more difficult to simulate, and less likely to represent the true dynamics of a physical system. This issue is particularly evident when working with empirical data. For the whirling motion of the pendulum, the inaccuracy in collecting the angular position is of the order of  $\pm 0.9\text{rad}$  (framewise). This error is further amplified when evaluating the derivatives, which necessitated the use of denoising techniques when preparing the dataset for DSR. It was known that the noise was of a higher frequency than the signal, and frame-averaging techniques were thus useful in this regard. Figure 9 shows the effect of the frame averaging process, which resulted in a noise level that was sufficiently low for DSR to discover the true equation. As elaborated upon above, overfitted expressions are more likely to do better than the true expression when no consideration is made for the dynamics represented by the discovered expressions.

Although this work is focused on the application of deep learning to the identification of dynamical systems, the DSR framework could be used in other workflows as well. The RNN utilised by DSR acts as a sequence generator; given a set of basic building blocks, the method creates sequences comprised of one or more of these blocks and repetition of the building blocks is generally permitted. The task of equation discovery is internally formulated as a discrete sequence optimisation problem, which is solved as the RNN is trained on the provided dataset. For the method to be used in a task other than symbolic regression, a few modifications must first be made; the task must be formulated as a discrete sequence optimisation problem, the requisite building blocks (tokens) and their properties should be added to the framework, and a method to interpret the generated sequences would be required. It is also not necessary that the generated sequence be interpreted as a unary-binary tree, just that the scheme of interpretation can be consistently applied to any generated sequence.



**Fig. 9** Angular Acceleration vs. Time, before and after frame averaging. For the whirling motion of the pendulum (shown here), 16 frames were averaged together to obtain a single data point, while two data frames were averaged together in the case of moderate oscillations.

## 5 Conclusion

In this work, the possibility of synthesising governing equations of dynamical systems from data was investigated. Deep Symbolic Regression was extended to the discovery of nonlinear dynamical ODEs and validated by testing equation recovery performance on computer generated-datasets for three strongly nonlinear ODEs and empirical data extracted a video of an oscillating simple pendulum. In all cases, DSR was able to recover the true equation underlying the dataset. The performance of the method was also unaffected in cases where varying the parameters of the same equation resulted in different dynamics. Coupled with suitable preprocessing techniques, DSR was also found to be robust against the absence of derivative measurements. Since DSR has seen limited application in the discovery of governing laws from empirical data, this work serves as a proof-of-concept for the method. Symbolic regression allows for the creation of interpretable, intuitive models for dynamical systems. DSR is especially notable since it incorporates a suite of constraints that can be used to incorporate domain knowledge into the equation discovery process in a straightforward manner. Further, the library of tokens employed for equation discovery ameliorates the scalability issues associated with other SR methods. Users may broadly specify which operators and variables are permitted in the generated expressions, but are not required to specify the terms themselves.

A significant outstanding limitation of the method is the absence of any consideration for the dynamics represented by the sampled equation. One possible approach to this would be to incorporate the dynamics of sampled equations into the corresponding value of the reward function. By treating a candidate expression as a differential equation (as opposed to an algebraic one), the complete state of the dynamical system represented by the expression may be obtained. This would allow for the evaluation of the fit over all state variables, and not just the highest order derivative. Such a training scheme may improve robustness to noise, since overfitted or unphysical expressions

are less likely to produce phase trajectories that are comparable to the true dynamics. Neural Networks are often treated as black boxes; more value is associated with the result as compared to the underlying functionality. Tools such as DSR demonstrate that there is much to be gained from a principled approach to leveraging the capabilities of neural networks in fields where their application is underexplored. A thorough understanding of this framework could enable a plethora of potential applications, from the quantification of nanoscale forces in Atomic Force Microscopy to the inverse design of origami structures. Much like imposing mathematical models on empirical observations, the application of deep learning-based tools is likely to be limited first by the researcher’s own creativity.

## References

- [1] Joseph Lorenzetti et al. “Reduced order model predictive control for setpoint tracking”. In: *2019 18th European Control Conference (ECC)*. IEEE. 2019, pp. 299–306.
- [2] Themistoklis P Sapsis and Andrew J Majda. “Statistically accurate low-order models for uncertainty quantification in turbulent dynamical systems”. In: *Proceedings of the National Academy of Sciences* 110.34 (2013), pp. 13705–13710.
- [3] Sheng Chen and Steve A Billings. “Representations of non-linear systems: the NARMAX model”. In: *International journal of control* 49.3 (1989), pp. 1013–1032.
- [4] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. “Discovering governing equations from data by sparse identification of nonlinear dynamical systems”. In: *Proceedings of the national academy of sciences* 113.15 (2016), pp. 3932–3937.
- [5] Urban Fasel et al. “Ensemble-SINDy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control”. In: *Proceedings of the Royal Society A* 478.2260 (2022), p. 20210904.
- [6] Kadierdan Kaheman, Steven L Brunton, and J Nathan Kutz. “Automatic differentiation to simultaneously identify nonlinear dynamics and extract noise probability distributions from data”. In: *Machine Learning: Science and Technology* 3.1 (2022), p. 015031.
- [7] Daniel A Messenger and David M Bortz. “Weak SINDy for partial differential equations”. In: *Journal of Computational Physics* 443 (2021), p. 110525.
- [8] Josh Bongard and Hod Lipson. “Automated reverse engineering of nonlinear dynamical systems”. In: *Proceedings of the National Academy of Sciences* 104.24 (2007), pp. 9943–9948.
- [9] Michael Schmidt and Hod Lipson. “Distilling free-form natural laws from experimental data”. In: *science* 324.5923 (2009), pp. 81–85.
- [10] Yuntian Chen et al. “Symbolic genetic algorithm for discovering open-form partial differential equations (SGA-PDE)”. In: *Physical Review Research* 4.2 (2022), p. 023174.

