[Re] Discovering Symbolic Models from Deep Learning with Inductive Biases

Final Project

University of Cambridge

Department of Physics

Prepared by:
Sabahattin Mert Daloglu

Supervised by:
Miles Cranmer

Word Count: 6813

June 30, 2024

Contents

1 Abstract

This paper[1] is a reproduction and analysis of the novel work presented in **Discovering Symbolic Models from Deep Learning with Inductive Biases** by Cranmer et al [3]. The study employs both explicit (architecture bottlenecks) and implicit (regularization) inductive biases to create sparse latent representations in the edge model of Graph Networks (GNs). These GNs are trained end-to-end on graph structured simulated data to minimize the difference between the node model outputs and instantaneous acceleration of particles in physical systems. Subsequently, symbolic regression is applied to the sparse message vectors to uncover the physical relationships between particles. The original paper successfully recoveres the correct force law equations for Newtonian, spring, and charge systems in both 2D and 3D. This paper reproduces the results for different regularization methods—L1, Kullback-Leibler, and Bottleneck— trained on 2D classical mechanical systems and compares their effectiveness at predicting the true accelerations and recovering known physical equations through symbolic regression. Additionally, this study explores two extensions involving different schedulers for the L1 regularization coefficient: a linear scheduler and a triangular scheduler. The linear scheduler progressively increases the coefficient throughout the training, aiming to balance the expressiveness of the latent space in earlier epochs with the enforcement of sparsity in the learned message vectors in later epochs. Conversely, the triangular scheduler increases the coefficient until the midpoint of training, then decreases it linearly until the end. This method is designed to periodically vary the regularization pressure, potentially enhancing the convergence to the true dimensionality of the physical system. These schedulers aim to optimize the balance between expressiveness early in training and the desired sparsity in the learned message vectors, facilitating convergence to the true physical dimensions. The implementation of these schedulers has shown to improve sparsity and overall L1 model performance. Additionally, a preprocessing step of data pruning was implemented to remove graphs with outlier acceleration values, which was crucial for maintaining the stability and accuracy of the training process as well as the improved performance observed in models trained with charge system.

2 Introduction

Deep learning models excel at classifying and predicting data in high-dimensional spaces but are often criticized for their opacity, earning them the label of "black-box" models due to their lack of interpretability. In contrast, symbolic closed-form expressions provide clear and direct relationships between input variables and outputs, enhancing interpretability. The work reproduced in this paper explores the fusion of these two methodologies to enhance the interpretability of deep learning models, thereby uncovering the underlying physical intuition behind GN architectures. Symbolic regression, which has gained popularity in scientific research following the introduction of Eureqa software[9], is employed to derive analytical functions from extensive experimental datasets. Despite its benefits for interpretability and expressiveness, symbolic regression's reliance on exhaustive search across the solution space is a significant limitation. To address this, evolutionary algorithms like genetic algorithms are utilized to iteratively explore the solution space, selecting optimal solutions based on a fitness function and employing genetic operations such as reproduction, crossover, and mutation, akin to natural selection [7]. However, the convergence to a desired solution can be slow, escalating exponentially with the complexity and dimensionality of the input and operator space. GNs mitigate this issue with their inherently separable structure (node, edge, and global), which aligns well with the demands of Newtonian physics problems. This modularity permits the application of symbolic regression to discrete sub-components of the model, thereby reducing the number of iterations needed to search the solution space. The symbolic expressions derived from these segments can subsequently be composed to form a comprehensive model expression learned by the entire GN. These synthesized symbolic equations are then evaluated against the GN models to assess their generalizability to new, unseen data, thereby testing the principle of Occam's Razor, which posits that science favors simpler explanations when other factors are equal [6].

[1]This paper follows the structure of the reproduction papers submitted to the ML Reproducibility Challenge 2023.

3 Theory and Framework

A natural starting point of discussion is a brief overview of the message-passing GNs used in this study. The terminology and structure of graphs are adopted from [1], which serves as a foundational reference throughout this paper. A graph in this context is composed of a set of nodes $V = \{\mathbf{v}_i\}_{i=1:N^v}$, where N^v is the total number of nodes, and a set of edges $E = \{\mathbf{e}_k, r_k, s_k\}_{k=1:N^e}$, where N^e represents the total number of edges. Each $\mathbf{v}_i$ is a node feature vector of dimension L^v. Similarly, each $\mathbf{e}_k$ is an edge feature vector with dimensions L^e, linking the sender node index s_k to the receiver node index r_k. This graph structure, which is the input for the GN model, is depicted in Figure 1. The fully connected nature of the edges ensures the model's equivariance to permutations of particles, an essential property for analyzing particle systems with inherent symmetrical characteristics.

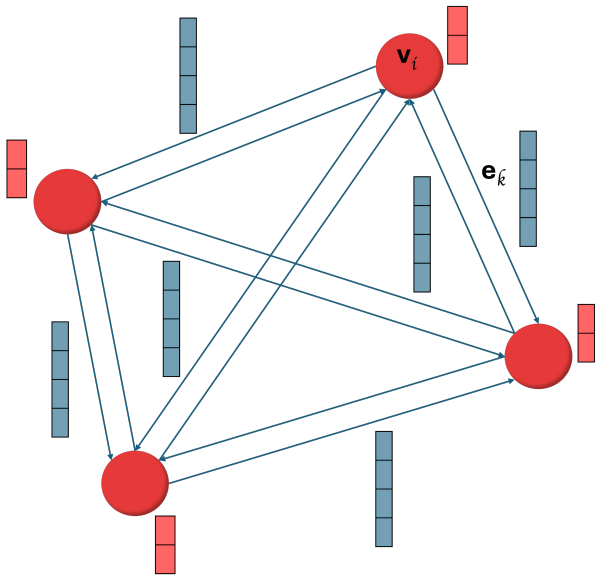

Figure 1: Graph structure that is used to represent a snapshot of a simulation data input to the GN model.

GNs consist of three separable components: the edge model ϕ^e, the node model ϕ^v, and the global model ϕ^u as well as the intermediate step of aggregation. These three models are used consecutively to predict the next state of the graph defined by set of V' nodes and E' edges. The models are described below:

- **Global Model** ϕ^u: This model is used to update some arbitrary global metric of the system represented in graph such as the gravitational field of the system under study [1]. This is a graph-level attribute and is generally used for predicting dynamical systems. Since, each time step (snapshot) of graph is treated as independent by considering instantaneous forces and accelerations, this model is not used in the original paper.

- **Edge Model** ϕ^e: A function ϕ^e with mapping $\mathbb{R}^{L^v} \times \mathbb{R}^{L^v} \rightarrow \mathbb{R}^{L^{e'}}$, usually consisting of multilayer perceptions (MLPs), that intakes two node feature vectors that are connected by an edge and outputs an edge message vector. The updated edge vector is found by $\mathbf{e}'_k = \phi^e(\mathbf{e}_k) = \phi^e(\mathbf{v}_r, \mathbf{v}_s)$. This message contains information about the interaction of the nodes (particles in this study) and is expected to correspond to pairwise force between connected nodes/particles in the system.

- **Aggregation**: This is an intermediate updating step which does not consist of MLPs but only a summation of the messages. This corresponds to the net force calculation in Newtonian systems. For a given receiving node, all the edges it is connected to have a corresponding message after the edge model is applied. These messages are summed to give the aggregated message of the receiving node: $\bar{\mathbf{e}}'_i = \sum_{k \in \{1:N^e | r_k = i\}} \mathbf{e}'_k$ where $\bar{\mathbf{e}}'_i$ is the aggregated edge vector of the receiving node r_i.

- **Node Model** ϕ^v: A function ϕ^v with mapping $\mathbb{R}^{L^v} \times \mathbb{R}^{L^{e'}} \rightarrow \mathbb{R}^{L^{v'}}$, usually consisting of multilayer perceptrons (MLPs), that intakes the aggregated message corresponding to a receiving node and its node features. The output of the node model is the prediction of the instantaneous acceleration of the receiving node/particle. The GN model used in this study is a node-focused GN [1], which means the output of the node model is used in loss calculation by comparing against the ground truth acceleration values. This part of the model corresponds to the Newton's second law of motion where the acceleration on a body is computed via dividing the net-force (aggregated messages) by the mass of the body (node feature).

