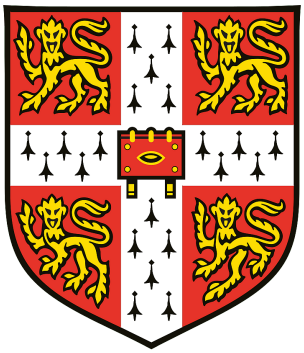


UNIVERSITY OF
CAMBRIDGE

Discovering Symbolic Models from Deep Learning with Inductive Biases - Reproduction Report

Vishal Jain

Department of Physics

MPhil in Data Intensive Science

Supervisor: Dr Miles Cranmer

A dissertation presented for the degree of
Master of Philosophy

University of Cambridge
July 2, 2024

Word Count: 6100

Contents

1 Introduction

The aim of this report is to reproduce the key results of the paper *Discovering Symbolic Models from Deep Learning with Inductive Biases* by *Cranmer et al.* [2]. The main contributions of the paper are two fold - first the presentation of a systematic framework whereby inductive biases can be employed to distill low dimensional symbolic approximations of learned neural networks. The second is showing its successful application of the framework in 3 contexts: rediscovering force laws, rediscovering Hamiltonians and discovering a new equation for a nontrivial cosmology problem. The scope of this report is limited to the reproducibility of the experiments relating to the rediscovery of force laws. However, in doing so, the wider contribution of the framework is validated.

1.1 Motivation

Why would it be desirable to approximate a high dimensional neural network with a low dimensional symbolic model? The answer lies in the scientific method itself and the unreasonable effectiveness of mathematics in the natural sciences. When using neural networks as a tool for science, the lack of interpretability is a major drawback. Science is not just about making predictions, but also about understanding the underlying mechanisms that govern the phenomena being studied. Neural networks are famously known to be black boxes, with the decision-making process being opaque to the user. If they are to be used for science, there needs to be some distillation of what the network has learned in a form that is interpretable to humans. Symbolic models are a natural choice for this task, as they provide a compact and interpretable representation of the underlying mechanisms due to their low dimensionality. Further, they are likely to generalise better to out of distribution data than the neural network itself, indeed this was the shown to be the case in the findings presented by the original paper. Choosing to describe the world in terms of closed form low dimensional mathematical equations seems to be a very good inductive bias.

1.2 Inductive Biases

Understanding effective inductive biases is crucial in machine learning. Inductive biases are the priors placed on the space of solutions by a given set of assumptions made explicitly and implicitly when using any modelling process. To demonstrate the importance of selecting appropriate inductive biases, consider an experiment involving a neural network and symbolic regression. In this experiment, depicted in Figure 1, a neural network with a single layer of 100 ReLU activations is trained on data sampled from a sinusoidal function. Due to its architecture, the network employs linear piece-wise approximations, which may fit the training data adequately but will extrapolate linearly, leading to poor generalization on out-of-distribution data. This illustrates a critical limitation of using such a high-dimensional neural network with ReLU activations for this task. Conversely, applying symbolic regression—a supervised machine learning method that constructs analytic equations—to the same dataset yields the equation $y = \sin(x)$, which not only perfectly describes the training data but also generalises perfectly to out of distribution data. This approach uses the inductive bias that the underlying function can be modelled by a simple, closed-form equation. This bias towards simplicity provides superior generalisation compared to the neural network model.

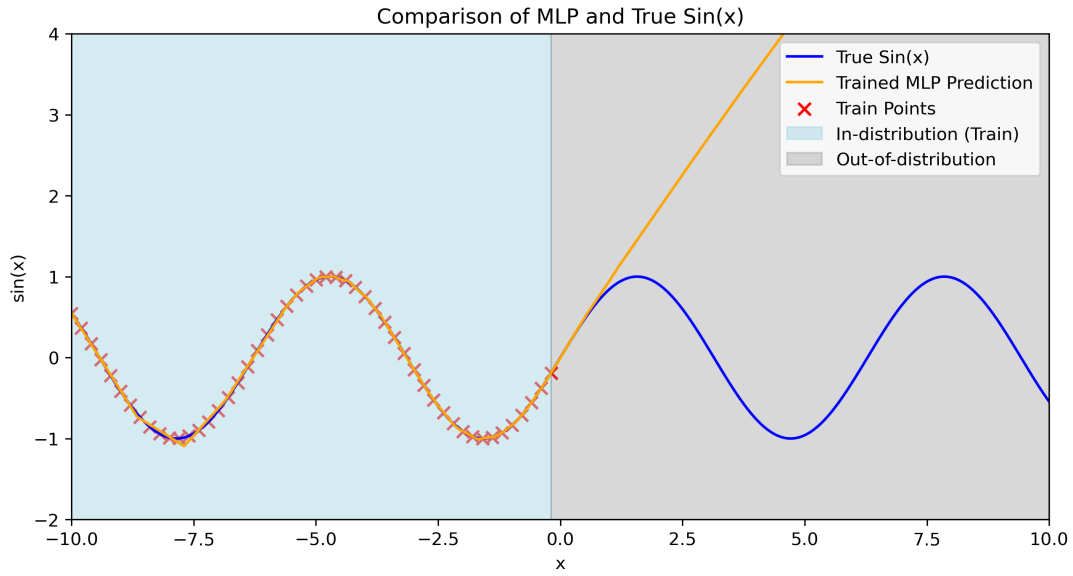


Figure 1: Neural network prediction (one layer network with width 100 and ReLU activations) versus symbolic model for sinusoidal dataset. The symbolic model is $y = sin(x)$.

1.3 Symbolic Distillation Framework

The general framework prescribed by the paper for symbolic distillation is as follows:

- Design a neural network with an architecture that has a seperable internal structure and an inductive bias that is well suited to modelling the underlying problem.
- Train the model with regularisation techniques that encourage the network to learn a low-dimensional representation of the data.
- Replace a learned component of the neural network with a symbolic model that approximates that component's output.
- Retrain the neural network and repeat the process until all components of the neural network have been replaced with low dimensional symbolic models.

This framework may prompt the question: what is the necessity of employing neural networks initially? Why not directly utilise symbolic regression? The issue arises from the high dimensionality of modern datasets, which renders the application of symbolic regression, essentially a brute-force search over the space of all possible equations, computationally infeasible. To illustrate this point and the motivation behind this work, consider the problem of discovering a force law from a dataset of observed instantaneous accelerations of particles $\mathbf{a}_i \in \mathbb{R}^n$ and their positions $\mathbf{x}_i \in \mathbb{R}^n$. Assume a force law of the form $\mathbf{f}_{ij} = -k|\mathbf{x}_i - \mathbf{x}_j|$, with $\mathbf{a}_i = \frac{\mathbf{F}_i}{m_i}$, where $\mathbf{F}_i = \sum_j \mathbf{f}_{ij}$. To learn symbolic models for the acceleration $\mathbf{p}(\mathbf{x})$ and the pairwise force $\mathbf{q}(\mathbf{x})$, one can apply the inductive bias $\mathbf{a}_i = \mathbf{p}(\sum_j \mathbf{q}(\mathbf{x}_i, \mathbf{x}_j))$. If the symbolic regression model needs to search N equations to fit a given model, the search space scales as $O(N^2)$. However, if neural networks are first used to approximate the functions $\mathbf{p}(x)$ and $\mathbf{q}(x)$, and symbolic regression is subsequently applied to approximate these neural networks, the models can now be fit independently, reducing the search space to $O(N)$. If the dimensionality of the problem n increases or the force law becomes more complex, the number of equations N required to search also increases significantly. Therefore, this factorization of the search space is essential to make symbolic regression tractable for high-dimensional problems.