- [11] Brenden K Petersen et al. “Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients”. In: *arXiv preprint arXiv:1912.04871* (2019).
- [12] Yulong Lu and Jianfeng Lu. “A universal approximation theorem of deep neural networks for expressing probability distributions”. In: *Advances in neural information processing systems* 33 (2020), pp. 3094–3105.
- [13] Silviu-Marian Udrescu and Max Tegmark. “AI Feynman: A physics-inspired method for symbolic regression”. In: *Science Advances* 6.16 (2020), eaay2631.
- [14] Hao Xu, Haibin Chang, and Dongxiao Zhang. “Dl-pde: Deep-learning based data-driven discovery of partial differential equations from discrete and noisy data”. In: *arXiv preprint arXiv:1908.04463* (2019).
- [15] Hao Xu, Dongxiao Zhang, and Junsheng Zeng. “Deep-learning of parametric partial differential equations from sparse and noisy data”. In: *Physics of Fluids* 33.3 (2021), p. 037132.
- [16] Georg Martius and Christoph H Lampert. “Extrapolation and learning equations”. In: *arXiv preprint arXiv:1610.02995* (2016).
- [17] Subham Sahoo, Christoph Lampert, and Georg Martius. “Learning equations for extrapolation and control”. In: *International Conference on Machine Learning*. PMLR, 2018, pp. 4442–4450.
- [18] Michael Zhang et al. “Deep Learning and Symbolic Regression for Discovering Parametric Equations”. In: *arXiv preprint arXiv:2207.00529* (2022).
- [19] Zichao Long, Yiping Lu, and Bin Dong. “PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network”. In: *Journal of Computational Physics* 399 (2019), p. 108925.
- [20] Aviv Tamar, Yonatan Glassner, and Shie Mannor. “Optimizing the CVaR via sampling”. In: *Twenty-Ninth AAAI Conference on Artificial Intelligence*. 2015.



# **Part III**

## **Reflection**

# Reflection

At the start of this project, I sought to find a way to leverage the capabilities of neural networks towards an application in nonlinear dynamics. Together with Dr. Aragón and Dr. Alijani, we narrowed down the focus of the project to extracting governing equations from data, from where Symbolic Regression was a natural choice. I began this project with a limited understanding of Neural Networks and their working, and had minimal experience with tensorflow and parallel processing. Furthermore, I had never taken up a year-long, self-driven academic project such as this one before. During the literature review phase, I quickly realised that meticulous planning and management would be key to my success in this project. At this stage, I devoted a considerable amount of time to maintaining spreadsheets and logs to track my progress. I believe the effort that was invested in planning is one of the main reasons this project was completed on schedule. I accomplished all of the targets that were laid out at the end of my literature review. The DSR package was enhanced with new capabilities and constraints, which facilitated the discovery of the Duffing equation (among others) at a range of forcing amplitudes. The real-world test was also successful. Additionally, an attempt was made to incorporate the dynamics of sampled expressions into the training of the neural network.

Admittedly, making good progress and receiving mostly positive feedback about my work had an impact on the way I was handling the project. During the testing phase, I neglected my planning and instead focused on getting as much done as quickly as possible. This resulted in the opposite of my intention, and I had to redo several tests because I did not pay close attention to tabulating results and logging the outputs from my tests. This resulted in a decline in progress, which drove me to try and work even faster. By the time I became cognizant of the vicious cycle I was in, I had already spent about four weeks being minimally productive. Thankfully, this did not have a significant impact on the eventual outcome, since I was ahead of schedule by this point. I realise now that I gave my best efforts when I felt I was doing poorly, and tended to slack off when things were going well. Self awareness and reflection at intermediate stages would be key in ensuring this does not happen in future projects.

Over the course of this project, I was also made more aware of my personal strengths and weaknesses. I found that I was willing to go the extra mile to obtain better results, I was capable of learning new concepts and applying them in a short span of time, and I could multitask efficiently. I also found a number of stumbling blocks in my thinking and mentality that need to be worked on. I tend to jump to conclusions without being systematic, if the conclusion suits me. I can also be overly confident in my work, and resistant to feedback and advice at times. I am grateful to have had these aspects of my personality revealed to me, positive and negative. In the future, I intend to leverage the positives in situations where doing so would result in a better outcome, and I will manage the negatives, so that they do not adversely impact my work or my interactions with my peers and mentors.

# **Part IV**

## **Appendix**

# Appendix

## A. Learning in Neural Networks

When an NN is first initialised, the weights of the edges are arbitrary. Before they can be used as a mapping, the values of the weights which would yield the expected relation must be learned. Learning in NNs can be broadly classified in three categories [20]; unsupervised learning, supervised learning, and reinforcement learning.

### Unsupervised Learning

In applications such as recommender systems and similarity detection, we are often not required to make specific predictions; rather, we would like to discover patterns from the input data to group examples into different classes. Unsupervised learning techniques allow NNs to discover subgroups and similarities in the data without the use of any pre-defined output labels or groups. An NN trained using an unsupervised learning method such as Principal Component Analysis or K-Means Clustering should be able to disambiguate between different input examples on the basis of similarities learned during the training process. Although unsupervised learning has found some application in discovering solutions for elliptic PDEs [21], the SR task involves relating inputs to known outputs and making predictions. Hence, these techniques are not widely used in NN-based SR.

### Supervised Learning

Supervised learning involves training an NN to match the output of the training dataset when the input from the dataset is provided to it. Supervised learning tasks can be broadly classified into *classification* problems, where data must be assigned to specific categories, and *regression* problems, where the relationship between the input and the output is learned. Once an NN is trained using a supervised learning method, it is expected to be able to predict the output for previously unseen input data. Supervised learning finds widespread application in NN-based SR [16, 17, 18, 19], owing to the excellent alignment of supervised learning with the goals of SR.

## B. Equation Represented by a Neural Network

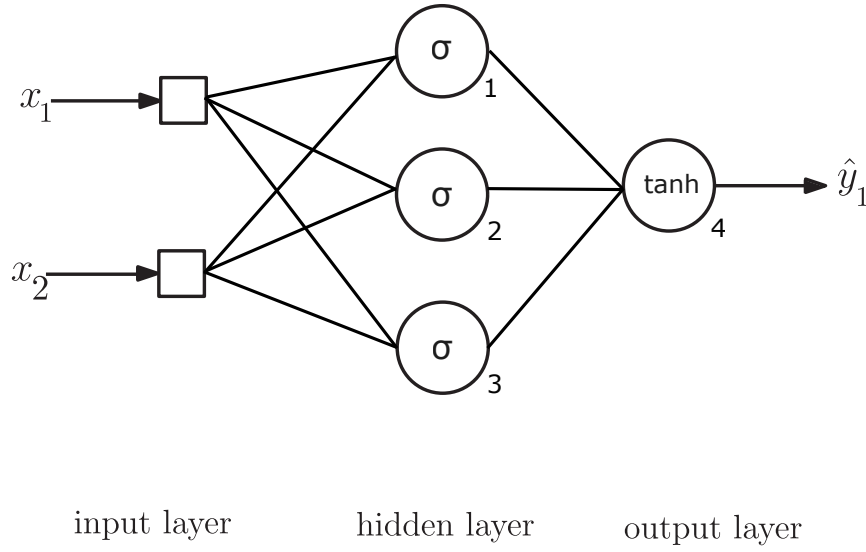
Neural networks transform the input given to them through a series of successive operations. In general, data passing through an edge of the network is multiplied with a weight, a learned parameter that is unique to each edge. At a node, the weighted input from all the edges connecting to that node are summed, and transformed with an activation function. The purpose of the activation function is to control the output from the node (and hence the input to other nodes). Often an activation function will also introduce a nonlinear transformation, improving the generalisation capabilities of the network.