The separable structure of the message-passing GN is illustrated in Figure 2, and the correspondence of the edge-node models to the force-acceleration calculations of the Newtonian systems under study is the cornerstone of the inductive biases introduced in the paper. Crucially, the MLPs employed in both edge and node models feature ReLU activations, consist of two hidden layers, and contain 300 hidden nodes each.

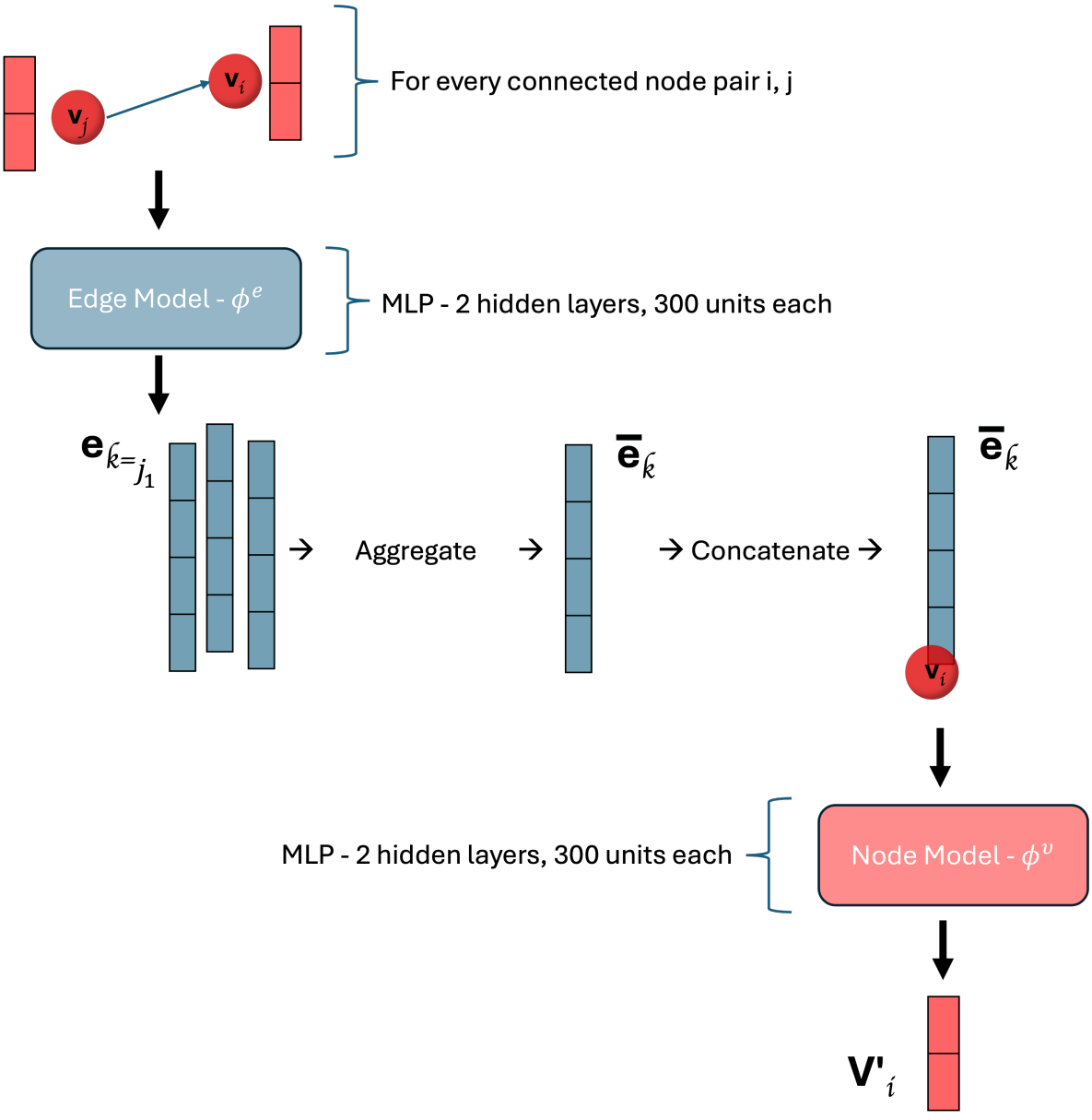

Figure 2: Message passing Graph Network architecture utilized in the study.

The next important inductive bias is the regularization of the message vectors to implicitly encour-

age sparsity in the learned edge vectors. This is achieved by incorporating either L1 regularization or a KL-divergence term into the loss function. It is important to recognize that through this regularization, we are aiming to bias the message vectors to converge to the true number of dimensions of the forces in the system, without explicitly providing the true forces or the number of dimensions. Consequently, the sparsely represented messages are learned as a linear transformation of the true forces in an unsupervised manner, as the training process is based on minimizing the acceleration prediction error. A rough mathematical explanation is given in the main paper, here we sketch a more detailed proof why this is the case. To mathematically prove that the learned message vectors are linear transformations of the true forces, consider the following:

1. **Definitions and Assumptions**:

   - Let $\mathbf{F}_{\text{true}}$ represent the true force vector in the system.
   - The edge model ϕ^e learns a message vector $\mathbf{e}'_k$ from node features $\mathbf{v}_r$ and $\mathbf{v}_s$ connected by an edge k.
   - The regularization term (either L_1 or KL-divergence) in the loss function minimizes the dimensionality of $\mathbf{e}'_k$, promoting sparsity.
   - The training minimizes the loss function

     $$\mathcal{L} = \|\mathbf{a}_{\text{pred}} - \mathbf{a}_{\text{true}}\|^2 + \lambda\|\mathbf{e}'_k\|,$$

     where $\mathbf{a}_{\text{pred}}$ is the predicted acceleration, $\mathbf{a}_{\text{true}}$ is the true acceleration, and λ is the regularization coefficient.
   - According to Newton's second law,

     $$\mathbf{a}_{\text{true}} = \frac{\mathbf{F}_{\text{true}}}{m},$$

     where m is the mass of the particle.
   - The model learns to predict $\mathbf{a}_{\text{pred}}$ such that

     $$\mathbf{a}_{\text{pred}} \approx \mathbf{a}_{\text{true}}.$$

2. **Proof of Linear Transformation**:

   - Let $\mathbf{f}_i$ be the net force on the i-th body, calculated as the sum of individual forces $\mathbf{f}_k$ acting on it:

     $$\mathbf{f}_i = \sum_{r_k=i} \mathbf{f}_k.$$

   - Assume the GN predicts accelerations perfectly. Then, if we ignore the mass of the target body, the predicted net force for body i can be approximated as:

     $$\hat{\mathbf{f}}_i \approx \mathbf{a}_i = \sum_{r_k=i} \mathbf{f}_k = \phi^v(\mathbf{v}_i, \bar{\mathbf{e}}_i),$$

     This is true for any number of bodies. For a two-body system with a single interaction, the predicted net force can be approximated as:

     $$\hat{\mathbf{f}}_i = \mathbf{f}_{k,r_k=i} = \phi^v(\mathbf{v}_i, \mathbf{e'}_{k,r_k=i}),$$

     The two-body system can be summed over the number of bodies to get the multi-body system's expression:

     $$\sum_{r_k=i} \mathbf{f}_k = \sum_{r_k=i} \phi^v(\mathbf{v}_i, \mathbf{e'}_k) = \phi^v(\mathbf{v}_i, \sum_{r_k=i} \mathbf{e'}_k).$$

     From this, we can see that ϕ^v has to be a linear operation. To find how this relates to the message vectors, we take the inverse of ϕ^v:

     $$\mathbf{e'}_k = (\phi^v(\mathbf{v}_i, \cdot))^{-1}(\mathbf{f}_k).$$

Since, ϕ^v is a linear operation, its inverse is also a linear operation. Moreover, ϕ^v is only invertible if the output of ϕ^v and $\mathbf{e'}_k$ have the same number of dimensions. Since, previously we assumed that ϕ^v perfectly predicts the instantaneous accelerations, this is only the case if the number of dimensions of $\mathbf{e'}_k$ is equal to the true dimensions of the system. Therefore, by introducing regularization to the loss function, and sparsing the message vector, we can ensure that ϕ^v is invertible, hence the learned messages $\mathbf{e}'_k$ are linear transformations of the true forces $\mathbf{f}_k$.