1.4 Graph Neural Networks

The work in the original paper and this report is primarily related to the class of problems which can be classed as interacting particle problems. This type of problem describes a very broad range of most problems encountered in physics. The graph neural network architecture applies a well suited inductive bias to these problems. This section will briefly describe the fundamental components of the graph neural network architecture and why it is well suited to interacting particle problems.

A graph neural network (GNN) is a specialised architecture designed to operate on graph data structures, where nodes represent particles and edges represent pairwise interactions between particles. In this context, each node has an associated feature vector encoding properties such as position, velocity, mass, and charge. The GNN operates through a scheme of message passing, involving two main components: the edge model (ϕ^e) and the node model (ϕ^v).

Edge Model (ϕ^e):

- The edge model is applied to each edge k in the graph.
- It takes as input the feature vectors of the two nodes $\mathbf{x}_i, \mathbf{x}_j$, connected by the edge.
- Formally, $\phi^e : \mathbb{R}^{d_v} \times \mathbb{R}^{d_v} \to \mathbb{R}^{d_e}$, where d_v is the dimensionality of the node feature vectors, and d_e is the dimensionality of the edge model's output, which is called the edge message $\mathbf{e}_k$.

Node Model (ϕ^v):

- Once every edge has an associated edge message, the node model is applied to each node in the graph.
- It takes as input the feature vector of the node, x_i, and the aggregated edge messages from all inbound edges, $\bar{e_k}$.
- Aggregation is performed using a summation operator.
- Formally, $\phi^v : \mathbb{R}^{d_v} \times \mathbb{R}^{d_e} \to \mathbb{R}^{d'_v}$, where d'_v is the dimensionality of the updated node feature vectors.
- The output of the node model is the updated node feature vector.

This message-passing process can be repeated for a number of layers, with the output of the final layer being the output of the graph neural network. The computational graph of the GNN is shown in Figure 2, illustrating the interactions between nodes and edges through the edge and node models. The graph neural network is an appropriate inductive bias for a variety of reasons. Firstly, many important forces in physics are defined on pairs of particles, analogous to the edge model in this context. Further, the summation that aggregates messages is analogous to the calculation of the net force on a receiving particle. Hence, the edge function can be interpretted as the application of the force law and the node function as the application of Newton's second law: acceleration equals the net force (the summed message) divided by the mass of the receiving particle. Further, the graph neural network architecture is invariant under particle permutations, which is a desirable property for a model which represents interacting particle problems. It will be the edge and node models of the graph neural network that will be replaced by symbolic models in the symbolic distillation framework. The goal will be to recover the underlying force law for the edge model and Newton's second law for the node model.

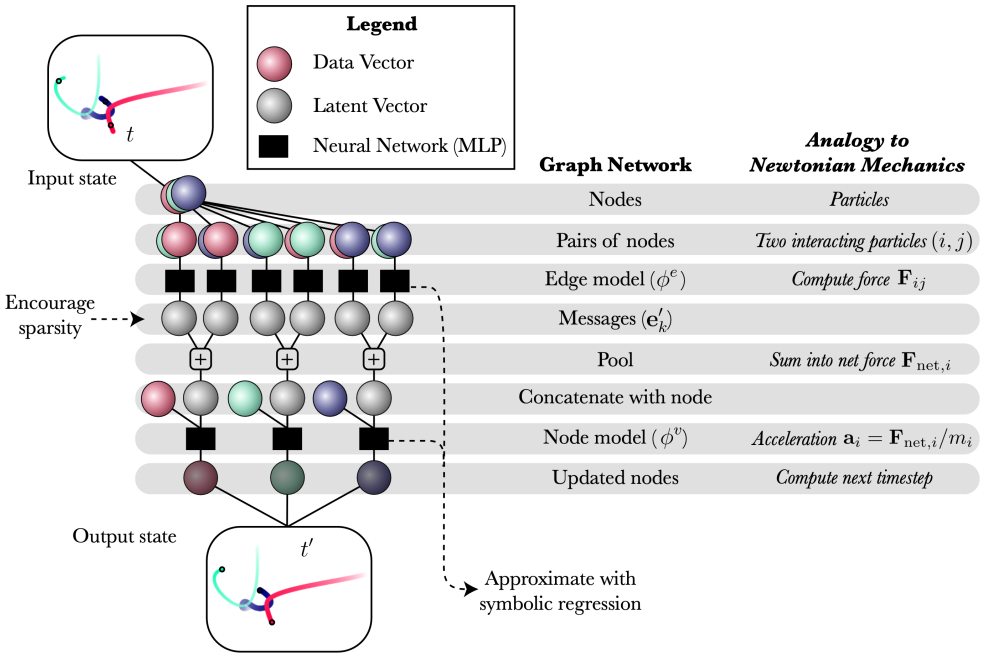


Figure 2: Computational graph of the graph neural network architecture. Diagram used from Cranmer et al. [2]

2 Data

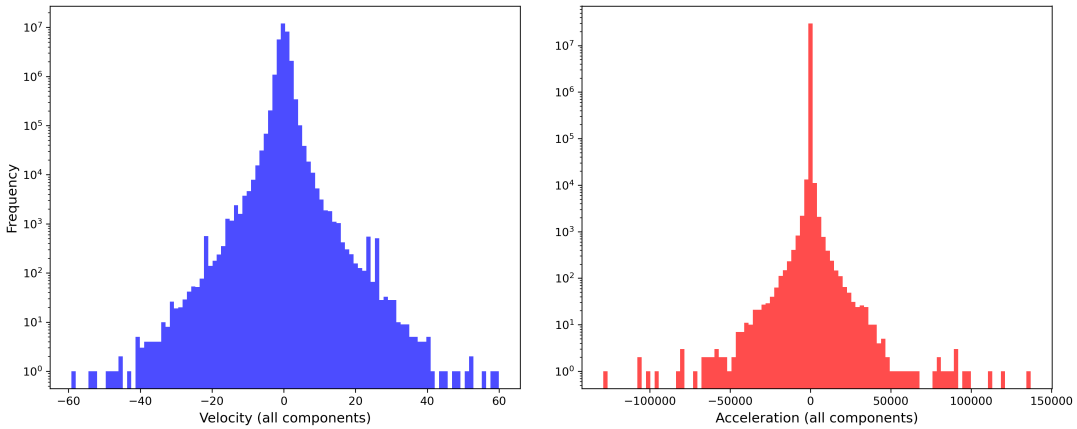
This section will outline the data used in the experiments and the preprocessing applied. Each dataset used describes an ensemble of N-body particle simulations under a specific force law. From the original paper, the following force laws are investigated:

- Spring: $\mathbf{f}_{ij} = -(|\mathbf{r}_{ij} - 1|)\hat{\mathbf{r}}_{ij}$
- $\frac{1}{r}$ Orbital force: $\mathbf{f}_{ij} = -\frac{m_i m_j \hat{\mathbf{r}}_{ij}}{|\mathbf{r}_{ij}|}$
- $\frac{1}{r^2}$ Orbital force: $\mathbf{f}_{ij} = -\frac{m_i m_j \hat{\mathbf{r}}_{ij}}{|\mathbf{r}_{ij}|^2}$
- Charge: $\mathbf{f}_{ij} = -\frac{q_i q_j \hat{\mathbf{r}}_{ij}}{|\mathbf{r}_{ij}|^2}$

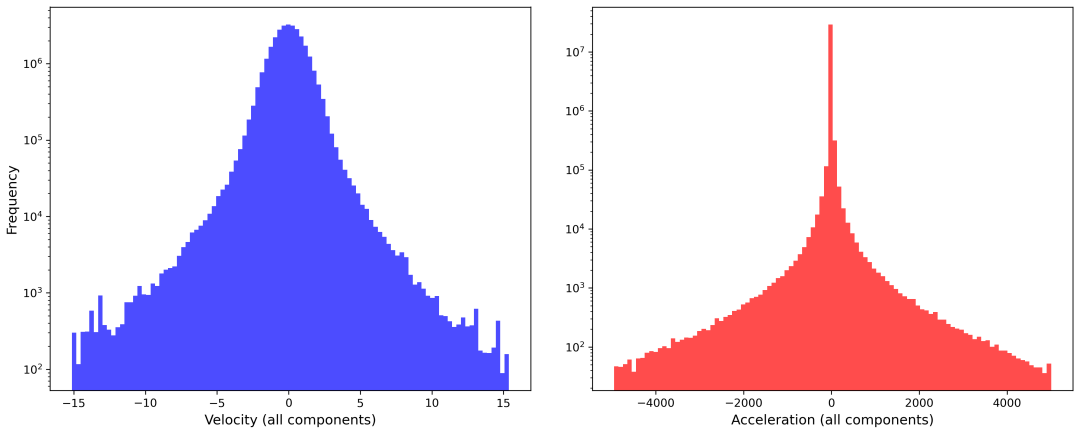
Each force law was simulated in 2D and 3D, with 4 and 8 particles respectively. For each experiment, 10000 simulations were run in parallel for 500 time steps. This was split 3:1 to obtain the training and validation set. The test set was generated separately and of the same size as the combined train and validation sets. The initial positions and velocities were sampled from a standard normal distribution, the mass was sampled from a log standard normal distribution and the charge was drawn randomly from the set $\{-1, 1\}$. It is worth noting that the original paper ran each simulation for 1000 time steps, however, to avoid model training times becoming prohibitively long, the simulations were run for half the number of time steps. The step size varies for each simulation due to the differences in scale. It is: 0.005 for $\frac{1}{r}$, 0.001 for $\frac{1}{r^2}$, 0.01 for Spring, 0.001 for Charge. The code used for generating the simulations is available in the repository within the `simulations` directory, sourced from the repository associated with the original paper.

2.1 Preprocessing

Due to the dependence of orbital and charge force laws on r raised to a negative power, force values can dramatically increase when particles are close together. This leads to data varying over a scale too extensive for effective neural network learning. To address this, a small epsilon (10^{-2}) was added to the denominator to compress r values into a more manageable range. Nonetheless, velocity and acceleration distributions in some experiments still displayed extremely large outliers. To further refine the data, velocities and accelerations were pruned based on the 0.01% and 99.99% percentiles. If any particle in a given timestep exceeded these bounds, the entire graph for that timestep was removed. This process ensured that only timesteps where all particle values were within acceptable limits were retained, enhancing data quality and consistency. Although this method disrupts the temporal continuity of the data, it is important to note that each timestep constitutes a complete graph input for the neural network, and thus time continuity need not be maintained. The impact of this preprocessing is illustrated on the 2D Charge data in Figure 3.



(a) Charge 2D original distribution



(b) Charge 2D pruned distribution

Figure 3: Velocity and acceleration distributions for data generated using the charge 2d force law before and after pruning outlier timesteps.

3 Method

This section outlines the method used to train a graph neural network to predict instantaneous accelerations and then distil symbolic models from the trained network.

3.1 Model Architecture

The architecture of the graph neural network (GNN) employed across various training strategies is consistent, except for variations in the output dimension of the edge model. The GNN comprises two multi-layer perceptrons (MLPs): the edge model and the node model.

Edge Model This model consists of two hidden layers, each with 300 units and ReLU activation functions. It takes as input the concatenated feature vectors from a pair of nodes connected by an edge in the graph. Each node's feature vector is a vector of size $(2d + 2)$ containing the position, velocity, mass, and charge, where d represents the problem's dimensionality (2 or 3). The output dimension of the edge model varies depending on the training strategy, discussed next.

Node Model Mirroring the edge model in structure, the node model takes as input the summed edge messages from all inbound edges to a given node, concatenated with the node's feature vector. Its output is a vector of dimension 2 or 3, corresponding to the predicted instantaneous acceleration for the node.

3.2 Training strategies

This report investigates 4 of the training strategies outlined in the original paper. The training strategies are as follows: standard, bottleneck, L_1 and KL.

Standard Strategy The standard training strategy serves as the baseline case where the edge message dimensionality is 100 and the loss does not include any regularisation terms. Here the loss function is given by the mean absolute error (MAE) between the predicted and true accelerations.

KL Strategy The KL strategy involves sampling edge messages from a multivariate normal distribution as defined by the output of the edge model, which is now a 200-dimensional vector, with the first 100 dimensions representing the mean and the second 100 dimensions representing the log variance. Additionally, the average Kullback-Leibler divergence penalty over all edge messages, using a standard normal as the prior, is added to the loss. The weight of this term is set to 1.0.

$\mathbf{L}_1$ Strategy The L1 strategy is the same as the standard, except it now incorporates the L1 norm of the edge message as a regularisation term, averaged over all edge messages, in the loss. The weight of this term is set to 1.0×10^{-2}.

Bottleneck Strategy The bottleneck strategy is the same as the standard, however now aligns the dimensionality of the edge message with the dimensionality of the problem (2 or 3).

3.3 Model Training

The network is trained over 100 epochs using the Adam optimizer with an initial learning rate of 0.001 and weight decay set to 1.0×10^{-8}. It is worth noting that this is half the number of epochs used in the original paper. The learning rate is modulated by the 1cycle learning rate policy [3], which increases the learning rate to a peak of 0.001 before gradually decreasing it by a factor of $100,000$, adhering to a pre-defined schedule. Random translation is used as a form of data augmentation to improve the model's generalisation capabilities. The model is trained on a single NVIDIA A100 GPU.

3.4 Symbolic Regression of the Network

After training the model end to end, the edge and node model are symbolically distilled using the `PySR` package [1]. PySR's internal search algorithm is a multi-population evolutionary algorithm, which consists of a unique evolve-simplify-optimise loop, designed for optimisation of unknown scalar constants in newly-discovered empirical expressions. 100 populations were used and evolved for 100 iterations. The operators considered in the fitting process are $+, -, \times, /$ as well as real constants. Note, this is a subset of the operators used in the original paper when fitting for symbolic models. This was done to reduce the search space and quickly check that the symbolic distillation framework was working as expected.