Figure 1 shows a 3-layer neural network with 2 input nodes, 3 hidden nodes, and 1 output node. The activation function is chosen to be the sigmoid function in the hidden nodes and the hyperbolic tangent function for the output node. The weight associated with the edge between any node  $i$  and a node  $j$  is represented by  $w_{ij}^j$ . Consider the topmost node of the hidden layer (node 1). The output of this node can be expressed as

$$h(x_1, x_2) = \sigma(w_1^1 x_1 + w_2^1 x_2) \quad (1)$$

The equations for the other nodes may be obtained in a similar manner. The output,  $\hat{y}_1$ , can be expressed as

$$\hat{y}_1 = \tanh(w_1^4 \sigma(w_1^1 x_1 + w_2^1 x_2) + w_2^4 \sigma(w_1^2 x_1 + w_2^2 x_2) + w_3^4 \sigma(w_1^3 x_1 + w_2^3 x_2)) \quad (2)$$



**Figure 1:** 3-layer NN with sigmoid activation in the hidden layer and tanh activation in the output layer. Adapted from [22].

It can be seen from equation 2 that even for a small network, the equation is fairly complex. This complexity is the foundation for the generalisability of these networks, since a wide variety of functions could be approximated using the appropriate weights. As discussed in Chapter I, however, this comes at the cost of interpretability.

### Universal Approximation Theorem

The Universal Approximation Theorem states that feedforward neural networks with just one hidden layer can approximate any continuous mapping from one finite-dimensional space to another, under some mild assumptions [23]. We observe from figure 1 that the computation within the node is linear in the weights and the input to the nodes. However, successive linear transformations are generally not sufficient to approximate arbitrary mappings. Hence, nonlinearity is introduced in the nodal transformations through the use of an activation function. The argument of the activation function is the aforementioned linear transformation of the nodal inputs, and the value of the function is the nodal output. Some commonly used activation functions are the Rectified Linear Unit (ReLU), sigmoid, and tanh functions. It must be noted, however, that the theorem does not elaborate on the neural architecture required to approximate any continuous function with the desired level of accuracy. It simply states that the network is theoretically realisable.

## C. Gradient Descent

Irrespective of which type of learning is used, the training of an NN involves updation of the weights of the network with respect to a pre-defined metric, encoded in the objective function. In unsupervised learning, this metric may be the Euclidean distance of a training example from the sub-group that it belongs to [24]. Supervised learning and reinforcement learning typically employ an error measure such as the  $L_2$  norm or the root mean square error [11, 16]. A common optimization routine used in training of NNs is Gradient Descent (GD). For the neural network in figure 1 with output  $\hat{y}_1$  as given in equation 2 and using the mean-squared error as the loss function, we have

$$L(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2 \quad (3)$$

$$w_i = w_i - \lambda \frac{\partial L}{\partial w_i} \quad (4)$$

where  $\lambda$  is the learning rate, to be decided by the user. The gradient information required for updating the weights is obtained via backpropagation, which involves the successive application of the chain rule backwards through the NN.

The standard GD algorithm does not guarantee good convergence and does not scale well to larger datasets. Several variants of the GD algorithm exist, such as Stochastic GD and Minibatch GD, which deal with the scalability issues of the algorithm [25]. The convergence rates of the algorithm can be improved with methods such as ADaptive Moment Estimation (ADAM) [26], or Adagrad [27].

### Unbounded Gradients in RNNs

Since the edge weights of an RNN evolve with successive inputs, the standard GD algorithm with backpropagation described above would not be sufficient, since equation 4 is only valid for one timestep, i.e., a single input vector. Hence, a modified formulation known as backpropagation through time is used. Consider a set,  $\theta = \{\mathbf{W}_{HH}, \mathbf{W}_{IH}, \mathbf{W}_{HO}, \mathbf{b}_H, \mathbf{b}_O\}$  of the weights and biases of the RNN, and a cumulative objective that is the sum of the objective at each timestep. Then

$$L = \sum_{t=1}^T L_t \quad (5)$$

$$\frac{\partial L}{\partial \theta} = \sum_{t=1}^T \frac{\partial L_t}{\partial \theta} \quad (6)$$

$$\frac{\partial L_t}{\partial \theta} = \sum_{k=1}^t \left( \frac{\partial L_t}{\partial \mathbf{h}_t} \cdot \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \cdot \frac{\partial \mathbf{h}_k}{\partial \theta} \right) \quad (7)$$

Equation 7 describes how the parameters  $\theta$  affect the objective over multiple timesteps. The term  $\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k}$  quantifies the error propagation to timestep  $t$  from timestep  $k < t$  and is given by

$$\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} = \prod_{i=k+1}^t \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} \quad (8)$$

From equation ?? and ??, we get

$$\prod_{i=k+1}^t \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} = \prod_{i=k+1}^t \mathbf{w}_{HH}^\top \text{diag}[f'_H(\mathbf{h}_{i-1})] \quad (9)$$

We observe from equation 8 that the propagation of the error through timesteps is multiplicative in nature. If the gradients between timesteps are small, then the repeated multiplication of these values causes what is known as the *vanishing gradient problem*. This causes the RNN to ignore long-term dependencies, since the propagation between temporally distant inputs is close to zero. Conversely, if the weights  $\mathbf{W}_{HH}$  are large, then their repeated multiplication can cause the gradients to become very large as it is propagated. This is called the *exploding gradient problem*, and the model becomes unstable as a result.

