Lagrangian and Hamiltonian Neural Networks With a Dissipative System

Anonymous authors
Paper under double-blind review

Abstract

We investigate the applicability of Lagrangian and Hamiltonian Neural Network models to a dissipative system that has explicit time dependence in its Lagrangian, Hamiltonian, and total energy. To do so we consider these neural network models for simulated systems of a harmonic one-dimensional, one-component oscillator with damping, as well as without damping for comparison. We find that both the Lagrangian and Hamiltonian approaches are able to predict the empirical physical behavior of the damped oscillator systems and to effectively "learn" to varying degrees the underlying Lagrangians and Hamiltonians, as has previously been shown to be the case with undamped oscillator systems. These investigations elucidate important properties of Lagrangian and Hamiltonian mechanics, including properties that are not manifest when considering systems without explicit time dependence.

1 Introduction

Artificial Neural Networks (NNs) are machine learning models which develop non-linear functional mappings from input parameters to outputs by assigning weights which relate the values of layers of neuron nodes, with the weights adjusted in training to minimize a loss characterizing the difference between the outputs and the known target outputs of a training set (1). NNs have widespread application including in physics (e.g. 2) and astronomy (e.g. 3; 4). As with other machine learning models, NNs are generally *empirical* in that the relations between input parameters and outputs are complicated highly non-linear functions that are empirically optimized to fit data but are effectively a "black box" to a user. This is the case even in typical applications of NNs in physics and astronomy.

However, there has been interest in NN models which are not solely empirical but rather can be used to elucidate underlying features of a system, as is the case with analytical models involving physical laws. In Physics-Informed Neural Networks (e.g 5) the loss function is designed to train the network model to both fit the data and have it conform to the known equations that model the behavior of the system. Recently, works have explored Lagrangian NNs (6; 7) and Hamiltonian NNs (8; 9). In classical mechanics the Lagrangian and Hamiltonian are functions of observable dynamical variables of a system from which the equations of motion for coordinates of the system can be derived, but are not themselves observable. While not directly observable, they reflect the underlying fundamental properties of a system such as its symmetries (e.g. 10; 11).

The fundamental idea behind both Lagrangian NNs and Hamiltonian NNs is that in the training process the NN model learns the Lagrangian or Hamiltonian of the training systems by taking certain dynamical variables (or observations which encode them) as inputs and then minimizing the loss between known target values for certain other dynamical variables and those predicted by derivatives of the Lagrangian or Hamiltonian. The model can then be applied to calculate the Lagrangian or Hamiltonian functions and dynamical variables of test or evaluation systems. This process necessitates calculating the derivatives of the learned Lagrangian or Hamiltonian with respect to the input features of the NN, which are realized as the in-network (also referred to as "in-graph") gradients. The relevant equations are discussed below. A schematic of the basic principle of a Lagrangian NN and a Hamiltonian NN as implemented in this work is shown in Figure 1.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

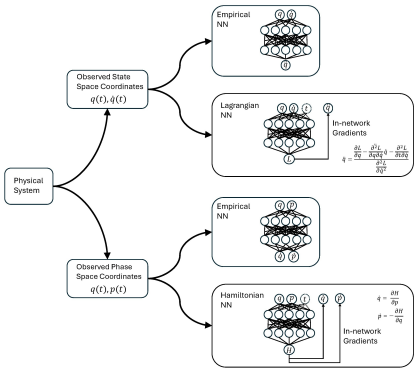


Figure 1: A schematic of the basic principle of a Lagrangian NN and a Hamiltonian NN as implemented in this work, contrasted with those of purely empirical NNs which would predict the same quantities. As discussed in §1, Lagrangian and Hamiltonian NNs can learn the Lagrangian and Hamiltonian functions of a system, which are not themselves directly observable but are important for understanding its properties, by using derivatives of the Lagrangian and Hamiltonian functions with respect to the inputs to predict other observables. There are more hidden layers, each with many more neurons, than depicted here for simplicity. Network architectures are given in Table 2.

An important limitation of the previous foundational works exploring Lagrangian and Hamiltonian NNs (6; 7; 8; 9) is that they considered systems in which the Lagrangian and Hamiltonian have no explicit time dependence. That condition usually corresponds to conservation of energy of the system. In physics many systems of interest are not isolated and are often dissipative, in which energy is lost to interactions such as friction and air resistance. The relation between the time dependencies of the Lagrangian L and the Hamiltonian H is

$$-\frac{\partial L}{\partial t} = \frac{\partial H}{\partial t} = \frac{dH}{dt}. \tag{1}$$

The Hamiltonian is thus the quantity describing a system in which all of the time dependence is explicit. It is often stated that the Hamiltonian is equal to the total energy of the system

$$H \stackrel{?}{=} E_{\text{tot}} = T + U \tag{2}$$

where T is the kinetic energy and U the energy of interactions. However this is only true if the system is such that the kinetic energy is quadratic in the generalized velocities and the interaction term does not depend on the generalized velocities (e.g. 10). In a typical damped system with a velocity-dependent damping force such as investigated in this work the Hamiltonian is not equal to the total energy, and, containing explicit time dependence, is also not a conserved quantity.

Sosanya & Greydanus (12) modeled dissipative systems in the Hamiltonian approach with a neural network that contained subnetworks for learning both the (non-dissipative) Hamiltonian function of the system and a function that described the dissipation. Sundararaghavan et al. (13) modeled dissipative systems in the Lagrangian approach with a transformation to a coordinate space containing twice the number of generalized coordinates.

Here we explore implementing Lagrangian and Hamiltonian NNs in a similar, straightforward method to the works of Cranmer et al. (6) and Greydanus et al. (8), respectively, but for a *simple dissipative system without conservation of total energy* and in which *the Hamiltonian is not constant nor equal to the total energy*, in which *the Lagrangian and Hamiltonian are explicitly time dependent*. We consider the system of a one-dimensional, one component harmonic oscillator, both without damping for a system with no explicit time dependence for comparison, and with damping for a system with explicit time dependence.