3. **Conclusion**: Under the assumption that the model predicts the instantaneous accelerations perfectly, and the regularization is applied to the messages such that they are sparse and correspond to the true number of dimensions, then it is shown that the messages are linear transformations of the true forces:

$$\mathbf{e}'_k = \mathbf{a} \cdot \mathbf{F}_{\text{true}} + \mathbf{b},$$

where $\mathbf{a}$ is a learned weight matrix and $\mathbf{b}$ is a bias vector.

In the Symbolic Regression section, we confirm that this holds true by fitting a symbolic regression on the sparsed message vectors and comparing the resulting equation to the true force equations of the classical mechanical systems. If the recovered equation is the linear transformation of the true force equations as shown for the spring system in Equation 2, then the model is able to learn the forces in the system and the proof above is correct experimentally as well.

$$\mathbf{F}_{\text{spring}} = -2(\Delta x, \Delta y) \cdot (\frac{r-1}{r}) \tag{1}$$

$$\phi^e_1 = -2\mathbf{a} \cdot (\Delta x, \Delta y) \cdot (\frac{r-1}{r}) + b \tag{2}$$

The final inductive bias for this study is the choice of the Adam optimizer which is another implicit regularization in the loss function as discussed by Simons Prince [8]. The choice of optimizer and loss function provides a bias for the learning process and the loss curve.

4 Scope of Reproducibility

In this paper, the following items from the original papers are aimed to be reproduced with main claims highlighted:

- Implementation of the GN architecture with a single message passing step.
- Train the GN using the selected hyperparameters and regularization techniques on the following 2-dimensional systems: spring, charge, r^{-1}, and r^{-2}.
- **Claim 1:** Achieve a sparse latent representation of the learned messages, corresponding to the true number of dimensions.
- Test the performance of L1 regularization, KL-divergence, and Bottleneck models on achieving sparse learned messages.
- Compare the performance of the Bottleneck model with the regularizer models.
- Learn a symbolic model from the sparse representation of the learned messages by applying symbolic regression using the **PySR** library [2] to approximate the message vectors.
- **Claim 2:** Recover the linear transformation of the known physical laws of the spring, charge, and Newtonian systems.
- **Claim 3:** Test the performance of the symbolic models on the test datasets and test the Occam's Razor arguments: if symbolic equations generalize better to unseen data than the GN.

5 Datasets

The 2-dimensional 4-particle classical mechanical systems used for this study are simulated based on the following potentials:

- Gravitational dynamics with $\frac{1}{r^2}$ potential: $U_{12} = -\frac{m_1 m_2}{r'_{12}}$
- Gravitational dynamics with $\frac{1}{r}$ potential: $U_{12} = m_1 m_2 \log(r'_{12})$
- Charge system: $U_{12} = q_1 q_2 / r'_{12}$
- Spring system: $U_{12} = (r'_{12} - 1)^2$

The data is generated using the dynamical system simulations code provided in the public repository of the original paper[2]. The number of simulations per system was set to $10,000$, each lasting over 500 time steps. Each time-step of the dynamical system is treated as an independent snapshot of the system and used to build a fully connected graph, with node embeddings as positions, velocities, mass, and charge values: x, y, v_x, v_y, m, q. The labels for each node are the true instantaneous acceleration values: a_x, a_y, which will be compared to the output of the node model ϕ^v.

The output acceleration features a_x, a_y, the positional features x, y, and the velocity features v_x, v_y are plotted in histograms of bin size 50 shown in Figure 3 for the charge-system. As can be seen from the boxplot on the top right of the figure, there exists very large acceleration values. These values are found to be on the order of 10^5 and were treated as outliers. There are two motivations for this choice. The first one is that these extreme high values were found also in the initial stages of the simulation not just towards the end. Hence, this indicates these high values are likely to be related to the initialization of the simulation as well as the random configuration of particles. Secondly, since we are treating every snapshot of each simulation as an independent graph with instantaneous acceleration values associated to each particle/node, and not as a propagation of a dynamical system, it would not matter to remove graphs with high acceleration values. The model will still learn to predict the instantaneous acceleration of particles in a supervised manner but the effect of outliers on the loss value will be reduced. It should be noted that these large acceleration values can bias the loss and thus affect the learning process of the model. This preprocessing step is the first deviation from the original paper. The lower and upper whisker values shown in the box plot in Figure 3 are calculated as 1.5 times the interquartile range (IQR) away from the first and third quartile respectively. Graphs with acceleration values that fall outside of the whisker limits are removed. The resulting pruned data distributions are shown in Figure 4.

[2] https://github.com/MilesCranmer/symbolic_deep_learning

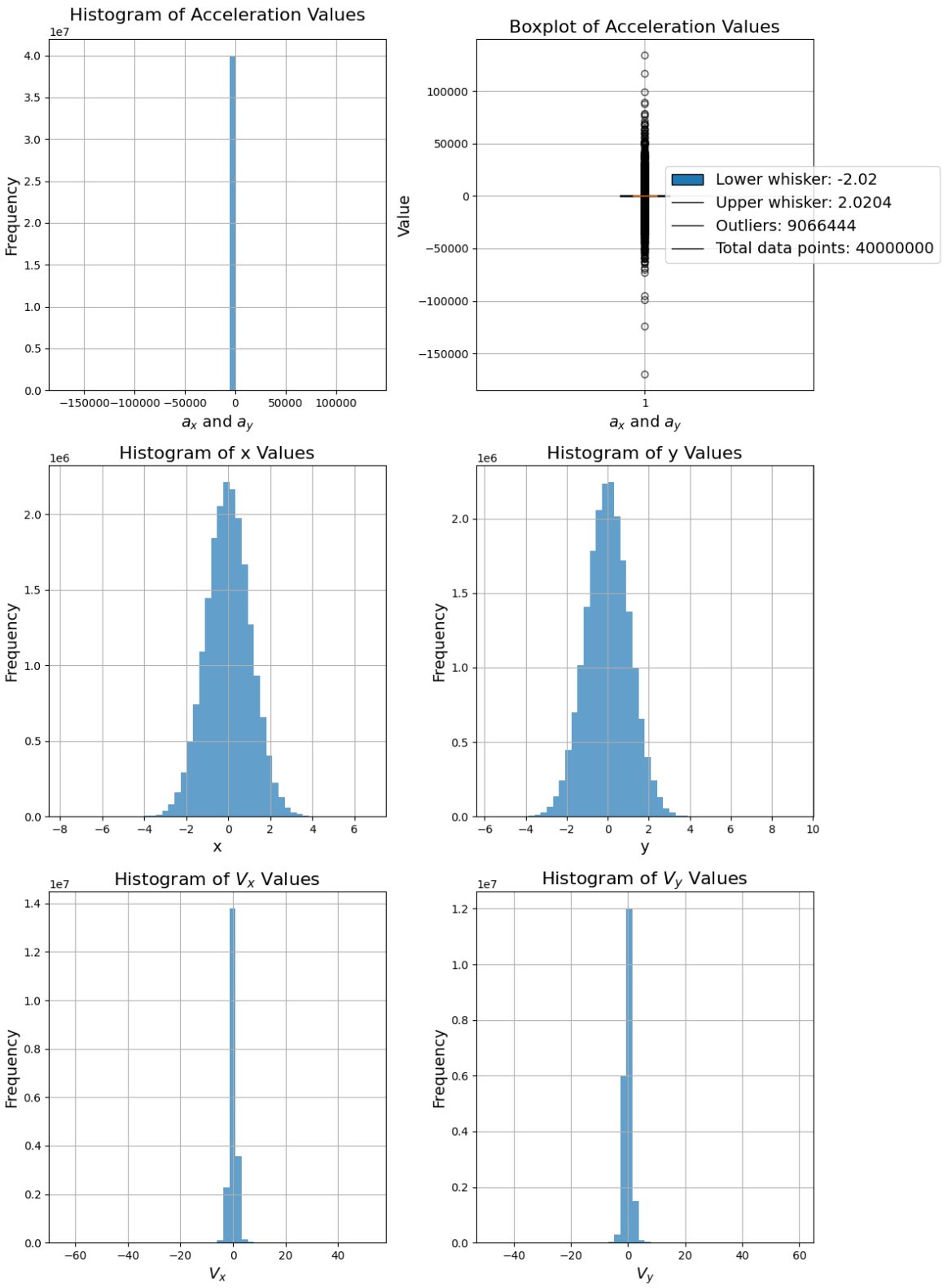

Figure 3: Histograms of positional features x, y, velocity features v_x, v_y, and acceleration features a_x, a_y before pruning for the simulated charge systems. The top right boxplot shows the distribution of acceleration values, highlighting the presence of outliers.