## D. LSTM Networks

To tackle the issue of unbounded gradients, Long-Short Term Memory (LSTM) networks were introduced. LSTMs employ a modified nodal architecture, which controls the information flow between nodes and timesteps. This structure allows LSTM networks to 'remember' information for longer periods. [28]

Figure 2 shows a typical LSTM node architecture. The information flow in and out of the node is controlled by four 'gates', namely the input, forget, output, and cell activation gates. These gates can preserve the data in the memory of the network for multiple timesteps, allowing LSTMs to handle

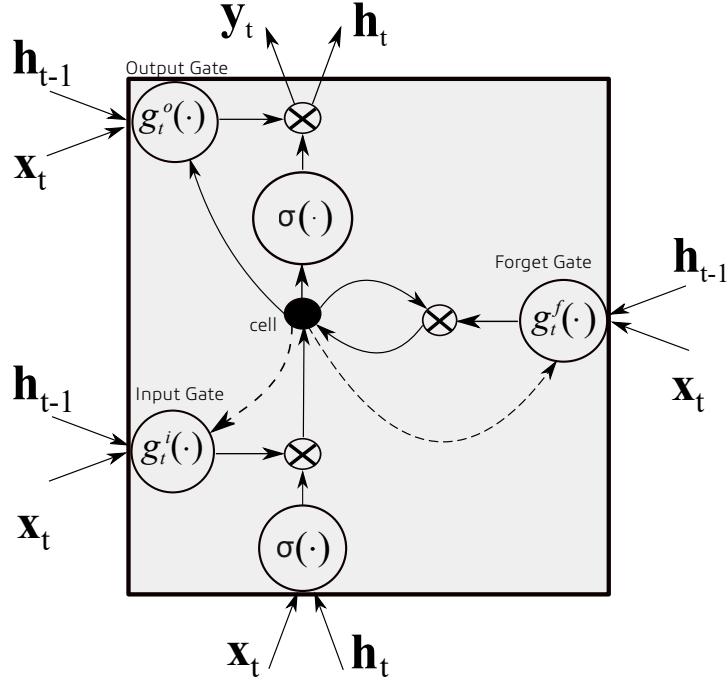


Figure 2: A standard LSTM node, from [28]

temporally distant relations better than standard RNNs, particularly in the area of sequence learning. The input gate to the LSTM is defined as

$$\mathbf{g}_t^i = \sigma(\mathbf{W}_{Ig^i} \mathbf{x}_t + \mathbf{W}_{Hg^i} \mathbf{h}_{t-1} + \mathbf{W}_{g^c g^i} \mathbf{g}_{t-1}^c + \mathbf{b}_{g^i}) \quad (10)$$

where  $\mathbf{W}_{Ig^i}$  is the weight matrix from the input layer to the input gate,  $\mathbf{W}_{Hg^i}$  is the weight matrix from the hidden state to the input gate,  $\mathbf{W}_{g^c g^i}$  is the weight matrix from the cell activation to the input gate, and  $\mathbf{b}_{g^i}$  is the bias of the input gate. Here, the nodal output  $\mathbf{h}_t$  is referred to as the hidden state. The forget gate is defined as

$$\mathbf{g}_t^f = \sigma(\mathbf{W}_{Ig^f} \mathbf{x}_t + \mathbf{W}_{Hg^f} \mathbf{h}_{t-1} + \mathbf{W}_{g^c g^f} \mathbf{g}_{t-1}^c + \mathbf{b}_{g^f}) \quad (11)$$

where  $\mathbf{W}_{Ig^f}$  is the weight matrix from the input layer to the forget gate,  $\mathbf{W}_{Hg^f}$  is the weight matrix from the hidden state to the forget gate,  $\mathbf{W}_{g^c g^f}$  is the weight matrix from the cell activation to the forget gate, and  $\mathbf{b}_{g^f}$  is the bias of the forget gate. The cell gate is defined as

$$\mathbf{g}_t^c = \mathbf{g}_t^i \tanh(\mathbf{W}_{Ig^c} \mathbf{x}_t + \mathbf{W}_{Hg^c} \mathbf{h}_{t-1} + \mathbf{b}_{g^c}) + \mathbf{g}_t^f \mathbf{g}_{t-1}^c \quad (12)$$

where  $\mathbf{W}_{Ig^c}$  is the weight matrix from the input layer to the cell gate,  $\mathbf{W}_{Hg^c}$  is the weight matrix from the hidden state to the cell gate, and  $\mathbf{b}_{g^c}$  is the bias of the forget gate. The output gate is defined as

$$\mathbf{g}_t^o = \sigma(\mathbf{W}_{Ig^o} \mathbf{x}_t + \mathbf{W}_{Hg^o} \mathbf{h}_{t-1} + \mathbf{W}_{g^c g^o} \mathbf{g}_{t-1}^c + \mathbf{b}_{g^o}) \quad (13)$$

where  $\mathbf{W}_{Ig^o}$  is the weight matrix from the input layer to the output gate,  $\mathbf{W}_{Hg^o}$  is the weight matrix from the hidden state to the output gate,  $\mathbf{W}_{g^c g^o}$  is the weight matrix from the cell activation to the output gate, and  $\mathbf{b}_{g^o}$  is the bias of the forget gate. The hidden state is then computed as

$$\mathbf{h}_t = \mathbf{g}_t^o \tanh(\mathbf{g}_t^c) \quad (14)$$

LSTM is one of the most popular methods to deal with the effects of gradient divergence in RNNs and learning long-term dependencies in data. This is especially important for DSR, due to the fact that the hierarchical position of a token in the expression tree is not related to the order in which the tokens are picked, i.e., tokens picked one after the other may not be close to each other in the expression tree. Hence, retaining sequential information for longer periods of time is crucial for the proper functioning of the method.

## E. Proof: Risk-Seeking Policy Gradient

Let  $J_{risk}(\theta; \epsilon)$  denote the conditional expectation of rewards above the  $(1 - \epsilon)$  quantile, denoted as

$$J_{risk}(\theta, \epsilon) = \mathbb{E}_{\tau \sim p(\tau|\theta)}[R(\tau) | R(\tau) \geq R_\epsilon(\theta)] \quad (15)$$

The gradient of  $J_{risk}$  is then given by

$$\nabla_\theta J_{risk}(\theta, \epsilon) = \mathbb{E}_{\tau \sim p(\tau|\theta)}[(R(\tau) - R_\epsilon(\theta)) \cdot \nabla_\theta \log(p(\tau|\theta)) | R(\tau) \geq R_\epsilon(\theta)]$$

Where  $\tilde{R}_\epsilon(\theta)$  is the empirical  $(1-\epsilon)$  quantile of rewards, and  $\mathbf{1}_x$  returns 1 if  $x$  is true and 0 otherwise. The following proof was provided by Petersen et al. [11], which was an extension of results obtained by Tamar et al. [29]