2 Simulated Oscillator Systems and Network Architectures

For training data we simulate between 30 and 1200 (depending on the case; see Table 2) different position versus time ($q(t)$) trajectories of a one-dimensional, one-component oscillator according to the $q(t)$ equations given by equation 16 for no damping present and equation 20 for damping

present, with time steps of 0.1 s, for 314 s. For simplicity we set the values of the restoring force constant k to 1 N/m, the mass m to 1 kg, and, in the case of the damped oscillator, the damping force constant b (in e.g. equation 18) to 0.02 kg/s to be in the weak damping regime (e.g. 10) allowing numerous oscillations before the system effectively comes to a stop. The initial total energy values are randomly assigned from a uniform distribution in the range between 0.2 and 1.0 J.

For the cases of added noise, we add Gaussian random noise to each data point drawn from a distribution with a standard deviation of $\sigma = 0.07$ or $\sigma = 0.04$, with the former value determined somewhat arbitrarily and the latter being the largest variance that resulted in successful training convergence. We supply every 100th time step data point for training in the undamped noise-free cases and for the damped noise-free Energy NN. We supply every 10th time step data point for all other cases, except the damped noisy Hamiltonian NN case, where successful training convergence required every 2nd time step data point (see Table 2). For testing data we used two additional simulated trajectories with initial total energies of 0.2 J and 1 J and used every timestep. In what follows we show results for the total initial energy of 0.2 J. Overall results for the total energy of 1 J are similar.

We extract the "observed" velocity $\dot{q}(t)$ versus time data by empirical differentiation

$$\dot{q}(t_i) = \frac{q_i - q_{i-1}}{t_i - t_{i-1}} \tag{3}$$

and the "observed" acceleration $\ddot{q}(t)$ versus time by the same procedure.

$$\ddot{q}(t_i) = \frac{\dot{q}_i - \dot{q}_{i-1}}{t_i - t_{i-1}} \tag{4}$$

For the "observed" linear momenta $p(t_i)$ needed for the Hamiltonian approach we use $m\dot{q}(t_i)$. We note that for the case of a damped oscillator this linear momentum is not equal to the generalized momentum which is used in the Hamiltonian, a subject which is discussed in §4 and §5. We extract the "observed" derivatives of the linear momenta versus time data again by empirical differentiation

$$\dot{p}(t_i) = \frac{p_i - p_{i-1}}{t_i - t_{i-1}}. \tag{5}$$

We take in-network gradients with respect to input parameters with the Pytorch function `torch.autograd()` which computes the gradients of a specified output parameter to a specified input parameter. To take second derivatives we set the output parameter to the already computed first derivative. In some cases we employ a lowpass filter with `savgol_filter` to reduce noise when computing gradients.

We determined network hyperparameters largely through a trial-and-error process. We do not claim that any hyperparameters implemented here have been optimized, nor is such an optimization within the scope of this work or necessary to demonstrate its conclusions. Hyperparameters are given in Table 2.

3 Lagrangian Neural Network

In classical mechanics, the Lagrangian

$$L(q_i, \dot{q}_i, t) = T - U \tag{6}$$

is a function of the generalized coordinates $q_i(t)$ and generalized velocities $\dot{q}_i(t)$ of the system. The equations of motion of the system for each generalized coordinate are given by the Euler-Lagrange equations

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = 0. \tag{7}$$

If there is only one generalized coordinate q we define

$$\frac{\partial L}{\partial \dot{q}} \equiv f(q, \dot{q}, t) \tag{8}$$

so that

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) = \frac{df(q, \dot{q}, t)}{dt}. \tag{9}$$

By the chain rule

$$\frac{df(q,\dot{q},t)}{dt} = \frac{\partial f}{\partial q}\dot{q} + \frac{\partial f}{\partial \dot{q}}\ddot{q} + \frac{\partial f}{\partial t}. \tag{10}$$

Substituting this into equation 7, gives

$$\frac{\partial^2 L}{\partial q \partial \dot{q}}\dot{q} + \frac{\partial^2 L}{\partial \dot{q}^2}\ddot{q} + \frac{\partial^2 L}{\partial t \partial \dot{q}} - \frac{\partial L}{\partial q} = 0 \tag{11}$$

and then solving for $\ddot{q}$ gives

$$\ddot{q} = \frac{\frac{\partial L}{\partial q} - \frac{\partial^2 L}{\partial q \partial \dot{q}}\dot{q} - \frac{\partial^2 L}{\partial t \partial \dot{q}}}{\frac{\partial^2 L}{\partial \dot{q}^2}}. \tag{12}$$

Therefore in principle it should be possible for a neural network to be trained to determine the Lagrangian of a system given a training set of trajectories with the observed $q(t)$ and $\dot{q}(t)$, and t (as necessary) as the input parameters, taking the the in-network gradients $\frac{\partial L}{\partial q}$, $\frac{\partial L}{\partial \dot{q}}$, $\frac{\partial^2 L}{\partial \dot{q}^2}$, and $\frac{\partial L}{\partial t}$ (as necessary) with respect to those input parameters, in order to predict the observed $\ddot{q}(t)$ as the target values. The generic loss function for a Lagrangian NN with one generalized coordinate should therefore be

$$\mathcal{L}_{\text{LNN}} = \left|\left| \left(\frac{\frac{\partial L}{\partial q} - \frac{\partial^2 L}{\partial q \partial \dot{q}}\dot{q} - \frac{\partial^2 L}{\partial t \partial \dot{q}}}{\frac{\partial^2 L}{\partial \dot{q}^2}} \right) - \ddot{q} \right|\right|_2 \tag{13}$$

where the $||x - y||_2$ notation represents the standard $L2$ loss function over all training examples.[1] $q(t)$ and $\dot{q}(t)$, and t (as necessary) data for a test system can then be input to the Lagrangian NN model to predict its Lagrangian and, with in-network gradients as above, predict its $\ddot{q}(t)$ which can be compared to the observed values to evaluate its performance. Here we implement Lagrangian NN models with the hyperparameters given in Table 2, train them on the training data set trajectories, and evaluate their performance on the test data trajectories.