3.4.1 Model Selection Criteria

The best symbolic model is selected by evaluating the MAE and the complexity of the model. The complexity is scored by counting the number of occurrences of each operator, constant, and input variable. PySR outputs several equations at each complexity level. The chosen formula is one that maximises the fractional drop in MAE over an increase in complexity from the next best model.

3.4.2 Symbolic Distillation of the Edge Model

Symbolic distillation of the edge model is achieved by sampling inputs and outputs from the test set. The recorded inputs are the concatenated feature vectors of the sending and recieving node, which contain their positions, velocities, masses, and charges. For symbolic distilation of force laws based on relative positions, the positions are transformed into relative displacements ($\Delta x = x_1 - x_2$, similarly for y and z in 3D) and distance ($r = \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}$) is calculated. These form the inputs to the symbolic model along with the charges and masses of the two bodies connected by the sampled edge. The recorded outputs are the corresponding 100 dimensional edge messages. For symbolic distilation, the outputs are derived by selecting the d most significant components of the edge message. Where significance is defined by the standard deviation of the component across the sampled edge messages and d refers to the problem's dimensionality (2D or 3D). This process is summarised in Figure 4. In the original paper, 5000 samples are collected from the test set, however in this analysis 1000 samples are used to reduce the computational cost of performing symbolic regression.

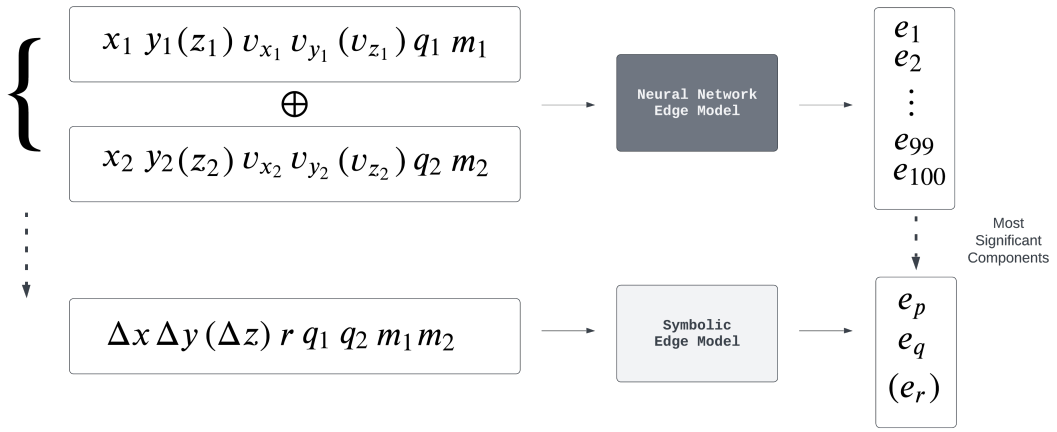


Figure 4: Symbolic distillation of the edge model.

3.4.3 Symbolic Distillation of the Node Model

The symbolic distillation of the node model is similar to that of the edge model. The recorded inputs are the node's feature vectors along with the aggregated edge messages which are calculated by summing over all the inbound edge messages. For symbolic distillation, only the d most significant components of the aggregated edge message are used. The output of the symbolic model is the instantaneous acceleration of the node. This process is summarised in Figure 5.

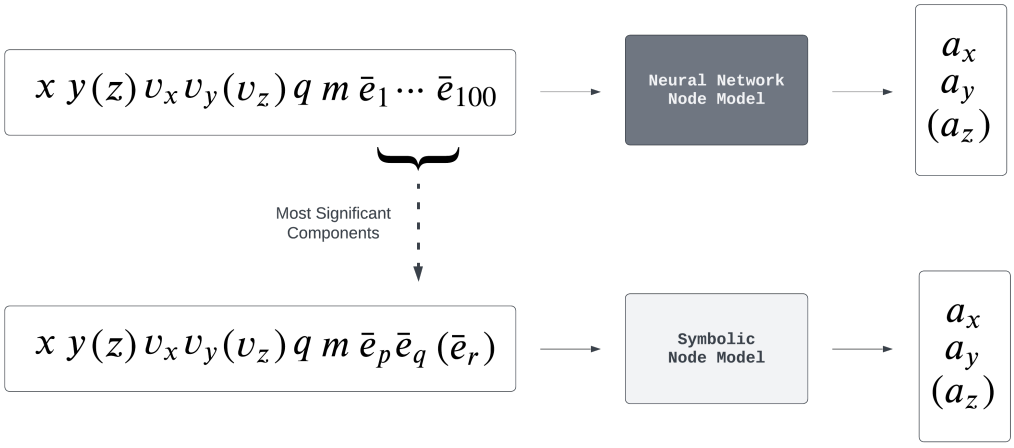


Figure 5: Symbolic distillation of the node model.

3.5 Evaluation Criteria

To evaluate model performance, the mean absolute error (MAE) between the predicted and true accelerations is calculated over the test set. This metric assesses the predictive capabilities of the trained model. Additionally, the fraction of the total standard deviation contained by the top d edge messages is computed to evaluate the compactness of the learnt representations. This metric indicates whether the training resulted in a model that correctly captured the problem's dimensionality. The R^2 metric between the significant edge message components and a linear transformation of the true pairwise force is also calculated. The specific linear transformation is found through a minimisation of the mean squared error (MSE) between observed message components and those predicted by a linear combination of force components. This is implemented through using the `minimize` function from the `scipy.optimize` module, employing the Powell method. The optimisation only considers the MSE errors within the 90th percentile of their absolute values, thereby reducing the influence of outliers. This is the same method used in the original paper. The appendix of the original paper provides a mathematical argument for why the model would learn a linear transformation of the pairwise force. The trained models are evaluated through symbolically regressing its edge and node models and checking how many components yield a successful reconstruction describing linear transformation of the true pairwise force and Newtons second law respectively. Finally, for experiments where all edge and node model components were successfully distilled, the symbolic models are compared to the neural networks by evaluating the MAE loss on the test set.

4 Results

4.1 Model Performance

The first set of results to be presented are the mean absolute error (MAE) loss on the test set for each simulation. The results are presented in Table 1. Two conclusions can be arrived at

from this set of results. Firstly, that the KL is the worst by a significant margin as it always has an MAE and standard deviation value that is atleast an order of magnitude larger than those given by the other strategies. The second observation is that while the bottleneck strategy gives the best performance in five out of eight simulations, the margin by which it leads is minimal. This suggests that the other strategies exhibit comparable efficacy in optimising for MAE, with no clear dominant strategy across the board.

