REDISCOVERING NEWTON'S GRAVITY AND SOLAR SYSTEM PROPERTIES USING DEEP LEARNING AND INDUCTIVE BIASES

Anonymous authors
Paper under double-blind review

ABSTRACT

We present an approach for using machine learning to automatically discover a physical law and associated properties of the system from real observations. We trained a neural network-based architecture, whose structure corresponds to classical mechanics, to simulate the dynamics of our Solar System from 30 years of observed trajectory data. We then used symbolic regression to extract a symbolic formula for the force law, which our results show matches Newtonian gravity. We find that by scaling the model's predicted acceleration by a trainable scalar variable, we could infer bodies' (relative) masses despite that they were not observable in the data itself. Though "Newtonian" gravity has of course been known since Newton, our approach did not require knowledge of this physical law, and so our results serve as a proof of principle that our method can extract unknown laws from observed data. This work takes a step towards using modern machine learning tools beyond data processing and analysis, but automated scientific theory formation and development.

1 INTRODUCTION

Recent advances in machine learning (ML) have been adopted by many scientific disciplines, from particle physics to cosmology, to process and analyse vast sets of observations. However, there have been relatively fewer applications of ML to a significant component of scientific discovery: theory formation. Here we show how ML methods can exploit established scientific frameworks to discover accurate physical laws and unobserved properties, in an analogous fashion to how scientists develop theories and physical parameters consistent with observations.

Symbolic regression saw its first breakthrough in the sciences with the work of Bongard & Lipson (2007); Schmidt & Lipson (2009), which resulted in the development of the genetic algorithm-based software *eureqa*. This package is commercial, yet remains the most powerful general algorithm capable of finding symbolic physical laws from data (including Lagrangians, Hamiltonians, repeated sub-equations, and other techniques) without known constants or priors on the physical nature of the system. Though we note a few recent exciting papers in this text, the long list of papers that cite this work demonstrate the rich applications of symbolic regression.

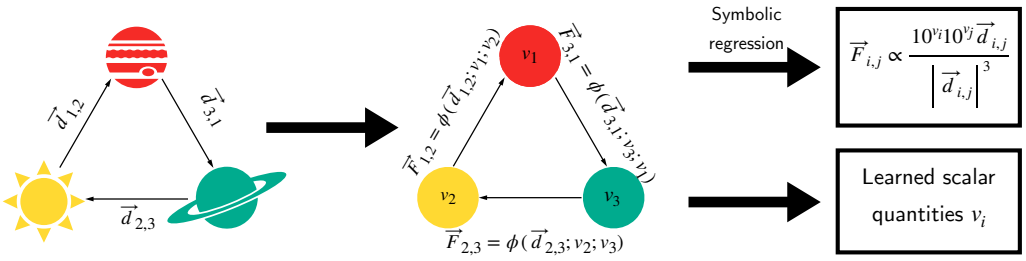

Figure 1: The input is a GN with given distances as edges, and nodes with learnable scalar properties. The GN updates the edges of this graph to compute forces. We then sum over all forces acting on each body (notice that we are assuming that $F_{i,j} = -F_{j,i}$) and divided by the nodes, to get accelerations. The loss function is obtained by comparing this predicted acceleration and the true acceleration. The function that the GN learned to update the edges is also used to recover an algebraic force using a symbolic regression algorithm.

Regarding recent fundamental advances in search algorithms, examples include: Sahoo et al. (2018), which uses gradient descent on a predefined equation up to some depth parametrized inside a neural network; Kusner et al. (2017), which gradient descent on a latent embedding of an equation; Li et al. (2019), a Monte Carlo Tree Search as a learned symbolic regression technique using a neural network guide; and Udrescu & Tegmark (2020), which uses physical priors such as symmetry and dimensional analysis. Brunton et al. (2016) propose SINDy, which targets PDEs, and uses a library-based approach to equation discovery, optimizing sparsity on basis function coefficients using a LASSO-like technique. This technique has evolved into a powerful Koopman theory-based approach which learns linear operators for neural network-learned spatial or time coordinates (Lusch et al., 2018; Lange et al., 2020). Other related PDE techniques include Both et al. (2019); Atkinson et al. (2019); Rackauckas et al. (2020); Chen et al. (2020); Vaddireddy et al. (2020). Finally the method of (Guimerà et al., 2020) develops a method for MCMC sampling in symbolic regression space. In this work we use the method described in Cranmer et al. (2020a), which extends symbolic regression to high-dimensional input such as graphs by using a neural network as a proxy model.