3.1 Lagrangian NN – Undamped case

The Lagrangian for the undamped one-dimensional, one-component oscillator is

$$L_{\text{SHO}}(q,\dot{q}) = \frac{1}{2}m\dot{q}^2 - \frac{1}{2}kq^2 \tag{14}$$

which has the equation of motion determined by equation 7

$$m\ddot{q} + kq = 0 \tag{15}$$

and solution for the behavior of the system

$$q_{\text{SHO}}(t) = A\ \cos(\omega_0 t + \phi) \tag{16}$$

where $\omega_0 \equiv \sqrt{\frac{k}{m}}$ and ϕ is a phase factor. For the undamped oscillator the $\frac{\partial^2 L}{\partial t \partial \dot{q}}$ term in equations 12 and 13 is 0 so

$$\mathcal{L}_{\text{LNN,SHO}} = \left|\left| \left(\frac{\frac{\partial L}{\partial q} - \frac{\partial^2 L}{\partial q \partial \dot{q}}\dot{q}}{\frac{\partial^2 L}{\partial \dot{q}^2}} \right) - \ddot{q} \right|\right|_2 . \tag{17}$$

[1] We note that $\mathcal{L}$ here represents a standard notation for a machine learning training loss function and should not be confused with the same variable often used to represent a Lagrangian density in field theory.

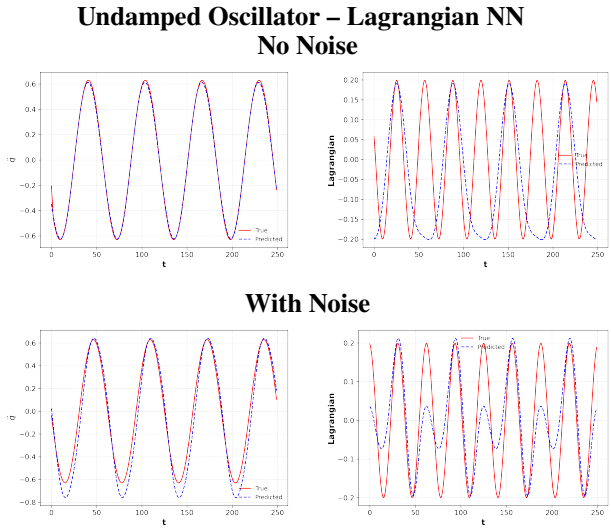


Figure 2: The actual and predicted acceleration $\ddot{q}_{\rm SHO}(t)$ versus time (**left**) and actual and predicted Lagrangian $L_{\rm SHO}$ vs time (**right**) for the undamped oscillator, for the cases of no noise (**upper**) and full simulated noise (**lower**) in the "observed" data. As discussed in §5 the Lagrangian is also ambiguous to the presence of a particular function of the generalized coordinates, which explains the apparent discrepancy between the predicted and actual Lagrangians.

The actual and predicted acceleration $\ddot{q}_{\rm SHO}(t)$ versus time and Lagrangian vs time for the undamped oscillator test data, for both the case of no simulated noise and full simulated noise in the "observed" data, are shown in Figure 2.

3.2 Lagrangian NN – Damped case

The Lagrangian for the damped one-dimensional, one-component oscillator is

$$L_{\rm D}(q,\dot{q},t) = e^{\frac{b}{m}t}\left(\frac{1}{2}m\dot{q}^2 - \frac{1}{2}kq^2\right) \tag{18}$$

where b is the damping force constant, which has the equation of motion determined by equation 7

$$m\ddot{q} + kq + b\dot{q} = 0. \tag{19}$$

For the case of 'weak damping' where $\frac{b}{2m} < \omega_0$ that we will consider here oscillations continue for multiple cycles and the solution for the behavior of the system is

$$q_{\rm D}(t) = A\, e^{-\frac{b}{2m}t}\, \cos(\omega_1 t + \phi), \text{ where } \omega_1 \equiv \sqrt{\omega_0^2 - \left(\frac{b}{2m}\right)^2}. \tag{20}$$

It is clear that in order to predict a Lagrangian of the form of equation 18, time must be an input parameter to the LNN. However we are also interested in whether a model without time as an input parameter can predict the dynamics of the system. We therefore try two LNN models: one with time as an input parameter and one without. With time as an input parameter, the loss function is given by the full equation 13. Without time as an input parameter, the loss function cannot involve calculating $\frac{\partial L}{\partial t}$ in graph, so we note that because $\frac{\partial L}{\partial t} = \frac{b}{m}L$ in this case, the $\frac{\partial^2 L}{\partial t \partial \dot{q}}$ in equations 12 and 13 can reduce to $\frac{b}{m}\frac{\partial L}{\partial \dot{q}}$ so

$$\mathcal{L}_{\rm LNN,D} = \left\| \left(\frac{\frac{\partial L}{\partial q} - \frac{\partial^2 L}{\partial q \partial \dot{q}}\dot{q} - \frac{b}{m}\frac{\partial L}{\partial \dot{q}}}{\frac{\partial^2 L}{\partial \dot{q}^2}} \right) - \ddot{q} \right\|_2 \tag{21}$$

can be used.

The actual and predicted acceleration $\ddot{q}_{\rm D}(t)$ versus time and Lagrangian versus time for the damped oscillator test data, for both the case of no simulated noise and full simulated noise in the "observed" data, for models with and without time as an input parameter, are shown in Figure 3.