Sim	Standard	Bottleneck	L_1	KL
Charge-3	**0.110** ± 0.015	**0.140** ± 0.022	0.157 ± 0.019	3.325 ± 0.168
Charge-2	**0.466** ± 0.211	**0.462** ± 0.186	**0.483** ± 0.178	3.267 ± 0.417
r^{-2}-3	**0.1753** ± 0.042	**0.1746** ± 0.043	**0.2128** ± 0.050	3.8264 ± 0.264
r^{-2}-2	**0.475** ± 0.169	**0.461** ± 0.167	**0.450** ± 0.142	4.0380 ± 0.557
r^{-1}-3	0.0283 ± 0.003	**0.0261** ± 0.003	0.0408 ± 0.004	3.251 ± 0.101
r^{-1}-2	0.0254 ± 0.005	**0.0238** ± 0.005	0.0326 ± 0.004	2.3365 ± 0.147
Spring-3	**0.0566** ± 0.003	0.0595 ± 0.003	0.0728 ± 0.004	2.6839 ± 0.057
Spring-2	0.0200 ± 0.001	**0.0182** ± 0.001	0.0249 ± 0.001	1.7455 ± 0.066

Table 1: Mean Absolute Error (MAE) loss on the test set, presented as mean ± standard deviation. The best values across different training strategies are shown in bold. Values are highlighted in bold when their confidence intervals overlap with the lowest MAE value for a given simulation.

For comparison, the MAE values for the original paper are presented in Table 2. Unfortunately the original paper does not provide the spread of the MAE values, so it is not possible to compare the standard deviations of the MAE values. However, the original paper also follows the general trend of the standard, bottleneck and L1 performing similarly, with the KL strategy performing significantly worse. However, as a whole, the MAE values in this report are lower than those in the original paper, this is likely due to the preprocessing steps taken to improve data quality that were not present in the original paper.

Sim	Standard	Bottleneck	L_1	KL
Charge-3	1.2	0.99	**0.94**	4.2
Charge-2	**49**	50	52	60
r^{-2}-3	4.0	3.6	**3.4**	9.8
r^{-2}-2	1.6	1.6	**1.2**	9.3
r^{-1}-3	0.051	**0.050**	0.055	3.5
r^{-1}-2	0.077	**0.069**	0.079	3.5
Spring-3	0.11	0.11	**0.090**	3.8
Spring-2	0.047	0.046	**0.045**	1.7

Table 2: Mean Absolute Error (MAE) loss on the test set **in the original paper**. The best values across different training strategies are shown in bold

4.2 Edge Message R^2

The next set of results presented are the R^2 values between significant edge message components and the linearly transformed true pairwise forces (or accelerations), averaged across all components. The results are presented in Table 3. For comparison, the R^2 values for the original paper are presented in Table 4. Example scatter plots of the edge message components against the linear transformation of the pairwise force for the different training strategies of the spring 2D simulation are shown in Figure 7. The standard strategy in this report is found to have a significantly different value than that in the original paper which finds that in all simulations, it

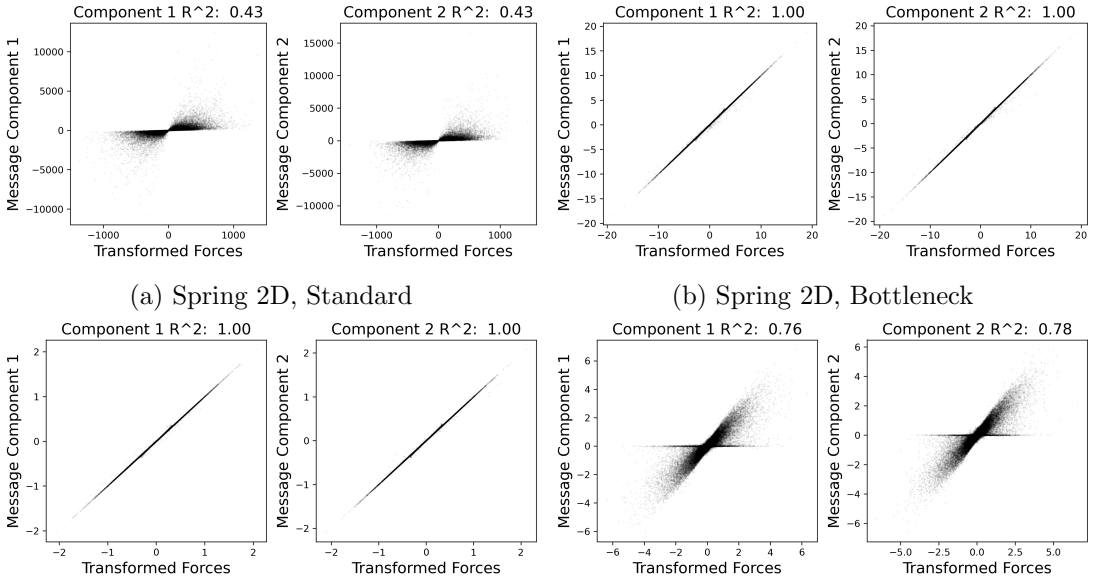
is very close to 0. It is not clear from the original papers results if these values were calculated strictly between the significant edge message components and the pairwise force, or if they also considered the pairwise accelerations. This discrepancy is looked at in more detail in the discussion section.

Sim	Standard	Bottleneck	L_1	KL
Charge-3	0.429 **(0.990)**	0.817 (0.788)	0.683 (0.897)	0.251 (0.472)
Charge-2	0.006 (0.050)	0.700 **(0.897)**	0.006 (0.055)	0.381 (0.431)
r^{-2}-3	0.502 (0.955)	0.518 **(0.998)**	0.505 (0.979)	0.323 (0.564)
r^{-2}-2	0.001 (0.397)	0.411 **(0.985)**	0.181 (0.528)	0.291 (0.426)
r^{-1}-3	0.402 **(0.999)**	0.403 **(0.999)**	0.364 (0.904)	0.333 (0.859)
r^{-1}-2	0.314 (0.962)	0.324 **(0.994)**	0.324 **(0.994)**	0.250 (0.835)
Spring-3	0.430 **(1.000)**	0.430 **(1.000)**	0.990 (0.438)	0.918 (0.532)
Spring-2	0.422 (0.999)	0.999 (0.431)	**1.000** (0.430)	0.768 (0.640)

Table 3: R^2 metric between the significant edge message components and a linear transformation of the true pairwise force (and pairwise acceleration shown in brackets). Best value across training strategy shown in bold (for either acceleration or force).

Sim	Standard	Bottleneck	L_1	KL
Charge-3	0.013	**0.980**	0.002	0.425
Charge-2	0.016	**0.947**	0.004	0.185
r^{-2}-3	0.002	**0.994**	0.977	0.214
r^{-2}-2	0.004	0.993	**0.990**	0.770
r^{-1}-3	0.000	**1.000**	**1.000**	0.332
r^{-1}-2	0.000	**1.000**	**1.000**	0.796
Spring-3	0.036	0.995	**1.000**	0.214
Spring-2	0.032	**1.000**	**1.000**	0.883

Table 4: R^2 metric between the significant edge message components and a linear transformation of the true pairwise force **in the original paper**. Best value across training strategy shown in bold.



(a) Spring 2D, Standard

(b) Spring 2D, Bottleneck

(c) Spring 2D, L_1

(d) Spring 2D, KL

Figure 6: Scatter plots of the edge message components against the linear transformation of the pairwise force for the different strategies of the spring 2D simulation.