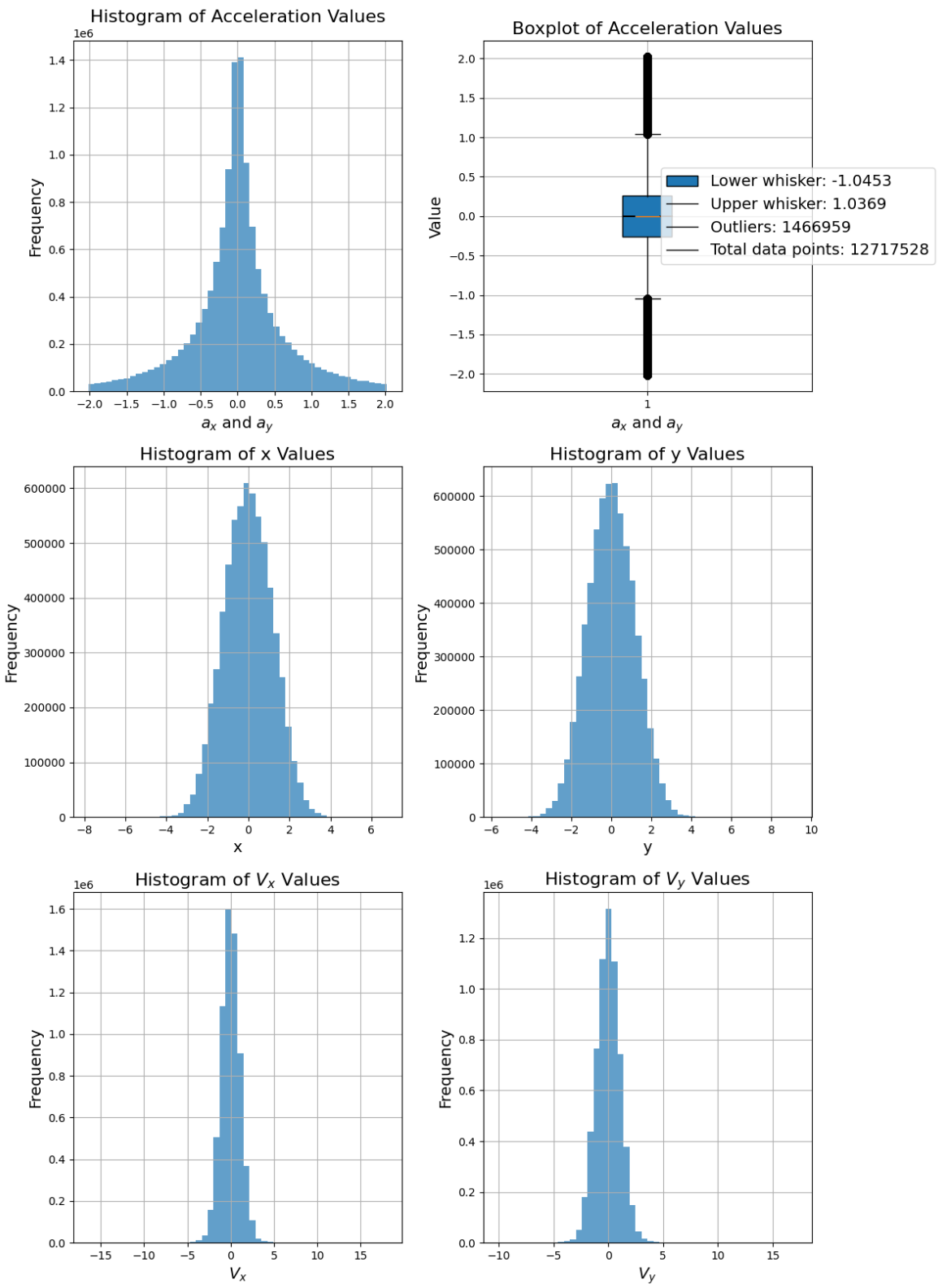

Figure 4: Histograms of positional features x, y, velocity features v_x, v_y, and acceleration features a_x, a_y after pruning for the charge systems. The removal of outliers results in a more representative distribution of the typical system dynamics.

This data pre-processing step is performed for each of the simulated systems' datasets. After the data pruning, only 1,000,000 graphs are selected for each system to ensure data pruning results in equal number of data for all systems. This number of data is later intrinsically increased by data augmentation during the training process to ensure the robustness of the model by introducing randomness in the training dataset.

6 Models

The sparsity of the learned message vectors is introduced by adding a regularization term to the total loss function. Two different regularization methods are used in this study. The first one is the Kullback-Leibler Divergence (KL) and the second one is the L1 regularization. They are both implicity encouraging the model to learn a sparse representation of the learned messages that has the same dimension as the true number of dimension as apposed to the explicit constraint used in the Bottleneck model. These methods are explained below:

6.1 Kullback-Leibler Divergence Model

This model treats the message vectors as a multinomial Gaussian distribution over the latent space and regularizes the model to minimize the KL divergence between the learned distribution and a unit normal distribution. In this case, the edge model will be designed to predict the distribution parameters: mean μ and log-variance $log(\sigma^2)$ of the normal distribution. Hence, the number of message features of the edge model output is set to 200 so that the first half of the outputs predicts the means and the second half predicts the log-variance of the normal distribution. The KL-divergence term for the message regularization in the loss is:

$$\alpha_1 \mathcal{L}_e = \alpha_1 \frac{1}{N_e} \sum_{k=\{1:N_e\}} \sum_{j=\{1:L_e/2\}} \frac{1}{2} \left(\mu_{k,j}^2 + \sigma_{k,j}^2 - \log\left(\sigma_{k,j}^2\right) \right) \tag{3}$$

The regularization strength α_1 is set to be 1.

6.2 L1 Regularization Model

This model uses L1 norm penalty on the values of the message vectors as shown in Equation 4. The the number of message features of the edge model output is set to 100. This is also known as the Lasso regularization and the absolute value encourages some of the *unimportant* message vector components to shrink to zero [4]. Hence, the use of L1 regularization on the message vector components, provides sparse message vectors.

$$\alpha_1 \mathcal{L}_e = \alpha_1 \frac{1}{N_e} \sum_{k \in \{1:N_e\}} |e_k| \tag{4}$$

Initially, the coefficient α_1 is consistently maintained at 0.01 throughout the training process, aligning with the methodology of the original paper. Additionally, two dynamic schedulers are implemented to adjust the L1 regularization coefficient: a linear scheduler and a triangle scheduler. The linear scheduler linearly increases the coefficient from 0.01 to 0.1, while the triangle scheduler starts at 0.01, peaks at 0.1 midway through the training, and then diminishes back to 0.01. These scheduling strategies, including the constant baseline of 0.01, are depicted in Figure 5.