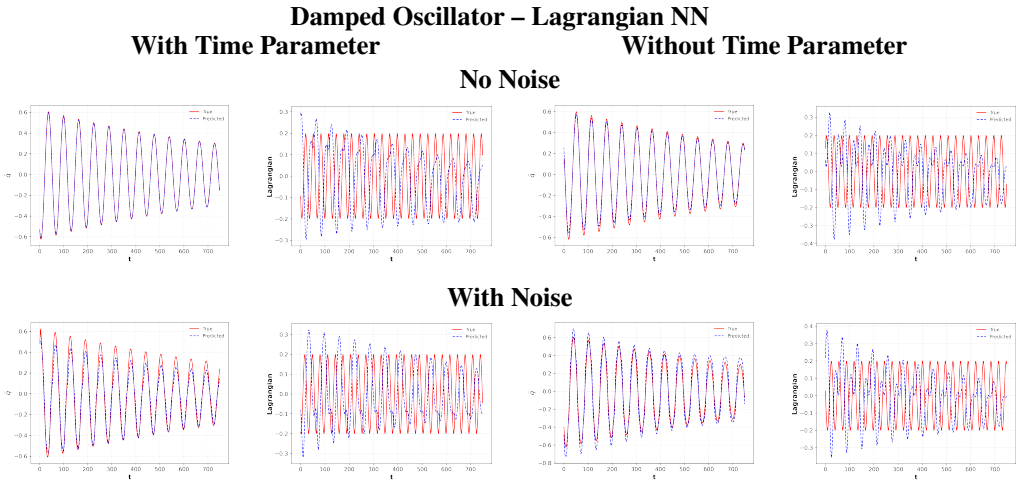


Figure 3: The actual and predicted acceleration $\ddot{q}_{\mathrm{D}}(t)$ versus time (**first, third column**) and actual and predicted Lagrangian L_{D} vs time (**second, fourth column**) for the damped oscillator, for the cases of no noise (**upper**) and full simulated noise (**lower**) in the "observed" data. As discussed in §5 the Lagrangian is also ambiguous to the presence of a particular function of the generalized coordinates, which explains the apparent discrepancy between the predicted and actual Lagrangians.

3.3 Lagrangian NN results

We see that for both the undamped and damped oscillator, the Lagrangian NN is able to successfully predict the dynamics of the system both without and with the presence of noise. However in all cases the prediction for the Lagrangian function itself seemingly does not match the system Lagrangians of equations 14 and 18. We discuss in §5 though how the models are nevertheless apparently predicting *a* valid Lagrangian for the system. In the damped case without time as an input parameter, we see that the Lagrangian NN model is still able to predict $\ddot{q}_{\mathrm{D}}(t)$. Given its ability to predict $\ddot{q}_{\mathrm{D}}(t)$ and the similarity of the predicted Lagrangians to the valid systems Lagrangians for the cases of time included as an input parameter, the form of the Lagrangians predicted by this model appear to also be valid system Lagrangians.

4 Hamiltonian Neural Network

In classical mechanics, the Hamiltonian

$$H(q_i, p_i, t) \equiv \sum_i p_i \dot{q}_i - L \tag{22}$$

is a function of the generalized coordinates (q_i) and generalized momenta (p_i) of the system, where

$$p_i \equiv \frac{\partial L}{\partial \dot{q}_i} \tag{23}$$

is the generalized momentum corresponding to generalized coordinate q_i. The equations of motion of the system are given by

$$\dot{p}_i = -\frac{\partial H}{\partial q_i} \ , \ \dot{q}_i = \frac{\partial H}{\partial p_i}. \tag{24}$$

Therefore in principle it should be possible for a neural network to be trained to learn the Hamiltonian of a system, given a training set of trajectories with the observed $q(t)$ and $p(t)$, and t (as necessary) as the input parameters, and to then take the in-network gradients $\frac{\partial H}{\partial q}$ and $\frac{\partial H}{\partial p}$ with respect to those input parameters in order to predict the observed $\dot{p}(t)$ and $\dot{q}(t)$ as the target values.

The generic loss function for a Hamiltonian NN with one generalized coordinate should therefore be

$$\mathcal{L}_{\mathrm{HNN}} = \left\| \frac{\partial H}{\partial p} - \dot{q} \right\|_2 + \left\| \left(-\frac{\partial H}{\partial q} \right) - \dot{p} \right\|_2, \tag{25}$$

where the $||x - y||_2$ notation again represents the standard $L2$ loss function over all training examples. However for the reasons discussed in Appendix 6.3 for the damped Hamiltonian NN cases and the noisy Energy NN case we instead use a SmoothL1 (Huber) loss, which is quadratic in $(x - y)$ for small residuals and linear for large residuals (Table 2). $q(t)$ and $p(t)$, and t (as necessary) data for a test system can then be input to the Hamiltonian NN model to predict its Hamiltonian and, with in-network gradients as above, predict its $\dot{q}(t)$ and $\dot{p}(t)$ which can be compared to the observed values to evaluate its performance. We implement Hamiltonian NN models with the hyperparameters given in Table 2, train them on the training data set trajectories, and evaluate their performance on the test data trajectories.

4.1 Hamiltonian NN – Undamped case

The Hamiltonian for the undamped one-dimensional, one component oscillator is

$$H_{\text{SHO}}(q, p) = \frac{p^2}{2m} + \frac{1}{2}kq^2 \tag{26}$$

and by equation 23

$$p_{\text{SHO}}(t) = m\,\dot{q}_{\text{SHO}}(t) = -m\,A\,\omega_0\sin(\omega_0 t + \phi). \tag{27}$$

In this case the generalized momentum is equal to the linear momentum, and the Hamiltonian is equal to the total system energy. Thus for the undamped case we can straightforwardly use the "observed" $q_{\text{SHO}}(t)$ and $p_{\text{SHO}}(t) = m\,\dot{q}_{\text{SHO}}(t)$ as input parameters to the Hamiltonian NN.

The actual and predicted $\dot{q}_{\text{SHO}}(t)$ versus time, actual and predicted $\dot{p}_{\text{SHO}}(t)$ versus $\dot{q}_{\text{SHO}}(t)$, and actual and predicted Hamiltonian (and total energy) versus time for the undamped oscillator test data are shown in Figure 4.