4.3 Edge Message Sparsity

Next, to investigate the sparsity of the edge messages under the different training strategies, the percentage of total edge message variance across all components explained by the top d components, where d is the dimensionality of the problem, is calculated. The results are presented in Table 5. The KL and L_1 strategies consistently outperform the standard strategy. This suggests that the KL and L_1 strategies are more effective than the baseline at promoting sparsity in the edge messages. Note, the bottleneck strategy will always have, by definition, 100% of the edge message variance explained by the top d components.

Sim	Standard	Bottleneck	L_1	KL
Charge-3	0.404	1.000	0.474	0.550
Charge-2	0.180	1.000	0.263	0.667
r^{-2}-3	0.583	1.000	0.795	0.925
r^{-2}-2	0.374	1.000	0.525	0.712
r^{-1}-3	0.877	1.000	0.743	0.719
r^{-1}-2	0.591	1.000	0.925	0.990
Spring-3	0.889	1.000	0.999	0.993
Spring-2	0.737	1.000	1.000	0.989

Table 5: Fraction of total edge message variance across all components explained by the top d components, where d is the dimensionality of the problem.

4.4 Symbolic Reconstructions

The next set of results relate to the symbolic distillation of the edge and node models. The edge and node models are distilled seperately using the method outlined. The edge model's distillation should yield an expression which is approximately analogous to a linear transformation of the true force, whereas the node model's distillation should yield an expression which resembles Newtons second law.

Sim	Standard	Bottleneck	L_1	KL
Charge-3	**3/3**	**3/3**	0/3	0/3
Charge-2	0/2	**2/2**	1/2	0/2
r^{-2}-3	1/3	**3/3**	**3/3**	0/3
r^{-2}-2	0/2	**2/2**	0/2	1/2
r^{-1}-3	**3/3**	**3/3**	2/3	**3/3**
r^{-1}-2	0/2	**2/2**	**2/2**	**2/2**
Spring-3	0/3	2/3	2/3	0/3
Spring-2	1/2	**2/2**	**2/2**	0/2

Table 6: Successful reconstructions of the pairwise force (or acceleration) after symbolically regressing the edge model. Experiments where all components are successfully reconstructed are highlighted in bold.

From the results obtained, it is clear that the bottleneck strategy consistently outperforms the other strategies in the symbolic distillation of the edge model, the L_1 strategy is the next most successful, followed by the standard strategy and the KL strategy. The success of the standard strategy is suprising as in the original paper, it was not found to give any successful reconstructions.

Example successful reconstructions of recovering the pairwise force (or acceleration) after symbolically regressing the edge model.

- Spring 2d, L_1 (expect $\phi_i^e \approx \mathbf{a} \cdot (1 - \frac{1}{r})\mathbf{\Delta r} + b$):

$$\phi_1^e = (-0.291 + \frac{0.287}{r})(\Delta x - 0.986\Delta y) + 0.00118$$

- r^{-1} 2d, KL (expect $\phi_i^e \approx \mathbf{a} \cdot \frac{m_1 m_2}{r^2}\mathbf{\Delta r} + b$):

$$\phi_2^e = -\frac{0.273 m_2(\Delta x + \Delta y)}{r^2}$$

- r^{-2} 3d, bottleneck (expect $\phi_i^e \approx \mathbf{a} \cdot \frac{m_1 m_2}{r^3}\mathbf{\Delta r} + b$):

$$\phi_3^e = \frac{6.06 m_2(-0.713\Delta x + \Delta y + 2.44\Delta z)}{r^3}$$

- Charge 3d, standard (expect $\phi_i^e \approx \mathbf{a} \cdot \frac{q_1 q_2}{r^3}\mathbf{\Delta r} + b$. Note, because $q_1, q_2 \in \{-1, 1\}$, $q_1 \times q_2 \equiv \frac{q_1}{q_2} \equiv \frac{q_2}{q_1}$):

$$\phi_1^e == -\frac{39.3 q_2(0.273\Delta x + \Delta y)}{q_1 m_1 r^3}$$

Here, $\mathbf{a} = (a_1, a_2)$ and b are constants, $\mathbf{\Delta r} = (\Delta x, \Delta y, \Delta z)$ represents the displacement vector, q_1, q_2 are the charges and m_1, m_2 are the masses. ϕ_i^e is the ith most significant component of the edge message, reflecting the interaction from body 2 to body 1. Note how the reconstruction for the shown spring 2d experiment is a linear transformation of the true pairwise force whereas the other expressions have divided through by the mass of the receiving particle, m_1, and thus represent the acceleration.

The results of the symbolic distillation of the node model are presented in Table 7. Here, the standard strategy yields the best results, followed by bottleneck, L_1 and finally, the KL strategy.

Sim	Standard	Bottleneck	L_1	KL
Charge-3	**3/3**	0/3	0/3	**3/3**
Charge-2	0/2	0/2	0/2	0/2
r^{-2}-3	**3/3**	**3/3**	**3/3**	0/3
r^{-2}-2	0/2	**2/2**	0/2	0/2
r^{-1}-3	**3/3**	**3/3**	2/3	0/3
r^{-1}-2	**2/2**	**2/2**	**2/2**	0/2
Spring-3	**3/3**	**3/3**	**3/3**	0/3
Spring-2	**2/2**	**2/2**	**2/2**	0/2

Table 7: Successful reconstructions of Newton's 2nd law after symbolically regressing the node model. Experiments where all components are successfully reconstructed are highlighted in bold.

Example successful reconstructions of recovering Newtons 2nd law after symbolically regressing the node model (expect $\phi_i^v \approx \frac{\mathbf{a}\cdot\bar{\mathbf{e}}+b}{m_1}$, where $\bar{\mathbf{e}}$ is the aggregated edge message and m_1 is the mass of the receiving particle).

- Spring 2d, L_1:
$$\phi_1^v = \frac{0.199\bar{e_1} - 0.509\bar{e_2} + 0.0316}{m_1}$$
- r^{-1} 3d, standard:
$$\phi_2^v = -0.00317\bar{e_3} - 0.00379\bar{e_1} - 0.00253\bar{e_2} + 0.00396$$
- r^{-2} 2d, bottleneck:
$$\phi_1^v = 7.76\bar{e_1} - \bar{e_2}$$
- Charge 3d, KL:
$$\phi_2^v = \frac{0.147(\bar{e_1} + \bar{e_2})}{m_1}$$

In some cases, the node model is a simple linear transform of the aggregated edge messages, this occurs when the edge model learns the pairwise acceleration, so the node model just acts as a sum over the individual accelerations to return the net acceleration. In other cases, like the charge 3d KL experiment and the spring 2d L_1 experiment, the node model explicitly learns the division by the mass of the receiving particle, m_1, to convert the force to an acceleration.

Finally, the performance of the symbolic models are compared to the neural network models in Table 8. This is done choosing experiments where all components of the edge and node model outputs could be successfully reconstructed and comparing the MAE and standard deviation on the test set from the symbolic model against the best performing neural network for that simulation. The neural network generally outperforms the symbolic model. It is worth noting that the constants in the symbolic model were not refit to the data as suggested in the original paper.