*Proof:* Consider a bounded random variable  $Z \in [-b, b]$ , generated from a parameterised distribution  $p(Z|\theta)$ . The  $(1 - \epsilon)$  quantile of  $Z$  is given by

$$Q_{1-\epsilon}(Z; \theta) = \inf\{z : CDF(z) \geq 1 - \epsilon\}$$

where  $CDF(z)$  is the cumulative distribution function corresponding to  $p(Z|\theta)$ . The risk-seeking objective  $J_{risk}(\theta; \epsilon)$  as the expectation of the  $\epsilon$  fraction of the *best* outcomes of  $Z$

$$J_{risk}(\theta; \epsilon) = \mathbb{E}_{Z \sim p(Z|\theta)}[Z | Z \geq Q_{1-\epsilon}(Z; \theta)]$$

The set of values of  $z$  above this quantile,  $D_\theta$ , is given by

$$D_\theta = \{z \in [-b, b] : z \geq Q_{1-\epsilon}(Z; \theta)\}$$

By construction,  $D_\theta$  is the interval  $[Q_{1-\epsilon}(Z; \theta), b]$ . Further,

$$\int_{z \in D_\theta} p(z|\theta) dz = \epsilon \quad (16)$$

The conditional expectation,  $J_{risk}$ , can be rewritten in integral form as

$$\begin{aligned} J_{risk}(\theta; \epsilon) &= \frac{1}{\int_{z \in D_\theta} p(z|\theta) dz} \int_{z \in D_\theta} p(z|\theta) z dz \\ &= \frac{1}{\epsilon} \int_{z \in D_\theta} p(z|\theta) z dz \\ &= \frac{1}{\epsilon} \int_{Q_{1-\epsilon}(Z; \theta)}^b p(z|\theta) z dz \end{aligned}$$

The calculation of the gradient of  $J_{risk}$  is done with the Leibniz integral rule, and can be expressed as



$$\begin{aligned}
\nabla_{\theta} J_{risk}(\theta; \epsilon) &= \nabla_{\theta} \frac{1}{\epsilon} \int_{Q_{1-\epsilon}(Z; \theta)}^b p(z|\theta) z dz \\
&= \frac{1}{\epsilon} \int_{Q_{1-\epsilon}}^b \nabla_{\theta} p(z|\theta) z dz - \frac{1}{\epsilon} p(Q_{1-\epsilon}(Z; \theta)|\theta) Q_{1-\epsilon}(Z; \theta) \nabla_{\theta} Q_{1-\epsilon}(Z; \theta)
\end{aligned}$$

The gradient of equation 16 may be obtained as

$$\begin{aligned}
0 &= \nabla_{\theta} \int_{z \in D_{\theta}} p(z|\theta) dz \\
&= \nabla_{\theta} \int_{Q_{1-\epsilon}(Z; \theta)}^b p(z|\theta) dz \\
&= \int_{Q_{1-\epsilon}(Z; \theta)}^b \nabla_{\theta} p(z|\theta) dz - p(Q_{1-\epsilon}(Z; \theta)|\theta) \nabla_{\theta} Q_{1-\epsilon}(Z; \theta)
\end{aligned}$$

This result is substituted in the equation for  $\nabla_{\theta} J_{risk}(\theta; \epsilon)$

$$\nabla_{\theta} J_{risk}(\theta; \epsilon) = \frac{1}{\epsilon} \int_{Q_{1-\epsilon}(Z; \theta)}^b \nabla_{\theta} p(z|\theta) (z - Q_{1-\epsilon}(Z; \theta)) dz$$

Multiplying  $\frac{p(Z|\theta)}{p(Z|\theta)}$  and using the derivative of a logarithm, the definition of conditional expectation yields

$$\begin{aligned}
\nabla_{\theta} J_{risk}(\theta; \epsilon) &= \frac{1}{\epsilon} \int_{Q_{1-\epsilon}(Z; \theta)}^b (z - Q_{1-\epsilon}(Z; \theta)) p(z|\theta) \nabla_{\theta} \log p(z|\theta) dz \\
&= \mathbb{E}_{Z \sim p(Z|\theta)} [(Z - Q_{1-\epsilon}(Z; \theta)) \nabla_{\theta} \log p(Z|\theta) | Z \geq Q_{1-\epsilon}(Z; \theta)]
\end{aligned}$$

Replacing  $Z$  with a scalar reward function  $R(\tau)$ , where  $\tau$  is generated from a parameterised distribution  $p(\tau|\theta)$ ,

$$\nabla_{\theta} J_{risk}(\theta; \epsilon) = \mathbb{E}_{\tau \sim p(\tau|\theta)} [(R(\tau) - R_{\epsilon}(\theta)) \nabla_{\theta} \log p(\tau|\theta) | R(\tau) \geq R_{\epsilon}(\theta)]$$

## F. Observations from Hyperparameter Tuning

The sensitivity of the method with respect to the hyperparameters of the NN was investigated. In the parameter tuning tests, DSR was required to discover the Selkov equation for the x-velocity (Equation ??), with a range of values for each investigated parameter. Only one parameter was varied per run, i.e. only a single parameter was altered from the default value for each run. It was found that default values for some of the six investigated hyperparameters were sub-optimal in terms of the epochs to convergence, as elaborated upon below. For each parameter, a convergence plot (Reward Function vs. Epochs for different parameter values) and the time to convergence as the parameter varies is presented.

It was observed that using a significantly larger batch size (relative to the default) resulted in faster convergence. Further, the fraction of unique expressions generated per batch was found to decrease with each epoch. Using a batch of 5000 expressions resulted in discovery of the correct equation in the first epoch (explaining the absence of this curve from figures 3a and 3c). This implies that a larger number of samples are required to fully capture the distribution of expressions represented by the NN.