There are also many examples of symbolic regression being applied to specific problems in sciences. A few astronomy-related examples include Graham et al. (2012; 2013), who were the first to apply the approach to the field, and more recently such works as Wadekar et al. (2020), which shares the software package *PySR* (Cranmer, 2020b).

Our approach is based on the framework introduced by (Cranmer et al., 2019; Cranmer et al., 2020a), which used a combination of graph networks (GN) (Battaglia et al., 2018) and symbolic regression. The key principle is that the "edge function" within the GN has a *loose* correspondence to forces in classical mechanics, and so if trained with the correct regularization, the GN's edge function neural network will tend to become equal to the forces, and can be isolated and interpreted using symbolic regression (Cranmer et al. 2020a show that analogous correspondences between Hamiltonians can also be exploited for symbolic formula discovery.). By training a GN to simulate orbital dynamics from real data, we were able to extract the edge function and correctly infer the formula for Newtonian gravitation. We also structured the GN-based simulator to predict accelerations by multiplying the model output by a scalar variable fit during training (corresponding to *force = mass* $\times$ *acceleration*) and found the learned scalars were proportion to the orbital bodies' true masses.

2 APPROACH

Our two-step approach—training a GN-based simulator, then using symbolic regression to find analytical formulae for forces—is summarized in Fig. 1. The data on which the GN is trained consists of the positions and velocities of orbital bodies. We convert this data into distances between each pair of bodies $d_{i,j}$, which are the input to our neural network (NN), and the acceleration per body a_i, which serves as output. Each node is assigned a scalar property v_i. This initial setup is shown in the left of Fig. 1. Our NN then uses the GN formulation to calculate two body interactions $F_{i,j}$ as a function of the distances and input quantities. The loss function is calculated by summing over all interactions for each body, and dividing by 10^{v_i} to predict an acceleration, motivated by $F = ma$. The loss function is then obtained by comparing this predicted acceleration and the observed one. The two body interactions learned by the NN as a function of distance and node properties, are used for symbolic regression, with the aim of recovering Newton's formula for the gravitational force.

This structure is also similar to the one used in Cranmer et al. (2019). The main differences are that we enforce that the acceleration is given as the sum of the interactions divided by the node, while Cranmer et al. (2019) use a second MLP to calculate node updates; and that instead of taking the masses as input data, we attempt to learn them as these node properties v_i. This is important for our approach, where we are trying to learn from observed data, as the masses are not observed, but can possible be inferred from orbits assuming Newtonian gravity.

3 METHODS

Data: We use Solar System data from NASA's HORIZONS On-Line Ephemerys System[1] (Giorgini et al., 1996; 2001). We extract orbits for the Sun, all planets, and the moons that have a mass above $10^{18} M_\odot$. Whilst more bodies could have been considered, their gravitational influence is negligible, therefore we do not expect that their omission will affect our results. A full list of bodies can be found in Tab. 1. In total, our problem consists of 31 bodies. We use data from January 1980 to January 2013 with a time step of 30 minutes, and use the first 30 years of

[1] `http://ssd.jpl.nasa.gov/?horizons`

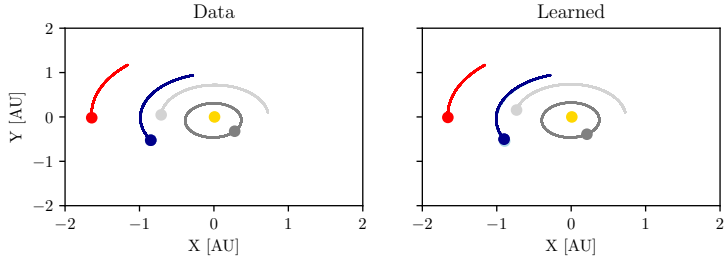

Figure 2: *Left:* data from the bodies of the inner Solar System. *Right:* same bodies evolved from same initial conditions using the learned interaction.

data (approximately one full orbit of Saturn) for training, and the last three for validation. From the HORIZONS interface, we extract positions and velocities in Cartesian coordinates, with the Solar System barycenter as the reference frame.

From this data, we extract the pair-wise distance vectors between bodies and each body's acceleration vector (calculated from changes in the velocities) at every step. Relative distances serve as the input to our model, meaning that our model is blind to a translation of the reference frame. The accelerations serve as the truth for our model training.