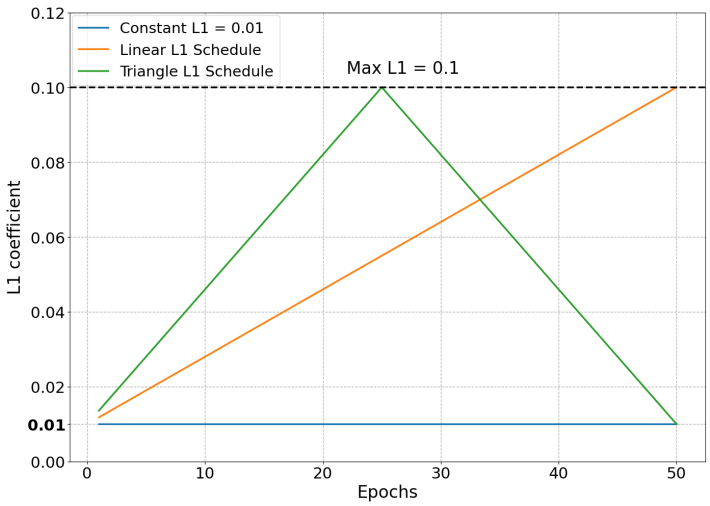

Figure 5: L1 regularization coefficient schedulers over training epochs.

6.3 Bottleneck Model

The sparsity of the message vectors is explicitly enforced by setting the output of the edge model to be the same dimension as the true number of dimensions of the system. Hence, there is no regularization added to the loss term, instead the edge model architecture is constrained explicitly to correspond to the true number of dimensions.

6.4 Standard Model

No regularizations or constraints are applied in this method. The output of the edge model ϕ^e is set to 100 dimensions. This method is not expected to learn the linear transformation of true forces since the sparsity of the latent space is not encouraged implicitly nor explicitly.

7 Methodology

The training of the models utilizes mini-batch gradient descent with the Adam optimizer. We employ a learning rate scheduler that begins at a low rate, peaks at a maximum, and then descends to a rate even lower than the starting rate. This dynamic adjustment of the learning rate is managed using the OneCycleLR policy, where the peak learning rate is set at 10^{-3}, and the final learning rate is reduced by a factor of 10^5. Additionally, we apply an L2 regularization with a weight decay set to 10^{-8} on the model parameters of the GN. The training process is performed end-to-end, where the output from the edge model is fed into the node model. The primary loss function used is the mean absolute error, which measures the discrepancy between the predicted and true instantaneous accelerations. The overall loss for the model, which integrates the base loss, network regularization, and L1 regularization (as an example of message regularization), is detailed in Equation 5.

$$
\begin{aligned}
\mathcal{L} &= \mathcal{L}_v + \alpha_1 \mathcal{L}_e + \alpha_2 \mathcal{L}_n, \\
\mathcal{L}_v &= \frac{1}{N_v} \sum_{i \in \{1:N_v\}} \left| v_i' - \hat{v}_i' \right|, \\
\mathcal{L}_e &= \frac{1}{N_e} \sum_{k \in \{1:N_e\}} \left| e_k \right|, \\
\mathcal{L}_n &= \sum_{l \in \{1:N_l\}} \left| w_l \right|^2, \\
\alpha_1 &= 10^{-2}, \quad \alpha_2 = 10^{-8}
\end{aligned}
\tag{5}
$$

Note that the $\mathcal{L}_e$ term is 0 for the bottleneck and standard models, and defined as the expression in 3 for the KL model.

8 Experiments and Results

Each model undergoes individual training for different physical systems over 50 epochs, utilizing a batch size of 64 and processing 750,000 training graphs per epoch. Following each epoch, the model's performance is assessed using a previously unseen test dataset comprising 1,000 graphs. The test loss, representing the mean absolute error between the predicted and actual acceleration values, is computed. Table 1 displays the loss values for each model at the conclusion of the training, after the final epoch.

Sim.	Standard	KL	Bottleneck	L_1-Constant	L_1-Linear	L_1-Triangle
Charge-2	0.01282	0.74032	**0.01045**	0.01328	0.01443	0.01473
r^{-1}-2	0.00867	1.79569	**0.00699**	0.01350	0.01460	0.01516
r^{-2}-2	0.01180	1.01969	**0.01071**	0.01867	0.01769	0.02166
Spring-2	0.01684	1.92537	**0.01361**	0.01667	0.02092	0.02047

Table 1: Test prediction losses for each model on selected datasets in two dimensions, with the lowest loss in each row highlighted in bold. The training was done with the same batch size, schedule, and number of epochs.

Following the assessment of the GNs' overall performance, the subsequent step involves evaluating the individual edge models for each system and analyzing the sparsity of the learned message vectors. This is achieved by fitting a linear transformation to the true forces (derived from the input node features) against the two most variant edge messages. The correspondence between message vectors and the linear combination fits of true forces is illustrated through scatter plots. The process is outlined in the following steps:

- Identify the two components with the highest variance from the 100-dimensional output of the edge model, which correspond to the true number of dimensions. In the case of the KL regularization model, select the dimensions with the highest KL divergence instead of variance. These components are termed the **most important** message vector components, expected to linearly relate to the true forces.
- Fit a linear transformation to these most important message vector components, using true forces as the input.
- Generate two scatter plots, each corresponding to a true dimension, to display the relationship between the learned message vector and the fitted linear transformation of the true forces.
- Display the standard deviation of the 15 most variant dimensions of the message vectors using box plots beneath the scatter plots. This visualization should confirm the achievement of sparsity in the edge messages, aligning them with the actual number of dimensions.

The initial analysis of the message vector components was conducted using the model with L1 regularization. The model trained on the spring system with a constant L1 coefficient did not achieve the desired sparsity in the edge messages, as shown in Figure 6. Ideally, we would observe a linear relationship between the true forces and the learned message vectors in the scatter plot, provided the message vectors are sparse and align with the true number of dimensions. The average R^2 value for both dimensions was 0.85, indicating that the model explains 85% of the variance in the message vectors. While this is significant, it falls short of the results reported in the main paper for the L1 model trained on the spring system, both in terms of R^2 value and edge message sparsity. Consequently, we explored different L1 coefficient scheduling models, as detailed in the subsection on **L1 Regularization** under the section **Models**.

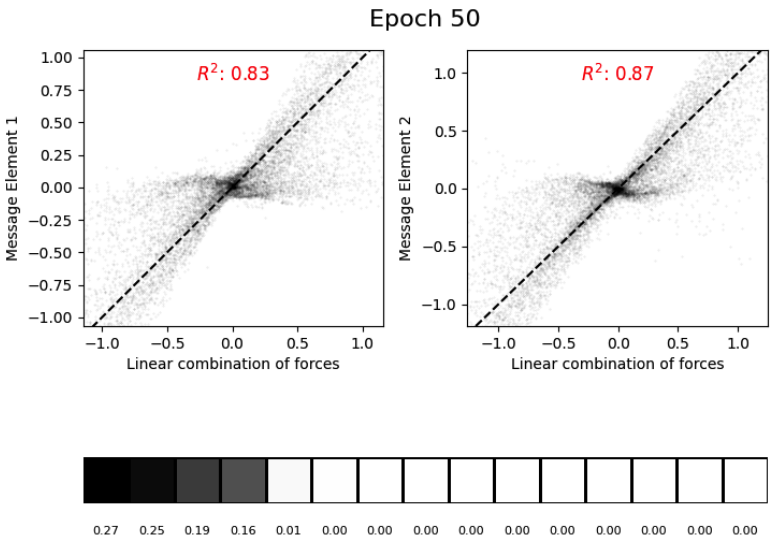

Figure 6: Scatter plot displaying the relationship between the learned message vector and the fitted linear transformation of the true forces for the L1-constant-coefficient model trained on the spring system.

A linear schedule was applied to the L1 coefficient, as depicted in Figure 5, incrementing it from 0.01 to 0.1 over 50 epochs. The resulting scatter plot, illustrating the sparsity of the learned messages for the spring system, is presented in Figure 7. The R^2 values for both dimensions increased to an average of 0.97, also confirmed by the alignment of the sparsity of edge messages with the true number of dimensions.