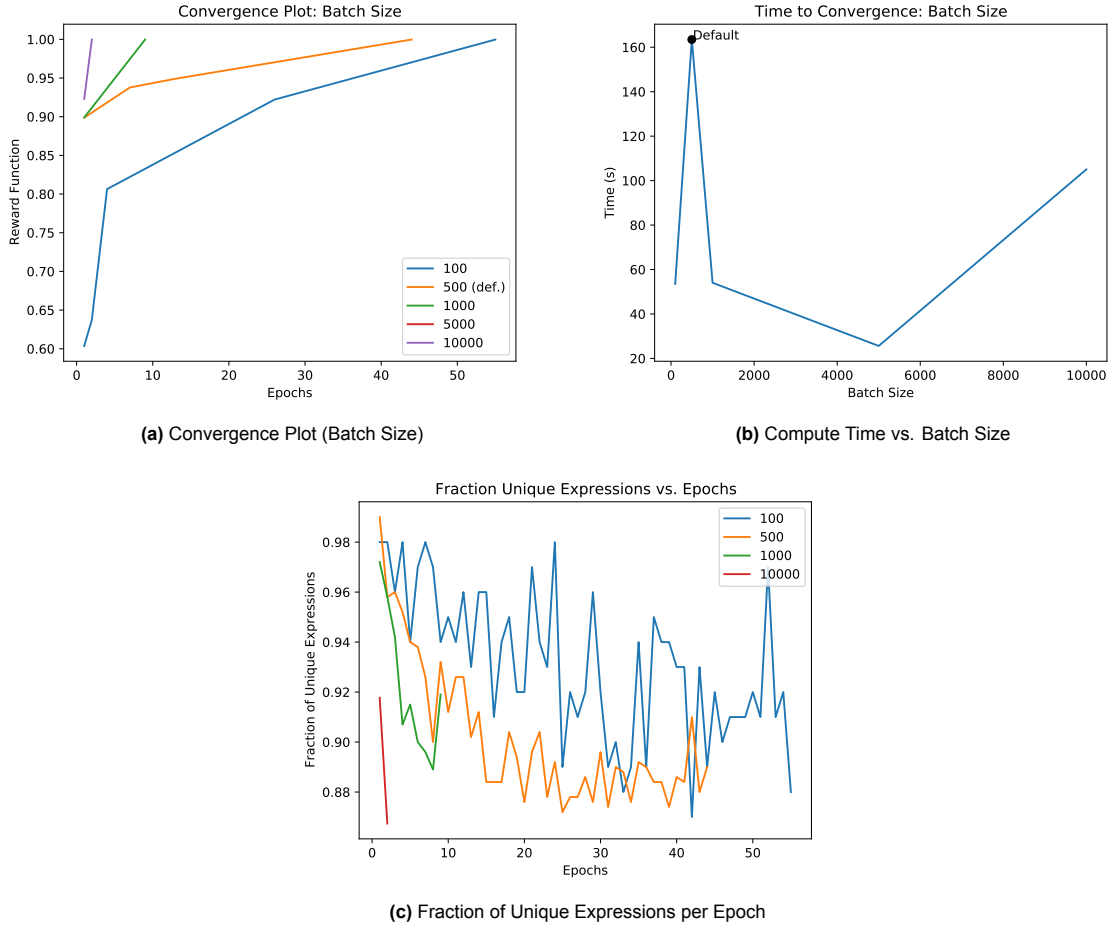


Figure 3: Performance of DSR with varying Batch Size

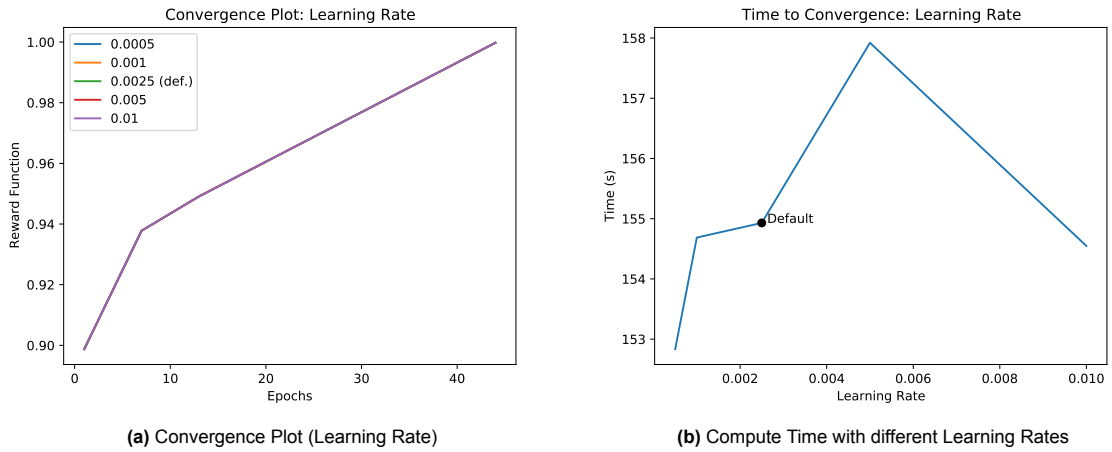
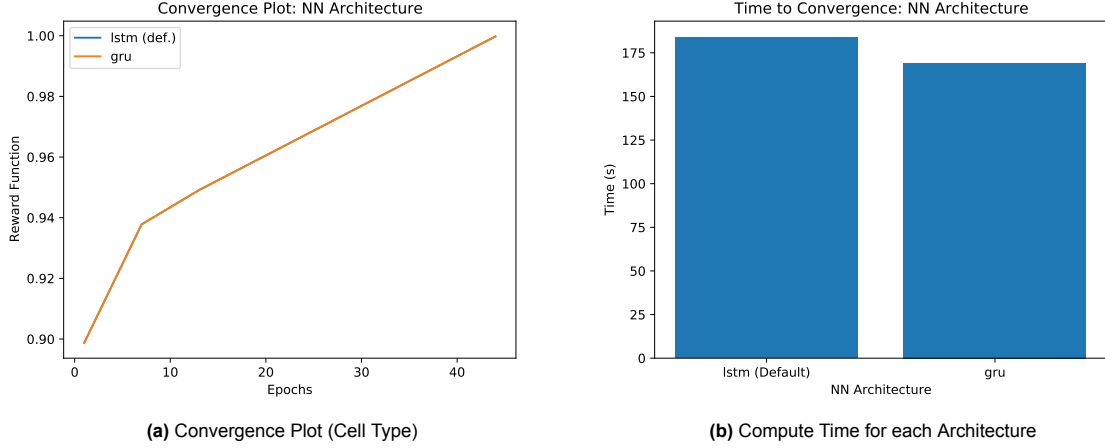


Figure 4: Performance of DSR with varying Learning Rate

An optimal learning rate was found to exist in terms of the learning rate for the discovery of the Selkov equation, which was double the default value. As seen in figures 4a and 4b, using a learning rate of  $5 \cdot 10^{-4}$  yielded the best convergence in the tested range.

Figures 5a and 5b compare the performance of the two available NN architectures for DSR. It was seen that the (default) LSTM architecture had performed worse than the GRU architecture in the discovery

of the Selkov equation, in terms of time to convergence. This may be due to the relative simplicity of the equation in terms of the length of the token sequence required to express it, favouring the more lightweight (and memory-frugal) GRU architecture.



**Figure 5:** Performance of DSR with different NN architectures

Varying the number of layers and units revealed that they do not affect the number of epochs taken to converge in this problem. However, a simpler network is computationally more efficient, and thus using a more lightweight network with fewer layers as units would yield better performance in this case, as seen in figure 6d and figure 6b. The effect of the random seed on the performance of the method is outlined in Figure 7. It is clear that some choices of seed work better than others, and the current default choice (zero) is among the worse performing choices.