Graph Network (GN): A GN is a type of neural network that act on graphs, a 3-tuple formed of three components[2]: **Nodes** $V = \{\mathbf{v}_i\}_{i=1:N^v}$, a number N^v of attribute vectors, each of length L^v; **Edges**: $E = \{(\mathbf{e_k}, r_k, s_k)\}_{k=1:N^e}$, N^e vectors each of length L^e and connecting two nodes, a sender s_k and a receiver r^k; and a single **Global**: $\mathbf{u} \in \mathbb{R}^{L^u}$ vector of length L^u.

Learned simulators based on GNs are able to model very complex particle and mesh systems Sanchez-Gonzalez et al. (2020); Pfaff et al. (2020), but have not been used to learn to simulate real data until now. Here, the input graph's nodes represent orbital bodies, and edges represent possible physical interactions between them. Our input graph has $N^v = 31$ nodes, which are trainable scalars $L^v = 1$. It has a single edge connecting every pair of bodies $N^e = N^v(N^v - 1)/2 = 465$, which has length $L^e = 3$ and is given by the distances between bodies. We do not use any global attributes. Our GN calculates a message function between every pair of bodies, that depends on the node and edge attributes: $\phi^e(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k})$, which is then used to update edge attributes. We then take the sum over all incoming and outgoing edges for each body, which gives us the force acting on each body. By dividing this by the exponential of the node attribute, we obtain a predicted acceleration. The reason we take the exponential of the node attribute, instead of the attribute itself, is to allow for this attribute to have values of largely varying magnitudes for all bodies by being learned in a log scale. This structure is very similar to an interaction network (Battaglia et al., 2016), with the main exception that the edge updates used both sender and receiver nodes, given that we want to impose the symmetry $F_{j,k} = -F_{k,j}$. Specific details about out NN can be found in Appendix A.

Symbolic Regression: Extracting a mathematical formula from input and output data, is a challenging problem due to the vast space of possible expression, and the necessity to fit optimal numerical parameters for each expression (Davidson et al., 2003). Despite recent advances in algorithms as outlined above, trying to learn physics using symbolic regression directly in the data is generally inefficient, and becomes computationally unfeasible once the data becomes large and complex. Thus, following (Cranmer et al., 2020a), our work uses GNs to learn the interaction from the data, and symbolic regression to extract a mathematical expression from the GN model.

We use the package *PySR* (Cranmer, 2020b)[3], which builds on the *eureqa* package (Schmidt & Lipson, 2009) but has several advantages, such as its efficiency, a python wrapper which makes it easily integrated with the rest of the code used in this work, and being open software.

We use the GNs model described above on validation data, to extract the force for different positions of the bodies. Each time step is randomly rotated for data augmentation. For our symbolic regression, we select 1000 of these points, each of which contains the distances between two randomly chosen bodies, the scalar quantities learned by the model for each body, and the three components

[2]This discussion follows the notation of (Battaglia et al., 2018).

[3]`https://github.com/MilesCranmer/pysr`

of the force between them. We then perform symbolic regression for each component of the force, where the input variables are the scalar quantities learned by the model, the three components of the distance vector, and its norm. The allowed operators between these input quantities are addition, subtraction, multiplication and division.

4 RESULTS

Our MLP successfully finds an interaction between bodies that matches the observed acceleration, with a loss of 0.0541 calculated from Eq. (2). This shows that GNs can learn interactions not only from simulations, but also from real data. The interaction learned by our algorithm is shown in comparison with the real data in Fig. 2.

The scalar properties v_i learned by our MLP, which are learned during training and multiplied by the forces to compute the accelerations, are shown in Fig. 3 along with the masses per body. We find that most of them agree within 10%, the exception being planets such as Mercury and Venus, which are planets that have little effect on the Sun's acceleration due to their low masses and their lack of satellites, or moons such as Hyperion, Nereid, or Phoebe, where again a low mass means that they barely affect the planet that they are orbiting. Far from being discouraging, this reinforces our belief that the algorithm is working correctly. Following the equivalence principle, the mass of bodies that lack significant gravitational influence on other bodies in the system is irrelevant to the problem, which is why our MLP fails to learn these masses. From comparison in Fig. 3, we can clearly interpret these learned scalar quantities as learned masses of the bodies.