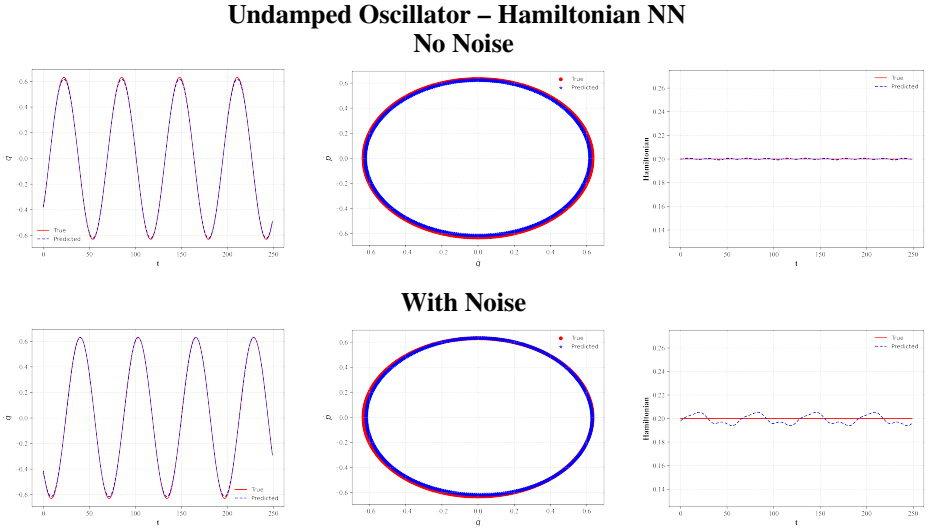


Figure 4: Actual and predicted velocity $\dot{q}_{\text{SHO}}(t)$ versus time (**left**), actual and predicted $\dot{p}_{\text{SHO}}(t)$ versus $\dot{q}_{\text{SHO}}(t)$ (**middle**), and actual and predicted Hamiltonian H_{SHO} vs time (**right**) for the undamped oscillator, for the cases of no noise (**upper**) and full simulated noise (**lower**) in the "observed" data.

4.2 Hamiltonian NN – Damped case

By equation 23 the generalized momentum for the damped, one-dimensional, one-component oscillator is

$$p_{\text{D}}(t) = e^{\frac{b}{m}t}\,m\,\dot{q}_{\text{D}}(t) = -e^{\frac{b}{2m}t}\,m\,A\left(\omega_1\sin(\omega_1 t + \phi) + \frac{b}{2m}\cos(\omega_1 t + \phi)\right). \tag{28}$$

By equation 22 the Hamiltonian is

$$H_{\text{D}}(q, p, t) = e^{-\frac{b}{m}t}\,\frac{p^2}{2m} + e^{\frac{b}{m}t}\,\frac{k}{2}q^2. \tag{29}$$

In this case the generalized momentum is not equal to the linear momentum – in fact it is an exponentially *increasing* quantity that is not a straightforward observable. We can directly observe, instead, the linear momentum

$$p_{\mathrm{D,linear}}(t) = m\,\dot{q}_{\mathrm{D}}(t) = e^{-\frac{b}{m}t}\,p_{\mathrm{D}}(t) = -e^{-\frac{b}{2m}t}\,m\,A\left(\omega_1 \sin(\omega_1 t + \phi) + \frac{b}{2m}\cos(\omega_1 t + \phi)\right), \quad (30)$$

differing only in the sign of the overall exponential factor, but this is an important distinction. The relation between the time derivatives $\dot{p}_{\mathrm{D}}(t)$ and $\dot{p}_{\mathrm{D,linear}}(t)$ is

$$\dot{p}_{\mathrm{D,linear}}(t) = e^{-\frac{b}{m}t}\left(\dot{p}_{\mathrm{D}}(t) - \frac{b}{m}p_{\mathrm{D}}(t)\right) \quad (31)$$

Equation 28 also allows the Hamiltonian of equation 29 to be written, somewhat contrary to the spirit of Hamiltonian mechanics but useful for elucidating relationships here, in terms of the position q and velocity $\dot{q}(t)$ as

$$H_{\mathrm{D}} = e^{\frac{b}{m}t}\left(\frac{1}{2}m\dot{q}^2 + \frac{1}{2}kq^2\right) = e^{\frac{b}{m}t}\,E_{\mathrm{D,tot}}. \quad (32)$$

In this case we see that the overall Hamiltonian is not equal to the total energy, due to the fact that there is an interaction which depends on velocity. However the portion in brackets is equal to the total non-dissipative (i.e kinetic + potential) energy of the oscillator, so the Hamiltonian is equal to this total non-dissipative energy modified by the exponential factor. This is the same exponential factor that relates the generalized momentum to the linear momentum (equation 30).

For the Hamiltonian NN including time as an input parameter and with the generalized momentum, the actual and predicted position $\dot{q}_{\mathrm{D}}(t)$ versus time, actual and predicted $\dot{p}_{\mathrm{D,linear}}(t)$ versus $\dot{q}_{\mathrm{D}}(t)$, and actual and predicted Hamiltonian vs time for the damped oscillator test data are shown in Figure 5. In this case the linear $\dot{p}_{\mathrm{D,linear}}(t)$ is recovered from the generalized $\dot{p}_{\mathrm{D}}(t)$ using equation 31. For the case with noise, see our implementation decisions in Appendix 6.3