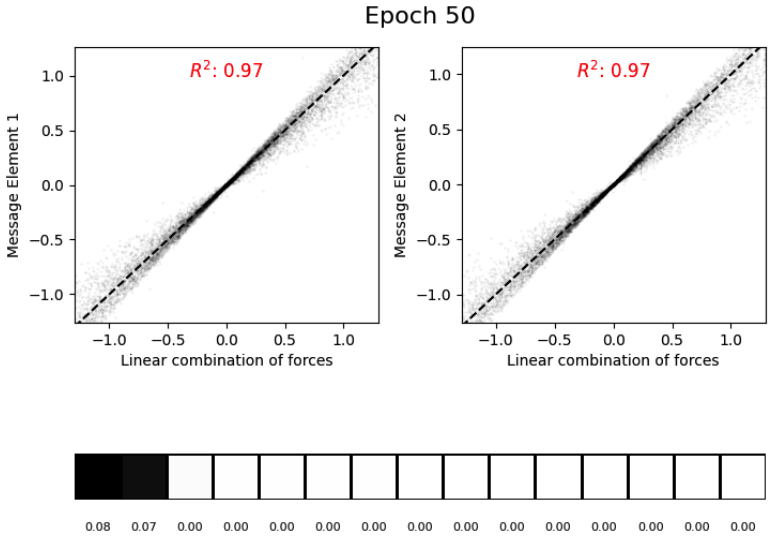

Figure 7: Scatter plot displaying the relationship between the learned message vector and the fitted linear transformation of the true forces for the L1-linear-schedule model trained on the spring system.

The L1 model was trained across all systems using three different scheduling approaches, and the resulting message vectors were fitted to the true forces. The R^2 values for these linear transformation fits are presented in Table 2, with the highest values for each system highlighted in bold. The linear schedule consistently achieved the highest R^2 values for the charge, r^{-1}, and spring systems, whereas the constant L1 coefficient schedule was more effective for the r^{-2} system. Given that the highest average R^2 values were observed in the charge and spring systems across all scheduling models, the linear schedule was selected for further use due to its superior performance compared to the triangle and constant L1 schedules in these contexts. Additional details on the selection and performance of

L1 regularization schedules are discussed in the **Inductive Biases** subsection within the **Discussion** section.

Sim.	L_1-Constant	L_1-Linear	L_1-Triangle
Charge-2	0.800	**0.810**	0.800
r^{-1}-2	0.545	**0.550**	0.500
r^{-2}-2	**0.545**	0.530	0.410
Spring-2	0.850	**0.970**	0.840

Table 2: The average R^2 value of a fit of a linear combination of true force components to the message components for different L_1 regularization schedules, applied on pruned data values across two dimensions. Numbers close to 1 indicate the messages and true force are strongly correlated.

Subsequently, each model was independently trained across all systems. The R^2 values for the KL regularization, Standard, Bottleneck, and L1-linear regularization models are displayed in Table 3. Four critical insights emerge from these results when compared to those reported in the original paper:

- The Standard model exhibits a stronger linear relationship with the true forces than previously reported.
- Among the models, only the Bottleneck and L1-Linear models, when applied to the spring system, achieve R^2 values exceeding 0.87.
- The Charge system, when trained with L1-Linear model, demonstrates significantly higher R^2 value of 0.810 compared to the 0.004 of the original paper.
- High R^2 values do not necessarily equate to the accurate recovery of known physical equations, a phenomenon also observed in the original paper with KL model trained on the spring system. We hypothesize that accurate equation recovery can occur at moderate R^2 values if the edge model's sparsity correctly reflects the system's true dimensions, as discussed in the subsequent section.

Sim.	Standard	Bottleneck	L_1-Linear	KL
Charge-2	0.515	0.605	**0.810**	-0.01
r^{-1}-2	**0.560**	0.549	0.550	-0.04
r^{-2}-2	0.525	**0.545**	0.530	-0.03
Spring-2	0.590	0.875	**0.970**	-0.08

Table 3: The average R^2 value of a fit of a linear combination of true force components to the message components for all the models, applied on pruned data values across two dimensions. Numbers close to 1 indicate the messages and true force are strongly correlated.

9 Symbolic Regression

Symbolic regression was performed using the **PySR** library, configured with 100 iterations and the binary operators ["+", "-", "*", "/"]. The loss function employed was the mean absolute error. Each sub-section of the GN - the edge model and the node model- was fitted independently. A dataset of $1,000$ unseen test samples was processed through the GN, capturing outputs from both the edge and node models. The symbolic regression on the edge model was performed to predict the most significant message elements corresponding to the number of dimensions of the system, with the node features as the input. The significance is calculated the same way as previously in preparation for the linear transformation fits above. The following section lists all successfully recovered known physical equations, utilizing L1 regularization with a linear schedule and Bottleneck constraints.

- Spring, 2D, L_1-Linear (expect $\phi_1^e \approx (a \cdot (\Delta x, \Delta y))\frac{(r-1)}{r} + b$).

$$\phi_1^e \approx \frac{(\Delta x + \Delta y) \cdot (r - 0.9916974)}{r \cdot (m_1 + 6.990457)}$$

- Spring, 2D, Bottleneck (expect $\phi_1^e \approx (a \cdot (\Delta x, \Delta y))\frac{(r-1)}{r} + b$).

$$\phi_1^e \approx 3.3775425 \cdot \frac{(\Delta x - 1.9880319 \cdot \Delta y - \frac{\Delta y}{m_1}) \cdot (r - 0.9960446)}{r}$$

- Charge, 2D, L_1-Linear (expect $\phi_1^e \approx (a \cdot (\Delta x, \Delta y))\frac{(q_1 q_2)}{r^3} + b$).

$$\phi_1^e \approx \frac{q_1 q_2 (0.19911274 \Delta x + 0.19911274 \Delta y)}{r^3 (m_1 + 1.7039468)}$$

- Charge, 2D, Bottleneck (expect expect $\phi_1^e \approx (a \cdot (\Delta x, \Delta y))\frac{(q_1 q_2)}{r^3} + b$).

$$\phi_1^e \approx 26.9520032830977 \cdot \frac{q_1 q_2 (\Delta x + \Delta y)}{r^3 m_1}$$

- $1/r$, 2D, L_1-Linear (expect $\phi_1^e \approx a \cdot (\Delta x, \Delta y)\frac{m_1 m_2}{r^2} + b$).

$$\phi_1^e \approx 0.026185596 \cdot m_2 \left(\frac{2.31938712143601 \cdot \Delta x + \Delta y}{r^2} - \frac{0.03043541}{m_2} \right)$$

- $1/r$, 2D, Bottleneck (expect $\phi_1^e \approx a \cdot (\Delta x, \Delta y)\frac{m_1 m_2}{r^2} + b$).

$$\phi_1^e \approx m_2 \left(\frac{3.99794777344893 \cdot \Delta x + \Delta y}{r \cdot (0.273109949422769 \cdot r + 0.00269607778608336)} \right) - 0.03861145$$

- $1/r^2$, 2D, Bottleneck (expect $\phi_1^e \approx a \cdot (\Delta x, \Delta y)\frac{m_1 m_2}{r^3} + b$).

$$\phi_1^e \approx 4.306725 \cdot m_2 \cdot \frac{(\Delta x + 4.12451991619306 \cdot \Delta y)}{r^3} + 0.16935113$$

- $1/r^2$, 2D, L_1-Linear (expect $\phi_1^e \approx a \cdot (\Delta x, \Delta y)\frac{m_1 m_2}{r^3} + b$).

$$\phi_1^e \approx -0.0667406 \cdot m_2 \cdot \frac{(-0.6503377 \cdot \Delta x + \Delta y)}{r^3}$$

It is important to notice that several symbolic expressions display m_1 term, which represents the mass of the target particle and should be treated as a constant. This term can be integrated into the constants a and b of the expected linear transformation equations. Specifically, in the context of spring and charge systems, m_1 consistently appears in the denominator, whereas in $1/r$ and $1/r^2$ systems, it gets canceled since the expected force involves multiplication of source and target particle's mass. From these results, we conclude that the learned latent space of message vectors is learned to be the linear transformation of the true forces normalized by the target particle's mass. While these latent variables closely approximate the acceleration of the target particles, they are not direct accelerations but rather the pairwise forces normalized by mass, aligning closely with Newton's second law in terms of pairwise interactions. This representation is anticipated, given that the entire model is trained end-to-end with acceleration labels. The subsequent aggregation step, which computes the net force, involves summing these normalized pairwise forces, thereby yielding linear transformation of accelerations. Furthermore, the node model learns the true instantaneous accelerations by deducing the inverse linear mapping from the normalized linear transformation of the net force to the actual acceleration values of the target particles. This analysis is applied to all systems and models, with the subsequent table illustrating those instances where the physical equations are successfully recovered:

Model/System	Standard	KL	Bottleneck	L_1-Linear
Charge-2	✗	✗	✓	✓
r^{-1}-2	✗	✗	✓	✓
r^{-2}-2	✗	✗	✓	✓
Spring-2	✗	✗	✓	✓

Table 4: Correct recovery of known physical equations across different models and systems.