When applying the symbolic regression we recover Newton's law of gravity, including the values of the scalars v_i which following Fig. 3 we now interpret as the logarithm of learned masses:

$$\vec{F} = -\frac{G_{\text{learned}} M_1 M_2}{r^3}\vec{r}, \tag{1}$$

where $G_{\text{learned}} = 2.88 \times 10^{-4}\ \text{AU}^3 \cdot \text{M}_\odot^{-1} \cdot \text{day}^{-2}$. Our algortithm learns a gravitational constant that is very similar to the true one: $G = 2.96 \times 10^{-4}\ \text{AU}^3 \cdot \text{M}_\odot^{-1} \cdot \text{day}^{-2}$. It is important to point out that all these quantities require an overall calibration, that in our case is the mass of the Sun; i.e. both the masses and the gravitational constant have been learned relative to the mass of the Sun. This is not specific to our approach, and is in fact a feature of the data, the only way to break this degeneracy is with laboratory experiments (Cavendish, 1798; Gillies, 1997).

5 CONCLUSIONS

Our results show that our two-step approach—training a neural network simulator with physical inductive biases, then interpreting what it has learned using symbolic regression—is a viable tool for discovering physical laws from real observations. We (re-)discovered Newton's formula for gravitational force from observed trajectories of the sun, planets, and moons of our Solar System, as well as their relative masses. Even though the law we discovered is already known of course, the purpose of this work is to confirm that known laws are *discoverable* with out method. This is a key step toward building more sophisticated tools for automating the process of scientific discovery, in particular data-driven theory formation and evaluation.

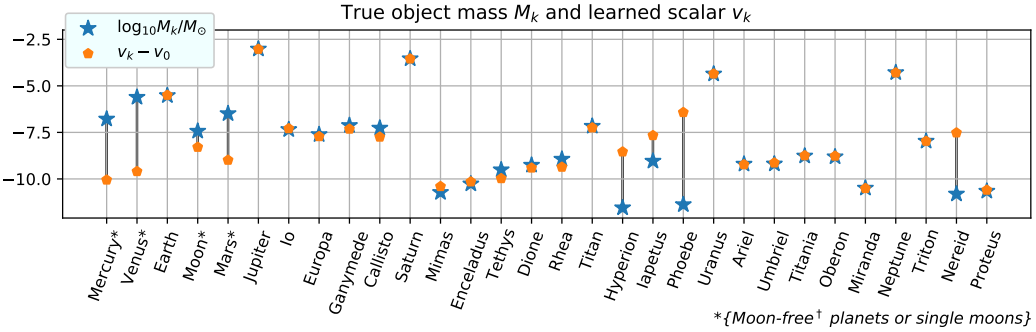

Figure 3: Comparison of the learned scalars v_k relative to the Sun v_0 and the known logarithmic masses of the Solar System bodies (in units of solar mass). Corresponding numbers can be foind in Appendix B. † *Though these planets have moons, they were not included in our learned system.*

REFERENCES

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

Abien Fred Agarap. Deep Learning using Rectified Linear Units (ReLU). *arXiv e-prints*, art. arXiv:1803.08375, March 2018.

Steven Atkinson, Waad Subber, Liping Wang, Genghis Khan, Philippe Hawi, and Roger Ghanem. Data-driven discovery of free-form governing differential equations. *arXiv preprint arXiv:1910.05117*, 2019.

Peter W. Battaglia, Razvan Pascanu, Matthew Lai, Danilo Rezende, and Koray Kavukcuoglu. Interaction Networks for Learning about Objects, Relations and Physics. *arXiv e-prints*, art. arXiv:1612.00222, December 2016.

Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *arXiv e-prints*, art. arXiv:1806.01261, June 2018.

Josh Bongard and Hod Lipson. From the Cover: Automated reverse engineering of nonlinear dynamical systems. *Proceedings of the National Academy of Science*, 104(24):9943–9948, June 2007. doi: 10.1073/pnas.0609476104.

Gert-Jan Both, Subham Choudhury, Pierre Sens, and Remy Kusters. DeepMoD: Deep learning for Model Discovery in noisy data, 2019.

Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016.

Henry Cavendish. Experiments to Determine the Density of the Earth. By Henry Cavendish, Esq. F. R. S. and A. S. *Philosophical Transactions of the Royal Society of London Series I*, 88:469–526, January 1798.

Zhao Chen, Yang Liu, and Hao Sun. Deep learning of physical laws from scarce data, 2020.

Miles Cranmer. Pysr: Fast & parallelized symbolic regression in python/julia, September 2020b. URL `https://doi.org/10.5281/zenodo.4052869`.