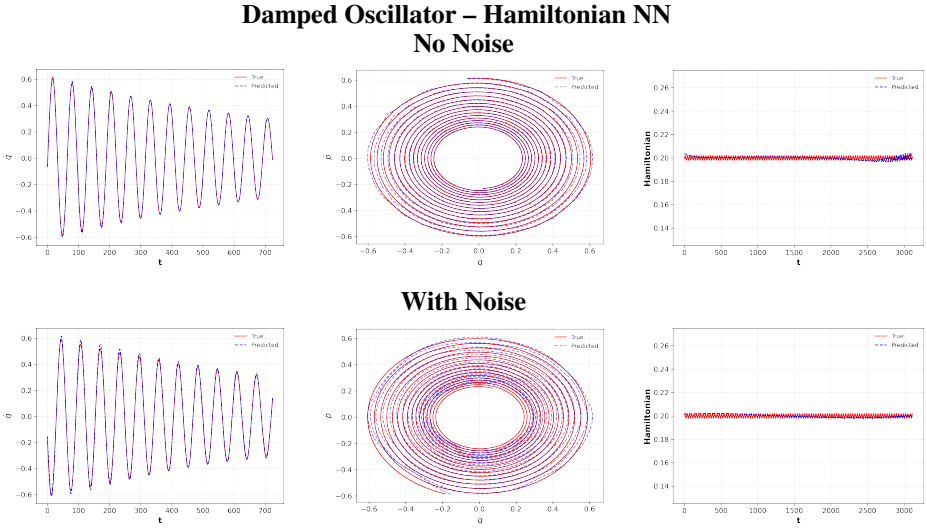


Figure 5: Actual and predicted velocity $\dot{q}_{\mathrm{D}}(t)$ versus time (**left**), actual and predicted *linear* $\dot{p}_{\mathrm{D,linear}}(t)$ versus $\dot{q}_{\mathrm{D}}(t)$ (**middle**), and actual and predicted H_{D} versus time (**right**) for the damped oscillator in the Hamiltonian NN, for the cases of no noise (**upper**) and full simulated noise (**lower**) in the "observed" data. The Hamiltonian NN uses the canonical momentum $p_{\mathrm{D}}(t)$ as input and provides the canonical $\dot{p}_{\mathrm{D}}(t)$. The conversion between $\dot{p}_{\mathrm{D,linear}}(t)$ and $\dot{p}_{\mathrm{D}}(t)$ is given by equation 31.

We also attempted to apply the Hamiltonian NN without time as an input parameter. It did not converge with the generalized momentum $p_{\mathrm{D}}(t)$ used as an input. However, with the *linear* momentum $p_{\mathrm{D,linear}}(t)$, we found that this did lead to convergence and the ability to produce a model that fits the observed $\dot{q}_{\mathrm{D}}(t)$ and $\dot{p}_{\mathrm{D,linear}}$. The results are shown in Figure 6.

4.3 Hamiltonian NN Results

We see that the Hamiltonian NN is able to successfully predict the dynamics of the system and its underlying Hamiltonian even for the damped case, when time is included as an input parameter for the latter. We note that the Hamiltonian of any system is insensitive to an overall additive constant which we have removed as appropriate from the predicted Hamiltonians. In the damped case, we see that without time as an input parameter and with the use of the linear momentum $p_{\mathrm{D,linear}}(t)$ instead of the generalized momentum $p_{\mathrm{D}}(t)$, the network actually learns an approximation of the *total non-dissipative (kinetic + potential) energy* $E_{\mathrm{D,tot}}$ of the system rather than the Hamiltonian. Because these are related by the same exponential factor (equation 32) as relates the generalized and linear momenta (equation 30), it can apparently recover the latter using the same differentiation process as a standard Hamiltonian NN.

5 Discussion

We see in the results of this work that the use of Lagrangian NNs and Hamiltonian NNs to predict the dynamical behavior of systems, as pioneered by (6) and (8), can be extended to systems in which the Lagrangian and Hamiltonian functions have explicit time dependence, where the Hamiltonian is not equal to the total energy, and where the total energy is not conserved – under certain conditions:

- A Lagrangian NN can be used to predict the dynamical behavior of a simple dissipative system if i) the time is provided as an input parameter *or* ii) when the explicit time dependence of the Lagrangian function of the system is such that $\frac{\partial L}{\partial t}$ is just a constant times the Lagrangian. We do note that in the latter case the resulting loss function (equation 21) essentially hard codes the known damping into the loss function, meaning that the LNN cannot be said to have independently modeled the damping of the system.
- A Hamiltonian NN can be used to predict the dynamical behavior of a simple dissipative system if i) the time is provided as an input parameter and the generalized momentum is used as the input momentum parameter *or* ii) if the relation between the generalized momentum and the linear momentum is the same as that between the total energy and the Hamiltonian, and the linear momentum is used as the input momentum parameter. However, a Hamiltonian NN is only able to predict the actual Hamiltonian function of the system in the former case, while in the latter case it predicts a function which is similar to the Hamiltonian but multiplied by a time-dependent exponential decay factor, which is in fact the total non-dissipative (kinetic + potential) energy of the system.

As can be seen in Figures 2 and 3, the Lagrangian NN models recovered *a* functional form for the Lagrangian of the systems but this is not the same functional form as those given by equations 14 and 18, even though these Lagrangian NN models successfully predicted the dynamics of the system (i.e. $\ddot{q}(t)$). This is apparently due to the property of Lagrangian mechanics that there can be multiple 'correct' Lagrangians for a given system. In particular, given a certain Lagrangian L, the equations of motion and therefore the overall dynamics of the system remain unchanged for $L \rightarrow L'$ if

$$L' = A' \left(L + \frac{dF}{dt} \right) + C \tag{33}$$

where $F = F(q_i)$ is a function of the generalized coordinates (10). Additionally the Lagrangian is insensitive to an overall multiplicative constant A' or overall additive constant C.

As a demonstration, we have visually fit $F(q)$ functions for the undamped and damped with time as an input parameter cases. Figure 7 shows the true (equations 14 and 18) and predicted Lagrangians for forms of equation 33 for these cases, with the functions and values for $F(q)$, A', and C given in Table 1. Comparing to the forms of the Lagrangians predicted by the Lagrangian NN models in Figures 2 and 3 we see that the Lagrangian NN models are indeed able to successfully recover *a* correct Lagrangian for the systems, even if it is not the simplest one. It is interesting to observe this property of Lagrangian mechanics made manifest. Why the NN models chose these seemingly more complicated system Lagrangian functions over the simplest ones of equations 14 and 18 is potentially a topic for future investigation.

References

[1] Simon S. Haykin. *Neural Networks*. Pearson Education, Upper Saddle River, NJ, third edition, 2009.