Sim	Symbolic	Neural Net
Charge-3	2.775 ± 0.305	**0.110** ± 0.015
r^{-2}-3	3.225 ± 1.084	**0.518** ± 0.998
r^{-2}-2	1.956 ± 1.110	**0.450** ± 0.142
r^{-1}-3	0.161 ± 0.027	**0.0261** ± 0.003
r^{-1}-2	0.0290 ± 0.006	**0.0238** ± 0.005
Spring-2	0.0234 ± 0.002	**0.0182** ± 0.001

Table 8: Symbolic model versus neural network, performance compared by MAE on the test set, presented as mean ± standard deviation. The best values across different models are shown in bold.

5 Discussion

In this section the results are discussed, both in a standalone manner and in the context of the original paper's results.

5.1 Model Performance

To understand the results, it is first imperative to understand that the force laws simulated in this report vary in complexity. This complexity can be understood in terms of dependencies on the distance r. Specifically, the spring force is linearly proportional to r whereas the orbital force laws follow r^{-1} and r^{-2} dependencies. The electrostatic force also follows a r^{-2} dependency. This leads to significant differences in how data varies across simulations, as detailed by the standard deviation of the acceleration labels in the training set (Table 9).

Sim	Standard Deviation
Charge-3	14
Charge-2	75
r^{-2}-3	31
r^{-2}-2	150
r^{-1}-3	5.3
r^{-1}-2	7.6
Spring-3	15
Spring-2	6.8

Table 9: Standard deviation of the acceleration labels in the training set for the different simulations.

Table 10 ranks the networks by the standard deviation of the accelerations in their training data and their MAE on the test set. The results show how the standard deviation of the accelerations in the training set can predict the performance of the model on the test set. Which makes intuitive sense, it is harder for the model to learn robust features if the data varies over too large a scale. Further, it explains why the performance on the 2D data for a given force law is worse than the 3D data, in all cases except spring where the situation is reversed due to it being proportional to a positive power of r. Due to the higher dimensionality of the problem, there is more 'space' so there is a smaller chance any pair of particles get too close to each other and the acceleration blows up. This is why the models trained on 3D data perform better than those trained on 2D data. For the spring experiment, the larger distances between particles in 3D means that the accelerations will be larger and hence why for spring, the 2D simulation performs better. It is worth noting that the MAE values presented in the original report, shown in Table 2 are also consistent with this point. The main point of contention is how the original

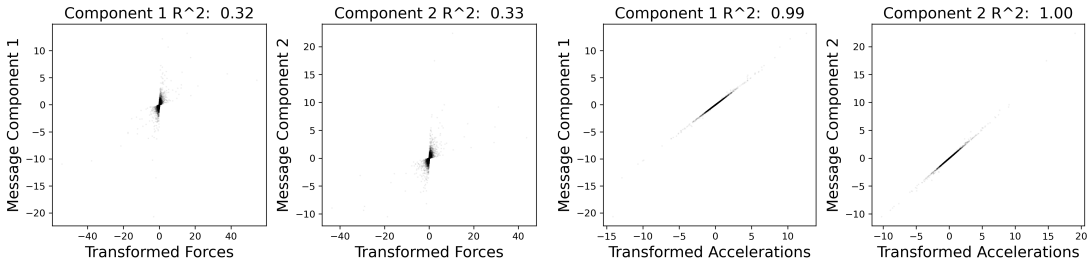
paper frames the results as the L_1 strategy being the best when both the reproduction and the original results show how the results are very close. Indeed, in this report the bottleneck strategy was found to be the best in five out of eight simulations by a slim margin, however factoring in the standard deviations in the MAE scores, one can see how the strategies are very close in performance. As such, there is no clear dominant strategy, only a clear worst one in the KL strategy.

Performance	Training Label Std. Dev.	Test Set MAE
1 (**Best**)	spring-2	spring-2
2	r^{-1}-3	r^{-1}-2
3	r^{-1}-2	r^{-1}-3
4	spring-3	spring-3
5	charge-3	charge-3
6	r^{-2}-3	r^{-2}-3
7	charge-2	r^{-2}-2
8 (**Worst**)	r^{-2}-2	charge-2

Table 10: Comparing the order of experiments ranked by the standard deviation of the acceleration labels in the training set and the best mean absolute error (MAE) on the test set.

Further, the original papers MAE's are consistently higher, despite training for twice as long on twice as much data. At its extreme, the difference between the MAE presented in this report versus the originals was 49, compared to 0.462 for the standard charge 2d model. This likely means that the original data contained more extreme values, otherwise such a result is unexplainable. This motivates the importance of the pruning preprocessing step, which was not used in the original paper, however was essential in this reproduction. Without pruning the data, the KL training strategy was not found to be stable, with it's gradients exploding shortly into training, giving rise to nans in the loss. It is worth noting that the pruning is not the only solution to handling the large accelerations. These outliers were only an issue for the KL training strategy, where the predicted log variance terms would blow up after exponentiating them to compute the loss. One solution would be to squash the log variance term, alternatively the epsilon term added to the distance could be increased when generating the data. It would be interesting to see if the recovered symbolic expressions would then learn the larger epsilon term.

5.2 Edge Message R^2



(a) (b)

Figure 7: Comparison of scatter plots of the edge message components for r^{-1} 2D, L1 experiment against a linear transformation of the a) pairwise force b) pairwise acceleration.

The next set of results to discuss are those presented in Table 3. These show the R^2 values of the edge message components against the transformation of the true force or acceleration. It is important to highlight that the edge model may sometimes learn pairwise acceleration directly, rather than pairwise force. This was also evident in the original paper, where the $1/r^2$ bottleneck experiment was shown to give the pairwise acceleration, however, the implications of this were not explored. Table 3 and Figure 7 explicitly demonstrate that R^2 values can be misleading if both pairwise force and acceleration are not separately considered. For clarity, two R^2 values are reported: one for pairwise force and one for pairwise acceleration. Generally, when one value is low, the other tends to be high, suggesting that the model captures relevant physical interactions rather than failing to represent the data appropriately. Contrary to the original paper, which suggests that the standard strategy yields almost zero R^2 values in all cases, findings in this report do not support this claim. The observed differences likely stem from variations in the datasets used. To further investigate this, multiple datasets could be generated using different seeds, and the analysis could be repeated.

5.3 Edge Message Sparsity

The next set of results to be discussed are those presented in the Table 5, showing the fraction of the total variance captured by the d most significant components in the edge message, where d is the dimensionality of the problem. The general argument presented in the original paper was that promoting sparsity in the edge message would lead to a greater probability of successful symbolic reconstructions. However, the findings for KL training strategy provide an interesting example where the edge messages were often found to be sparser than those coming from networks trained under the L_1 strategy, however it was the strategy which had the lowest number of successful symbolic reconstructions. It is possible that the inductive bias placed on the model by the KL strategy was not appropriate for the task at hand. Alternatively, it may be the case that weight of the regularisation term was simply too large. The high MAE values over the test set also indicate that the model may have over prioritised sparsity in the edge message. Further, it is interesting to note that the opposite case is also true, where a lack of sparsity in the edge messages does not necessarily mean that the symbolic regression will fail. The charge 3D standard experiment is a good example of this, where the edge messages were not particularly sparse, with the top 3 components only capturing 40% of the total variance, however all components were successfully reconstructed.