The symbolic distillation applied on L1-Linear and Bottleneck models successfully recover the correct physical law equations across all systems. In contrast, the Standard and KL models fail to correctly recover these equations for any system. Further analysis involved examining the sparsity of the message vectors. We define the correct sparsity of latent space as having the standard deviations of the two most significant message vector components exceed those of the other components by at least a factor of 100. Notably, the Bottleneck models inherently enforce this sparsity through architectural constraints, which is indicated by boxed check marks in the accompanying table that displays the successful attainment of the desired sparsity dimension of 2.

Model/System	Standard	KL	Bottleneck	L_1-Linear
Charge-2	✗	✗	✓	✓
r^{-1}-2	✗	✗	✓	✓
r^{-2}-2	✗	✗	✓	✓
Spring-2	✗	✗	✓	✓

Table 5: The correct dimensions of the sparse latent space corresponding to the true number of dimensions of the system across different models.

We observe that the R^2 values for each system under the Bottleneck and L1-Linear models range from 0.530 to 0.970, generally lower than the near-perfect values reported in the original paper - except for the charge system. Moreover, symbolic distillation of all the systems with the correct sparsity dimension, regardless of the R^2 values, found to be successful. A more detailed discussion of these findings is provided in the **Discussion** section.

Next we perform symbolic regression on the node model to extract expressions that map normalized linearly transformed net forces to the target's true acceleration. This process utilizes the **most important** aggregated message vector components and node features as inputs. The aggregation step is performed manually by using the connectivity matrix of the graphs to calculate the cumulative message vector on each node by other nodes. This step introduces a strong inductive bias in the symbolic regression, as it inherently assumes that accelerations are derived from the net force on a particle—an assumption not explicitly stated in the original paper but critical to the aggregation logic of the GN implemented in symbolic regression. It is crucial to note that the node model does not directly reflect Newton's second law of motion, as the division by the target mass is pre-learned in the edge model, as previously demonstrated. By integrating the symbolic equations derived from both node and edge models, we calculate the mean absolute error loss for acceleration components on a test dataset of size $1,000$. This metric is averaged over components to obtain a single loss value, and the results presented in Table 6 pertain only to the successfully recovered force equations from Table 5 for the L1-Linear and Bottleneck models across all systems. The test prediction losses for the fully trained GNs, as presented in Table 1, are consistently lower across all cases. Additionally, the losses from the Bottleneck symbolic equations are smaller than those from the L1-Linear symbolic equations -except from the $1/r$ system- , mirroring the trend observed in the GN losses. It is important to note that in the original paper the claim of symbolic equations generalizing better to the unseen data compared to the trained GNs is only made for the cosmology example with the dark matter density prediction. In this paper, we do not replicate the cosmology example but rather test this claim on classical mechanics systems which does not hold.

Sim.	Bottleneck Symbolic Equation	L_1-Linear Symbolic Equation
Charge-2	1.41716	2.71826
r^{-1}-2	3.30026	1.32959
r^{-2}-2	1.27363	1.49345
Spring-2	1.64237	2.85806

Table 6: Test prediction losses for symbolic equations extracted from Bottleneck and L1-linear models trained separately on all the systems.

10 Discussion

10.1 Claims

In summary, majority of the claims were reproduced mainly:

- **Claim 1:** The sparse latent representation of the learned messages, corresponding to the true number of dimensions of the systems is achieved using L1 regularization with linear schedule and Bottleneck models.
- **Claim 2:** Symbolic equations of the spring, charge, and Newtonian systems are successfully recovered using symbolic regression which yielded linear transformation of true physical laws.
- **Claim 3:** The test performance of the symbolic models on an unseen data were found to be worse than the fully trained GN performance, which disagreed with Occam's Razor argument on selected classical systems.

The contributions in the light of reproducing these results and analysis are:

- Implementing a linear and triangular scheduler on the L1 regularization coefficient in order to provide flexibility to the GN in early stages of training to explore wider range of latent space representations. The start value is set to 0.01 which is the default value in the original paper, and is increased up to 0.1 in order to achieve similar level of sparsity in reduced number of epochs-50- compared to the original paper-200-.
- Data-pruning based on simulated acceleration values. We observe that this step is crucial for discarding extreme high acceleration values on the order of 10^5 which contributed to the next point.
- Achieving desired sparsity and higher R^2 values compared to those reported in the original paper for charge system with L1 regularization and Bottleneck models, as well as recovering correct charge laws, that were not able to be recovered in the original paper.

10.2 Inductive Biases

A key inductive bias in this study is the regularization strategy used to achieve a sparse representation of the message vectors. The constant L1 coefficient previously employed did not yield the desired sparsity or an accurate linear transformation of the true forces. Consequently, we adopted two different scheduling approaches —triangle and linear— to modulate the L1 coefficient during training. Both schedules start at a minimum value of 0.01 and escalate to a maximum of 0.1. The motivation for selecting the maximum value for the schedules(0.1) as ten times of the constant value in the original paper (0.01) is to achieve a similar level of sparsity in a reduced number of training iterations—50 epochs as opposed to 200 in the original study. Additionally, the initial low value of 0.01 allows the GN to explore a broader range of solutions early in training, thus preventing premature convergence to suboptimal sparse configurations. This approach mirrors strategies used in learning rate scheduling within stochastic gradient descent algorithms, where initial phases feature lower rates to prevent overshooting the optimal solution [10].

We hypothesize the reason for triangle schedule to not perform as good as the linear schedule might be due to the sharper increase of the coefficient, reaching the maximum value halfway through the

training. This shorter exploration period in the latent space could potentially explain the diminished performance compared to the linear schedule, which allows more gradual adaptation.

10.3 Explicit vs Implicit Regularization

Among the implicit regularization methods, where the GN architecture is not explicitly constrained by a predetermined dimensionality, the L1 regularization with a linear schedule outperformed the KL-divergence approach. The KL model's R^2 fit values are significantly less than those reported in the original paper, indicating that neither sparsity of messages nor accurate linear transformation of true forces was achieved using KL divergence as the regularization term in the loss function. This suboptimal performance of the KL model may stem from an inappropriate choice of reference distribution for minimizing the KL divergence. While the original paper uses a unit Gaussian prior—a reasonable choice, and arguably superior to a uniform distribution—a more effective prior might be selected to better promote sparsity in message vectors. The Laplace distribution, with its sharp peak at zero, naturally encourages the reduction of insignificant values while preserving significant ones [5]. Therefore, enhancing sparse latent representations of message vectors, the Laplace distribution could improve the performance of the KL model and would be interesting to investigate for future research.

Interestingly, while Bottleneck models demonstrate the lowest test prediction losses across all physical systems, as indicated in Table 1, the L1 regularization with a linear schedule exhibits significantly higher R^2 fit values in the charge (0.810) and spring systems (0.970), as shown in Table 3. This observation suggests that the linear scheduling of L1 regularization facilitates a more gradual adaptation to the true forces, allowing the GN to explore a broader range of latent spaces. This approach potentially enables a more precise convergence to the linear transformation of the forces compared to the fixed constraints imposed by the Bottleneck models on message dimensions. It is important to note that the final test performance of the models, as recorded, is based solely on the base loss described in Equation 5 and is assessed only on the instantaneous accelerations. There exists a possibility that while Bottleneck models may struggle to learn an accurate latent space for linearly transformed true forces within the edge model, the latent space they do learn could be ideally suited for minimizing the acceleration prediction error of the node model.

10.4 Data Pruning

Data pruning is essential for eliminating large acceleration values on the order of 10^5, an additional step beyond the original paper's methodology. This pruning is crucial for the results, although such high acceleration values can be justified as they may arise from later stages of simulations involving closely interacting particles. Moreover, for future studies, introducing a minimal threshold for pairwise distances in gravitational systems ($1/r$ and $1/r^2$) could prevent potentials from approaching infinitely large values when particles are extremely close.