## G. Discovery of Modal Contributions in a Cantilever Beam using a Single-Point Measurement

Let the displacement of the beam at the point of measurement be  $w$ . Then,

$$w = \phi_1 q_1 + \phi_2 q_2$$

where  $\phi_i$  is the mode shape of mode  $i$  and  $q_i(t)$  is the time-dependent amplitude associated with mode  $i$ . Here, modes 1 and 2 simply denote two modes whose contribution we wish to identify in the vibration of the beam. They need not necessarily be the first and second modes of the beam (although the first two modes would have the highest contribution to the vibration). For a cantilever beam [30],

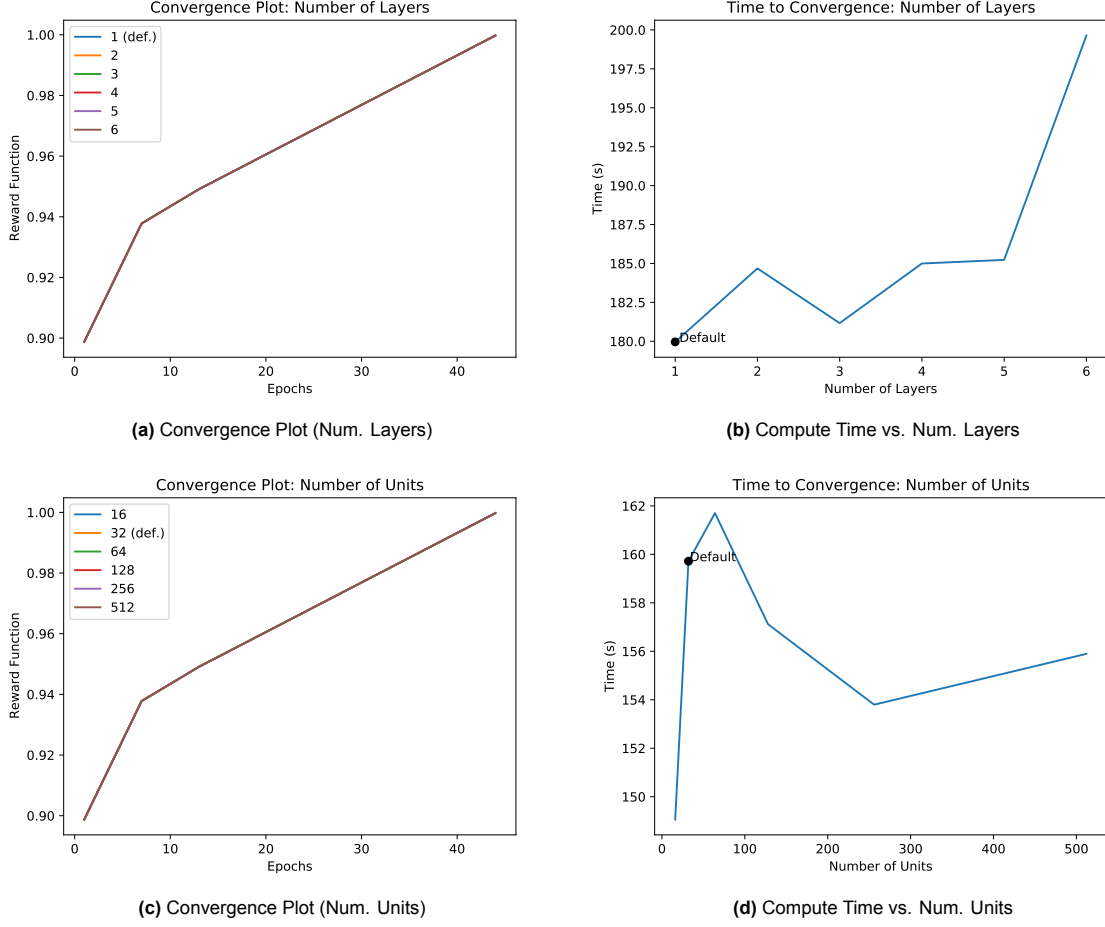
$$\begin{aligned} \phi_n(x) &= (\cos(\beta_n x) - \cosh(\beta_n x)) - K * (\sin(\beta_n x) - \sinh(\beta_n x)) \\ K &= \frac{\cos(\beta_n L) + \cosh(\beta_n L)}{\sin(\beta_n L) + \sinh(\beta_n L)} \end{aligned}$$

here,  $L$  is the length of the beam. The value of  $\phi_i$  may be evaluated at any point along the beam. Hence,  $\phi_i$  takes a constant value for fixed  $x$  and  $i$ . We also have the result

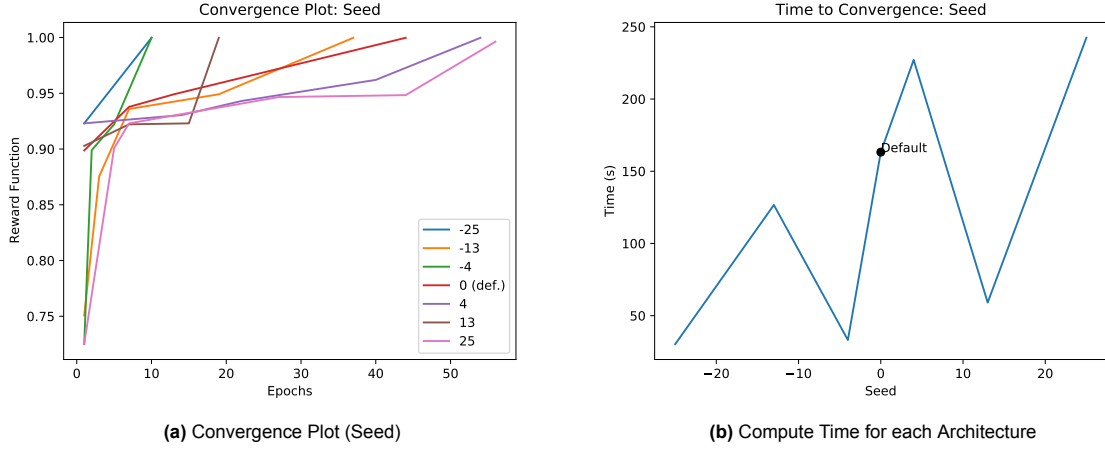
$$\cos(\beta_n L) \cosh(\beta_n L) + 1 = 0$$