Miles Cranmer, Alvaro Sanchez-Gonzalez, Peter Battaglia, Rui Xu, Kyle Cranmer, David Spergel, and Shirley Ho. Discovering Symbolic Models from Deep Learning with Inductive Biases. 6 2020a.

Miles D. Cranmer, Rui Xu, Peter Battaglia, and Shirley Ho. Learning Symbolic Physics with Graph Networks. *arXiv e-prints*, art. arXiv:1909.05862, September 2019.

J.W. Davidson, D.A. Savic, and G.A. Walters. Symbolic and numerical regression: experiments and applications. *Information Sciences*, 150(1):95 – 117, 2003. ISSN 0020-0255. doi: https://doi.org/10.1016/S0020-0255(02)00371-7. URL `http://www.sciencedirect.com/science/article/pii/S0020025502003717`. Recent Advances in Soft Computing.

George T. Gillies. The Newtonian gravitational constant: recent measurements and related studies. *Reports on Progress in Physics*, 60(2):151–225, February 1997. doi: 10.1088/0034-4885/60/2/001.

J. D. Giorgini, D. K. Yeomans, A. B. Chamberlin, P. W. Chodas, R. A. Jacobson, M. S. Keesey, J. H. Lieske, S. J. Ostro, E. M. Standish, and R. N. Wimberly. JPL's On-Line Solar System Data Service. In *AAS/Division for Planetary Sciences Meeting Abstracts #28*, volume 28 of *AAS/Division for Planetary Sciences Meeting Abstracts*, pp. 25.04, September 1996.

J. D. Giorgini, P. W. Chodas, and D. K. Yeomans. Orbit Uncertainty and Close-Approach Analysis Capabilities of the Horizons On-Line Ephemeris System. In *AAS/Division for Planetary Sciences Meeting Abstracts #33*, volume 33 of *AAS/Division for Planetary Sciences Meeting Abstracts*, pp. 58.13, December 2001.

Matthew J. Graham, S. G. Djorgovski, Ashish Mahabal, Ciro Donalek, Andrew Drake, and Giuseppe Longo. Data challenges of time domain astronomy. *arXiv e-prints*, art. arXiv:1208.2480, August 2012.

Matthew J. Graham, S. G. Djorgovski, Ashish A. Mahabal, Ciro Donalek, and Andrew J. Drake. Machine-assisted discovery of relationships in astronomy. MNRAS, 431(3):2371–2384, May 2013. doi: 10.1093/mnras/stt329.

Roger Guimerà, Ignasi Reichardt, Antoni Aguilar-Mogas, Francesco A. Massucci, Manuel Miranda, Jordi Pallarès, and Marta Sales-Pardo. A Bayesian machine scientist to aid in the solution of challenging scientific problems. *Science Advances*, 6(5):eaav6971, January 2020. doi: 10.1126/sciadv.aav6971.

Matt J. Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar Variational Autoencoder, 2017.

Henning Lange, Steven L. Brunton, and Nathan Kutz. From Fourier to Koopman: Spectral Methods for Long-term Time Series Prediction. *arXiv e-prints*, art. arXiv:2004.00574, April 2020.

Li Li, Minjie Fan, Rishabh Singh, and Patrick Riley. Neural-guided symbolic regression with asymptotic constraints. *arXiv preprint arXiv:1901.07714*, 2019.

Lu Lu, Yeonjong Shin, Yanhui Su, and George Em Karniadakis. Dying ReLU and Initialization: Theory and Numerical Examples. *arXiv e-prints*, art. arXiv:1903.06733, March 2019.

Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9:4950, November 2018. doi: 10.1038/s41467-018-07210-0.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W Battaglia. Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*, 2020.

Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, and Ali Ramadhan. Universal differential equations for scientific machine learning. *arXiv preprint arXiv:2001.04385*, 2020.

Subham Sahoo, Christoph Lampert, and Georg Martius. Learning Equations for Extrapolation and Control. volume 80 of *Proceedings of Machine Learning Research*, pp. 4442–4450, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL `http://proceedings.mlr.press/v80/sahoo18a.html`.

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, pp. 8459–8468. PMLR, 2020.

Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009. ISSN 0036-8075. doi: 10.1126/science.1165893. URL `https://science.sciencemag.org/content/324/5923/81`.