A significant observation was made regarding the charge system when trained on various models. While the original paper generally shows superior performance across trained models, our results in the charge system excel in achieving the desired sparsity and R^2 values for linear transformation fits. This distinction is likely linked to higher number of outliers in the charge system data which are filtered in this study. In comparison, the spring system data exhibits the narrowest range of acceleration values, from -300 to 300, whereas the charge system's values range from -150,000 to 100, 000. We observed a clear correlation between the range of acceleration values in the simulation data and the performance of the trained models. Specifically, the spring system, which had the narrowest data range, exhibited the best performance. This correlation implies that extreme acceleration values may skew loss calculations and subsequently affect model training. By excluding these outliers from the simulation data prior to model input, we enhanced model performance, leading to the successful symbolic distillation of charge systems using L1 regularization in our experiments.

10.5 Symbolic Regression

We observed successful recovery of known equations, reflected in moderately high R^2 values. However, the R^2 values for the L1-Linear and Bottleneck models were lower than those reported in the original paper, with the exception of the charge system. Despite this, all models that correctly distilled the

equations managed to achieve the appropriate number of dimensions in their latent space representations. This suggests that accurate equation recovery is possible with moderate R^2 values, provided that the learned sparse message vectors match the system's true dimensions.

Furthermore, the theoretical framework suggests that learned messages should represent a linear transformation of the true forces if the messages are sparse and dimensionally accurate. However, our experimental results, with R^2 values ranging from 0.530 to 0.970, did not fully align with this expectation. Detailed possible explanations for these discrepancies are provided in the next subsection.

10.6 Possible Explanation for the Difference in Performance

We identify three potential explanations for the discrepancies in fit results (R^2 values) between the original study and our findings. Although the majority of the symbolic equations are successfully distilled, the correlation between the linear transformation of true forces and the learned message vectors was not as strong as reported in the original study. The possible reasons for this variance are outlined below:

- There may be inaccuracies in our code implementation for fitting the linear transformation, despite thorough checks. We invite readers to review our code and propose any enhancements.
- The theoretical framework which suggests that the output of the edge model, given it matches the true number of dimensions, is guaranteed to represent a linear transformation of the true forces, may need to be reevaluated. This reconsideration is supported by the original paper's findings, where the R^2 values for the KL model trained on the 3D spring system and the 3D $1/r$ system were only 0.214 and 0.332, respectively, yet both models successfully recovered the corresponding force laws through symbolic regression.
- Achieving a linear transformation of the true forces might require more training epochs than what was used-50- in this reproduction study, which was constrained by time. While the chosen number of epochs was adequate for converging to the correct dimensions of the sparse latent space, it raises concerns about the interpretability of the learned messages, which do not seem to strongly represent a linear transformation of the true forces. Interestingly, symbolic regression was capable of deriving the correct physical laws from these dimensionally accurate yet arbitrary latent spaces. These observations do not apply to models trained on the spring system, where R^2 values as high as 0.970 were achieved, as illustrated in Table 3. In these cases, the learned messages accurately correspond to the linear transformation of true forces.

10.7 Utilizing Computer Resources

For this reproduction study, the Cambridge Service for Data Driven Discovery (CSD3) was utilized to train the models. The sustainability aspect of such resources were considered to make mindful choices of the use of computational resources. The training time of each model and simulated system makes up a significant part of the computational cost of this study. Hence, the total number of training epochs was limited to 50, instead of 200 as in the original paper. Considering there are 4 systems and 6 regularization model each to be analyzed, this reduction in training time saved $6 \times 24 = 144$ hours of single GPU node use. This consideration is both efficient for the reproduction study time and the computational cost of the study while being able to reproduce most of the results in the main paper.

11 Conclusion

In conclusion, this reproduction study has effectively demonstrated the application of symbolic regression within the framework of a message-passing GN, specifically targeting its separable components. By integrating deep learning with symbolic regression, we have improved the interpretability of the models, revealing the underlying physical principles governing the system architectures. The inherent modularity of GNs, with their distinct node and edge structures, aligns well with Newtonian physics, making the symbolic regression tasks more manageable and reducing the computational complexity involved in exploring the solution space. We adopted the use of L1 regularization and the Bottleneck model to achieve a sparse latent representation that accurately reflects the true dimensional structure

of the systems studied. Although the R^2 values for most systems were lower than those reported in the original study—with the exception of the spring and charge systems, where we achieved values as high as 0.970—the correct physical laws were still successfully derived from these representations. This indicates that while the limited number of training epochs-50- was sufficient to reach the correct dimensions, it may not have fully captured the linear transformation of the true forces. This limitation was particularly noticeable outside of the spring system scenarios, suggesting a potential area for further investigation and optimization in future studies. Additionally, methodological improvements such as data pruning and the implementation of a scheduling mechanism for L1 regularization led to higher R^2 values and the successful symbolic distillation in the charge system which was not accomplished in the original paper. These improvements underscore the significance of adaptability and methodological precision in reproducing foundational research in the intersection of deep learning and symbolic regression, leveraging strong inductive biases to enhance the interpretability of black-box models. Future studies could further optimize training epochs and consider using a Laplacian distribution as a prior for the KL Model to promote better sparsity compared to a Gaussian prior.

12 Acknowledgments

The author would like to thank Miles Cranmer, the first author of the original paper, for his guidance and suggestions on how to improve the analysis of the study. The idea of using a scheduler for the L1 regularization coefficient was suggested by him, during a fruitful discussion on enhancing the exploration of a broader range of latent spaces in the early training stages, and preventing premature convergence to suboptimal sparse configurations.

References

[1] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu. Relational inductive biases, deep learning, and graph networks, 2018.

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

[3] M. Cranmer, A. Sanchez-Gonzalez, P. Battaglia, R. Xu, K. Cranmer, D. Spergel, and S. Ho. Discovering symbolic models from deep learning with inductive biases, 2020.

[4] T. Hastie, R. Tibshirani, and M. Wainwright. *Statistical Learning with Sparsity: The Lasso and Generalizations*. Chapman & Hall/CRC, 2015.

[5] J. Lu, D. G. Politte, and J. A. O'Sullivan. Laplacian prior variational automatic relevance determination for transmission tomography, 2017.

[6] J. McFadden. Razor sharp: The role of occam's razor in science. *Annals of the New York Academy of Sciences*, 1530(1):8–17, 2023.

[7] H. Oh, R. Amici, G. Bomarito, S. Zhe, R. Kirby, and J. Hochhalter. Genetic programming based symbolic regression for analytical solutions to differential equations, 2023.

[8] S. J. Prince. *Understanding Deep Learning*. The MIT Press, 2023.

[9] D. R. Stoutemyer. Can the eureqa symbolic regression program, computer algebra and numerical analysis help each other?, 2012.

[10] Y. Wu and L. Liu. Selecting and composing learning rate policies for deep neural networks, 2022.

A Generative AI

ChatGPT 4o was used for suggesting alternative wordings, grammar suggestions, and proofreading while writing the report. The following prompts were used during this process

- How to represent/write *input symbol* in LaTeX?
- What is another word for *input word*?
- Detect the grammar mistakes in this paragraph: *input paragraph*
- Does this sound logical to a reader? *input paragraph*
- How can I write this in a more clear and direct way? *input sentence*

The outputs were selectively adapted, incorporating only alternative wordings and not the entire output while rephrasing the discussion parts of the report.

Furthermore, the suggestions from the autocomplete feature of **GitHub Copilot**, as well as the **Cursor** tool, were utilized during the documentation of the software and code development of the project. These tools were particularly helpful in editing and adjusting plotting settings for images and figures, as well as writing the LaTeX format labels for the plots generated. Moreover, the sections below are adapted from Dr. Miles Cranmer's public repository for the original paper for the sake of this reproduction study:

- 'utils/simulate.py': Contains functions and classes for particle simulations.
- Functions 'linear_transformation_2d()', 'out_linear_transformation_2d()', and 'percentile_sum()' in the 'src/analyze.py' script.