Further,

$$\begin{aligned} q_i(t) &= A_i \cos(\omega_i t) + B_i \sin(\omega_i t) \\ \omega_i &= (\beta L)^2 \sqrt{\frac{EI}{\rho A L^4}} \end{aligned}$$



**Figure 6:** Performance of DSR with varying Number of Units/Layers



**Figure 7:** Performance of DSR with different Random Seeds

where  $E$  is the Young's Modulus of the beam material,  $I$  is the area moment of inertia of the beam's cross section,  $\rho$  is the density of the material and  $A$  is the cross-sectional area of the beam. Hence the displacement of the measured point may be expressed as,

$$w = \phi_1 A_1 \cos(\omega_1 t) + \phi_1 B_1 \sin(\omega_1 t) + \phi_2 A_2 \cos(\omega_2 t) + \phi_2 B_2 \sin(\omega_2 t) + \epsilon$$

If data from multiple modes are supplied in the dataset, DSR can select the modes with the highest contribution and provide an equation for the displacement of the measured point in terms of these modes. The constant optimisation routine will determine the values of  $\phi_1 A_1$ ,  $\phi_1 B_1$ ,  $\phi_2 A_2$ ,  $\phi_2 B_2$ . Here,  $\epsilon$  denotes the contribution to the tip displacement from the modes not considered. The presence of  $\epsilon$  may result in an extraneous constant in the expression, a reduction in the value of the reward function, or both. The above equation can be used to discover the two modes with the highest contribution, but the method could, in theory, determine the contribution from more than two modes as well. It is worth noting, however, that the equation discovery performance tends to deteriorate as the number of input variables increases.

# References

- [1] Joseph Lorenzetti et al. “Reduced order model predictive control for setpoint tracking”. In: *2019 18th European Control Conference (ECC)*. IEEE. 2019, pp. 299–306.
- [2] Themistoklis P Sapsis and Andrew J Majda. “Statistically accurate low-order models for uncertainty quantification in turbulent dynamical systems”. In: *Proceedings of the National Academy of Sciences* 110.34 (2013), pp. 13705–13710.
- [3] Sheng Chen and Steve A Billings. “Representations of non-linear systems: the NARMAX model”. In: *International journal of control* 49.3 (1989), pp. 1013–1032.
- [4] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. “Discovering governing equations from data by sparse identification of nonlinear dynamical systems”. In: *Proceedings of the national academy of sciences* 113.15 (2016), pp. 3932–3937.
- [5] Urban Fasel et al. “Ensemble-SINDy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control”. In: *Proceedings of the Royal Society A* 478.2260 (2022), p. 20210904.
- [6] Kadierdan Kaheman, Steven L Brunton, and J Nathan Kutz. “Automatic differentiation to simultaneously identify nonlinear dynamics and extract noise probability distributions from data”. In: *Machine Learning: Science and Technology* 3.1 (2022), p. 015031.
- [7] Daniel A Messenger and David M Bortz. “Weak SINDy for partial differential equations”. In: *Journal of Computational Physics* 443 (2021), p. 110525.
- [8] Josh Bongard and Hod Lipson. “Automated reverse engineering of nonlinear dynamical systems”. In: *Proceedings of the National Academy of Sciences* 104.24 (2007), pp. 9943–9948.
- [9] Michael Schmidt and Hod Lipson. “Distilling free-form natural laws from experimental data”. In: *science* 324.5923 (2009), pp. 81–85.
- [10] Yuntian Chen et al. “Symbolic genetic algorithm for discovering open-form partial differential equations (SGA-PDE)”. In: *Physical Review Research* 4.2 (2022), p. 023174.
- [11] Brenden K Petersen et al. “Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients”. In: *arXiv preprint arXiv:1912.04871* (2019).
- [12] Yulong Lu and Jianfeng Lu. “A universal approximation theorem of deep neural networks for expressing probability distributions”. In: *Advances in neural information processing systems* 33 (2020), pp. 3094–3105.
- [13] Silviu-Marian Udrescu and Max Tegmark. “AI Feynman: A physics-inspired method for symbolic regression”. In: *Science Advances* 6.16 (2020), eaay2631.
- [14] Hao Xu, Haibin Chang, and Dongxiao Zhang. “DI-pde: Deep-learning based data-driven discovery of partial differential equations from discrete and noisy data”. In: *arXiv preprint arXiv:1908.04463* (2019).
- [15] Hao Xu, Dongxiao Zhang, and Junsheng Zeng. “Deep-learning of parametric partial differential equations from sparse and noisy data”. In: *Physics of Fluids* 33.3 (2021), p. 037132.
- [16] Georg Martius and Christoph H Lampert. “Extrapolation and learning equations”. In: *arXiv preprint arXiv:1610.02995* (2016).
- [17] Subham Sahoo, Christoph Lampert, and Georg Martius. “Learning equations for extrapolation and control”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 4442–4450.
- [18] Michael Zhang et al. “Deep Learning and Symbolic Regression for Discovering Parametric Equations”. In: *arXiv preprint arXiv:2207.00529* (2022).
- [19] Zichao Long, Yiping Lu, and Bin Dong. “PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network”. In: *Journal of Computational Physics* 399 (2019), p. 108925.

- [20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.
- [21] Zhiqiang Cai et al. “Deep least-squares methods: An unsupervised learning-based numerical method for solving elliptic PDEs”. In: *Journal of Computational Physics* 420 (2020), p. 109707.
- [22] Varun Kumar Ojha, Ajith Abraham, and Václav Snášel. “Metaheuristic design of feedforward neural networks: A review of two decades of research”. In: *Engineering Applications of Artificial Intelligence* 60 (2017), pp. 97–116.
- [23] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators”. In: *Neural networks* 2.5 (1989), pp. 359–366.
- [24] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. “The global k-means clustering algorithm”. In: *Pattern recognition* 36.2 (2003), pp. 451–461.
- [25] Sebastian Ruder. “An overview of gradient descent optimization algorithms”. In: *arXiv preprint arXiv:1609.04747* (2016).
- [26] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).
- [27] John Duchi, Elad Hazan, and Yoram Singer. “Adaptive subgradient methods for online learning and stochastic optimization.” In: *Journal of machine learning research* 12.7 (2011).
- [28] Hojjat Salehinejad et al. “Recent advances in recurrent neural networks”. In: *arXiv preprint arXiv:1801.01078* (2017).
- [29] Aviv Tamar, Yonatan Glassner, and Shie Mannor. “Optimizing the CVaR via sampling”. In: *Twenty-Ninth AAAI Conference on Artificial Intelligence*. 2015.
- [30] Singiresu S Rao. *Vibration of continuous systems*. John Wiley & Sons, 2019.