Silviu-Marian Udrescu and Max Tegmark. AI Feynman: a Physics-Inspired Method for Symbolic Regression. *Sci. Adv.*, 6(16):eaay2631, 2020. doi: 10.1126/sciadv.aay2631.

Harsha Vaddireddy, Adil Rasheed, Anne E. Staples, and Omer San. Feature engineering and symbolic regression methods for detecting hidden physics from sparse sensor observation data. *Physics of Fluids*, 32(1):015113, 2020. doi: 10.1063/1.5136351. URL https://doi.org/10.1063/1.5136351.

Digvijay Wadekar, Francisco Villaescusa-Navarro, Shirley Ho, and Laurence Perreault-Levasseur. Modeling assembly bias with machine learning and symbolic regression. *arXiv e-prints*, art. arXiv:2012.00111, November 2020.

A NEURAL NETWORK

The GN uses a TensorFlow (Abadi et al., 2015) model with three multilayer perceptrons (MLPs) with 128 hidden nodes per layer. Furthermore, our model has the followng:

- **Activation function**: We use a hyperbolic tangent 'tanh' activation function. While this is slower than the very commonly used Rectified Linear Unit (ReLU) activation function (Agarap, 2018), our problem is very susceptible to the dying ReLU problem (Lu et al., 2019) due to the very different values of both inputs and outputs.
- **Loss function**: For the loss function, we use the relative mean weighted error:

$$\text{Loss} = \sum \frac{\left(a^{\text{pred}} - a^{\text{true}}\right)^2}{\left(a^{\text{true}}\right)^2}. \tag{2}$$

  The reason we use the relative mean weighted error is to make sure the NN tries to fit the orbits of all bodies considered, not only the ones with the largest acceleration.
- **Spherical coordinates**: Our NN takes as inputs a 3-vector for every pair of bodies representing the distances, and outputs a second three vector representing the force. However, due to the large variation in the magnitude of the inputs, we transform the input distance from Cartesian into spherical coordinates, and then use the base-ten logarithm of the magnitude, as well as the two angular coordinates, as inputs. Similarly, the output force is transformed from base ten logarithm and angular coordinates back into Cartesian, as this allows the NN to learn forces of very different magnitudes.
- **Data augmentation**: During training, a random three-dimensional rotation is applied to every point. This serves as a data augmentation technique, and also as a way to prevent a bias in the learning as some of the bodies with the largest orbital period do not complete an entire orbit in the thirty years that are used as training data.
- **Training noise:**: During training, we corrupted the input states with Gaussian noise to improve the model's robustness to error over long rollouts at test time. This technique has been used widely in GNN-based learned simulators Sanchez-Gonzalez et al. (2020); Pfaff et al. (2020): it is believed to help the model close the gap between the distribution of training input states, which are always from the true observations, and rollout input states, which are predicted by the model.
- **Early stopping**: We stop training after 20 steps with no improvement in the validation loss, to prevent overfitting.

B LEARNED MASSES

The exact values for the learned and true masses are shown graphically in Fig. 3, can be found in Tab. 1

Name	$\log_{10} M/M_\odot$	$v_k - v_0$	Name	$\log_{10} M/M_\odot$	$v_k - v_0$
Mercury	-6.78	-10.05	Rhea	-8.94	-9.36
Venus	-5.61	-9.60	Titan	-7.17	-7.25
Earth	-5.52	-5.50	Hyperion	-11.55	-8.54
Moon	-7.43	-8.29	Iapetus	-9.04	-7.66
Mars	-6.49	-8.99	Phoebe	-11.38	-6.42
Jupiter	-3.02	-3.03	Uranus	-4.36	-4.35
Io	-7.35	-7.30	Ariel	-9.20	-9.23
Europa	-7.62	-7.72	Umbriel	-9.19	-9.14
Ganymede	-7.13	-7.32	Titania	-8.77	-8.75
Callisto	-7.27	-7.75	Oberon	-8.81	-8.77
Saturn	-3.54	-3.55	Miranda	-10.49	-10.51
Mimas	-10.72	-10.40	Neptune	-4.29	-4.29
Enceladus	-10.27	-10.27	Triton	-7.97	-7.97
Tethys	-9.51	-9.98	Nereid	-10.81	-7.52
Dione	-9.26	-9.41	Proteus	-10.66	-10.60

Table 1: The true masses of the Solar System bodies considered in this problem, and the scalar quantities learned by our GN. We subtract by v_0, as we can pick an overall normalization, equivalent to choosing units.