[2] Gabriel da Silva Macedo, Mariama Rebello de Sousa Dias, and Anibal Thiago Bezerra. Leveraging machine learning to harness non-parabolic effects in semiconductor heterostructures. *Physica E: Low-dimensional Systems and Nanostructures*, 146:115513, 2023.

[3] Jack Singal, Gerald Silverman, Evan Jones, Tuan Do, Bernadette Boscoe, and Yujie Wan. Machine Learning Classification to Identify Catastrophic Outlier Photometric Redshift Estimates. *ApJ*, 928(1):6, March 2022.

[4] Evan Jones, Tuan Do, Yun Qi Li, Kevin Alfaro, Jack Singal, and Bernie Boscoe. Redshift Prediction with Images for Cosmology Using a Bayesian Convolutional Neural Network with Conformal Predictions. *ApJ*, 974(2):159, October 2024.

[5] Maziar Raissi, Paris Perdikaris, and George E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[6] Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2019.

[7] Michael Lutter, Christian Ritter, and Jan Peters. Deep lagrangian networks: Using physics as model prior for deep learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[8] Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[9] Peter Toth, Danilo J. Rezende, Andrew Jaegle, Sébastien Racanière, Aleksandar Botev, and Irina Higgins. Hamiltonian generative networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[10] John R. Taylor. *Classical Mechanics*. University Science Books, 2005.

[11] Herbert Goldstein. *Classical Mechanics*. Addison-Wesley, 1980.

[12] Andrew Sosanya and Sam Greydanus. Dissipative hamiltonian neural networks: Learning dissipative and conservative dynamics separately, 2022.

[13] Veera Sundararaghavan, Megna N. Shah, and Jeff P. Simmons. Lagrangian neural networks for reversible dissipative evolution, 2024.

6 Appendix

6.1 Reproduction Statement

All the results in the paper are reproducible with the submitted source code. For notebooks that contain the variable `REPRODUCE_PAPER`, set it to `True` to load the saved reference models, and run the respective notebooks to reproduce the paper results. The damped Hamiltonian NN notebooks (Figure 5) retrain their models from a fixed random seed. We produced 7 by visual trial-and-error fitting.

6.2 Use of LLMs

We used LLMs to assist with composing and verifying plotting code, such as figure font settings or grid style. The real data was plotted: LLMs were only used to assist us with the code for plotting the experimental data. We manually reviewed all suggested changes. We didn't use the LLMs to develop the methodology, implement the neural networks, design the experiments, interpret the results, or formulate the paper's scientific claims.

6.3 Hamiltonian NN – Damped, Noise case

For the case with noise, the conversion from the observed linear momentum to the generalized momentum (equation 30), and correspondingly of $\dot{p}_{\mathrm{D,linear}}(t)$ to $\dot{p}_{\mathrm{D}}(t)$, multiplies the noise by $e^{\frac{b}{m}t}$, which reaches ~ 500 at the end of the training trajectories. With an $L2$ loss these late-time training points dominate the loss, and we find that the network fails to learn the dynamics. For this case we therefore use a SmoothL1 loss, a larger training set, and a fixed rescaling of the inputs applied inside the network (so that the in-network gradients with respect to the physical inputs are unaffected), as given in Table 2.

6.4 Figure 6 – Energy NN

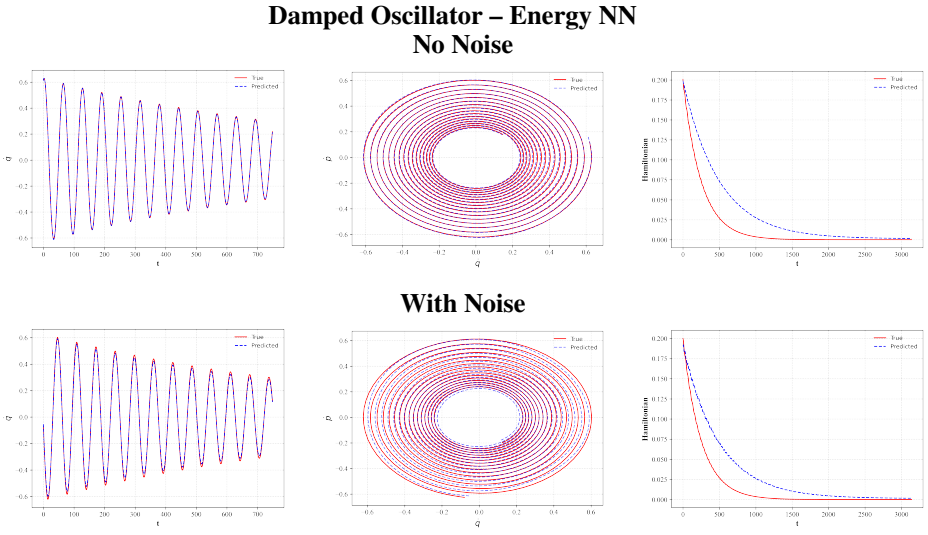


Figure 6: Actual and predicted velocity $\dot{q}_{\mathrm{D}}(t)$ versus time (**left**), actual and predicted *linear* $\dot{p}_{\mathrm{D,linear}}(t)$ versus $\dot{q}_{\mathrm{D}}(t)$ (**middle**), and actual and predicted non-dissipative energy $E_{\mathrm{D,tot}}$ versus time (**right**) for the damped oscillator in the Energy NN, for the cases of no noise (**upper**) and full simulated noise (**lower**) in the "observed" data. The Energy NN uses the linear momentum $p_{\mathrm{D,linear}}(t)$ as input and provides $\dot{p}_{\mathrm{D,linear}}(t)$ directly.