5.4 Symbolic Reconstructions

Finally, the symbolic reconstructions are discussed. First the edge model's reconstructions are considered. It is no suprise that the bottleneck strategy is the most successful, as it places the strongest inductive bias, explicitly constraining the message latent space to be of the correct dimensionality. However, what is suprising is that the standard model performed better than in the original paper, which found that no model trained under the standard strategy could yield a successful reconstruction. Here, it was found that the standard strategy was able to give some successful reconstructions, despite not having any explicit strategy to promote sparsity in the edge messages. Which suggests that the inductive biases of a message-passing neural network, which calculates acceleration from aggregated messages, might be better than originally presented. This conclusion is supported by the R^2 values observed, which were comparable to those obtained under strategies explicitly encouraging sparsity.

To frame this discussion, it is important to understand the relevance of the R^2 Table 3. Since the true underlying force law is known and can be expressed perfectly in terms of summed messages, it is known that the symbolic model for the edge message will be a linear transformation of the true force. As such an R^2 value close to 1 is a sufficient condition for one to conclude that the model can be expressed as a symbolic equation. Therefore, the R^2 values can be used as

a proxy to determine which training strategies were successful, which show that the standard, bottleneck and L1 strategies should all have more successful reconstructions than have been reported. The issue with retrieving a symbolic model is that the search space is vast, despite using a subset of the operators considered in the original paper to reduce the search space. For example, the spring 3d bottleneck provides a good example of this point. Since it had an R^2 value close to 1, it was assumed that the model had learned an appropriate representation of the pairwise force. However, the symbolic search was unable to find any successful reconstructions in the first attempt. It took several more attempts before a search yielded components that were close to the true pairwise force. Other experiments such as the spring 2D, spring 3D and r^-1 2D and standard experiments all have R^2 values close to 1, however the single search performed was unable to find a successful reconstruction. It is very likely that if a longer search was done, as was the case for the spring 3D bottleneck experiment, a successful reconstruction would have been found. Therefore, the reconstructions presented should not be used as an absolute measure of the success of the training strategy, but rather an indication of what is possible to recover with a quick search. It is also important to note that while it is not strictly necessary that the experiment have a high R^2 value in order for symbolic regression to be successful, it is also the case that one would expect the R^2 value to indicate moderate correlation if the symbolic regression was successful. However, the original paper presents two odd cases, where for the KL strategy, the spring and r^{-1} 3D experiments had very low R^2 values of was 0.214 and 0.332, respectively, yet both were stated to give successful reconstructions. Such a result is quite rare and would be interesting to investigate further.

Next the node model results are discussed. The results show that the standard, bottleneck and L1 and standard strategies all perform well. It is interesting to note how much better the node reconstructions were for the standard strategy compared to the edge reconstructions. This is likely because of the fact that neural networks like to learn simple identity functions where possible. In this case, the architecture of the GNN allows for the node model to act as a simple sum function which may explain the higher success rate of the node reconstructions compared to the edge reconstructions. Future work could include creating R^2 tables for the node model to better understand if reconstructions were missed as discussed earlier or if the experiments were truly unsuccessful. This could be done by finding the correlation between the node model outputs and a linear transform of the aggregated edge message and node features in the same way as was done for the edge model.

It is worth noting that sometimes the correct reconstruction was not the best, however were excluded, to strictly adhere to the selection criteria outlined. In practice this would be relaxed. Further, some reconstructions were very close to the expected result, examples of close reconstructions are given in the appendix.

The final set of results to discuss are those from comparing the symbolic model's performance to the trained networks. The results show that the symbolic model generally performs worse than the corresponding neural networks. This is likely to be because the constants in the symbolic model were not refit to the data as suggested in the original paper. Given that there is no noise in the dataset, one would expect the symbolic model to perform at least as well as the neural network, if not better. It is likely performing worse due to the accumulated approximation error from the multiple levels of function fitting. It would be interesting to see how the symbolic model would perform if the constants were refit to the data.

5.5 Future work

There are two primary means by which this reproduction study could be further improved. The first being to repeat the analysis for different random seeds and seeing if the different sets of trained models are consistent with the results presented in this report. This would help give an understanding of what is natural variation in the results due to the stochastic nature of the training process versus what is a systematic discrepancy between the reproduced study and the

original findings. Such an approach would solidify the conclusions drawn and provide a clearer comparison between the varying strategies employed for both the edge and node models.

The second means by which this reproduction study could be further improved is to investigate the effect incorporating more inductive biases and seeing if the trained models are easier to distill as would be expected. A natural bias to add to the model would be to include Newtons third law, through introducing a penalty to the loss term that enforces the sum of $\phi^{\mathbf{e}}(\mathbf{x_i}, \mathbf{x_j}) + \phi^{\mathbf{e}}(\mathbf{x_j}, \mathbf{x_i}) = \mathbf{0}$.

6 Conclusion

This report has presented the results of a reproduction study of the paper "Discovering Symbolic Models using Deep Learning with Inductive Biases". Here a subset of the experiments introduced in the original paper were investigated. This report managed to find the same overall trends in the MAE values as the original paper. Symbolic reconstructions of the edge and node models were successfully found. With the bottleneck and L1 training strategies being the most successful in yielding the most reconstructions of the true force law as reported by the original paper. However, the standard strategy was found to be far more successful in learning a compact latent space than was suggested by the original paper. Thus, the main claim of the original paper, that symbolic regression can be used to distill the learned models of neural networks trained using appropriate inductive biases, was found to be true.

7 Appendix

7.1 Failed Symbolic Reconstructions

Example of some failed edge reconstructions:

- Spring 2D, KL (clearly wrong):

$$\phi_2^e = \frac{\Delta x + \Delta y}{-r + m_1(0.647 - m_1)}$$

- Charge 2D, L_1 (almost correct):

$$\phi_2^e = \frac{0.351 \Delta y q_2}{r^2(r + 0.169 m_1)}$$

- r^{-2} 3D, standard (almost correct):

$$\phi_1^e = \frac{8.43 m_2 (0.26 + \frac{\Delta x - \Delta y + 3.52 \Delta z}{r})}{r^2}$$

Example of some failed node reconstructions:

- Charge 3D, KL (clearly wrong):

$$\phi_3^v = \bar{e_1} \bar{e_2}^2$$

- Charge 3D, bottleneck (almost correct):

$$\phi_1^v = (0.153 + \frac{0.171}{m_1})(\bar{e_2} + \bar{e_1} + 1.11)$$

- r^{-2} 2D, standard (clearly wrong):

$$\phi_1^v = 3.62 \times 10^{-5} \bar{e_1}^2$$

7.2 CoPilot and ChatGPT Usage

Copilot was used extensively for formatting the equations and plots used in this report and writing the doc strings for the code. The script `src/view_symbolic_eq.py` was generated by chatGPT.

References

[1] Miles Cranmer. Interpretable machine learning for science with pysr and symbolicregression.jl, 2023.

[2] 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, 2020.

[3] Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates, 2018.