6.5 Figure 7 – L' Functions

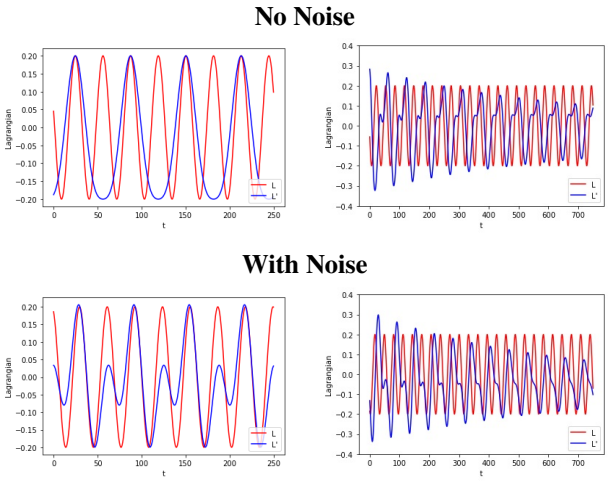


Figure 7: Calculated true default Lagrangian functions for the undamped (**left**) and damped with time as an input parameter (**right**) given by equations 14 and 18, respectively, along with the visually approximated alternate L' functions of the form of equation 33 for each case. The forms of the functions $F(q)$ and the values for the constants A' and C for the plotted L' functions are given in Table 1. As discussed in §5 these L' functions can be compared to the Lagrangians found by the LNN models in Figures 2 and 3 to see that the LNN models are able to successfully recover *a* correct Lagrangian for the systems, even if not the simplest default Lagrangian. We note that the damping cases may require additional higher order terms for a more precise fit to the recovered Lagrangian functions.

6.6 Tables

Table 1: LNN $F(q)$ functions for $L' = A'\left(L + \frac{dF}{dt}\right) + C$

	Undamped, No Noise	**Undamped, Noise**	**Damped, No Noise**	**Damped, Noise**
$F(q)$	$\frac{1.3\,\mathrm{N}}{\omega_0}\,q$	$\frac{0.13\,\mathrm{N}}{\omega_0}\,\mathrm{q} - \frac{0.09\,\mathrm{N\,s}}{\omega_0^2}\frac{dq}{dt}$	$\frac{0.9\,\mathrm{N}}{\omega_0}\,q + \frac{0.045\,\mathrm{N}}{\omega_0^2\,\mathrm{m}}q^2 + \frac{0.09\,\mathrm{N}}{\omega_0^2\,\mathrm{m}^3}q^4$	$\frac{-7.15\,\mathrm{N\,s}}{\omega_0^2}\frac{dq}{dt} - \frac{5.23\,\mathrm{N}}{\omega_0^2\,\mathrm{m}}q^2 + \frac{0.07\,\mathrm{N}}{\omega_0^2\,\mathrm{m}^3}q^4$
A'	0.15	0.62	-0.25	0.03
C	-0.035 J	0 J	0 J	-0.05 J

$q(t)$ functions are given by equations 16 and 20 for the undamped and damped cases, respectively. Figure 7 shows the default L from equation 12 and the L' functions given the values in this table, which can be compared with the predicted Lagrangians in Figures 2 and 3, as discussed in §5.

Table 2: Network Hyperparameters

LNN

Parameter	Undamped, Noise-free	Undamped, Noisy	Damped, Noise-free	Damped, Noisy
Data Generation				
num_trajectories	40	40	40	40
sample_every	100	10	10	10
noise_std	0	0.04	0	0.04
damping_coeff	0	0	0.02	0.02
savgol_window	3	3	3	7
savgol_polyorder	2	2	2	2
numerical_epsilon	1e-6	1e-6	1e-6	1e-6
input_scaling	MinMax $[0, 1]$	MinMax $[0, 1]$	MinMax $[0, 1]$	MinMax $[0, 1]$
Network Architecture				
hidden_layer_width	200	300	300	300
num_hidden_layers	4	8	8	10
dropout	0.1	0.1	–	–
initialization	Xavier	Xavier	Xavier	Xavier
Training Parameters				
learning_rate	1e-3	1e-3	1e-3	1e-3
epochs	50	100	100	100
batch_size	32	32	32	32
weight_decay	1e-5	1e-5	1e-5	1e-5
loss_function	SmoothL1	SmoothL1	SmoothL1	SmoothL1
lr_scheduler	ReduceLROnPlateau	ReduceLROnPlateau	ReduceLROnPlateau	ReduceLROnPlateau
gradient_clipping	1.0	1.0	1.0	1.0

HNN

	Hamiltonian NN				Energy NN	
Parameter	**Undamped, Noise-free**	**Undamped, Noisy**	**Damped, Noise-free**	**Damped, Noisy**	**Damped, Noise-free**	**Damped, Noisy**
Data Generation						
num_trajectories	30	30	100	1200	30	30
sample_every	100	10	10	2	100	10
noise_std	0	0.07	0	0.04	0	0.04
damping_coeff	0	0	0.02	0.02	0.02	0.02
inputs	(q, p)	(q, p)	(q, p_{D}, t)	(q, p_{D}, t)	$(q, p_{\mathrm{D,linear}})$	$(q, p_{\mathrm{D,linear}})$
input_scaling	–	–	$q/1.5,\ p/2,\ (t-157)/157$	$q/1.5,\ p/2,\ (t-157)/157$	–	–
Network Architecture						
hidden_layer_width	200	200	256	256	200	200
num_hidden_layers	4	6	5	5	4	6
Training Parameters						
learning_rate	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3
epochs	300	500	200	15	300	500
batch_size	full training set	full training set	256	2048	full training set	full training set
weight_decay	Default	Default	Default	Default	Default	Default
loss_function	MSE	MSE	SmoothL1	SmoothL1	MSE	SmoothL1
lr_scheduler	–	–	Cosine (1e-3 → 1e-5)	Cosine (1e-3 → 1e-5)	–	–
gradient_clipping	–	–	–	–	–	–

All models use the Adam optimizer and `tanh` activation functions. `hidden_layer_width` is the number of neurons in each hidden layer; `num_hidden_layers` is the total number of hidden layers between the input and the output. `noise_std` is the standard deviation of the Gaussian noise added to q (LNN) or to q and p (HNN). The LNN ReduceLROnPlateau scheduler uses factor 0.5 and patience 15. The damped LNN settings apply to models both with and without t as an input.