Hamiltonian Neural Networks Project

John Crossman
(Dated: September 21, 2024)

In this project, I replicate and extend some of the results of "Hamiltonian Neural Networks" by Greydanus, S. et al. [1], modifying the original code [2] to accomodate both multilayer perceptrons (MLPs) and Kolmogorov-Arnold Networks (KANs) [3], [4]. Specifically, I compare the performance of a KAN-based Hamiltonian Neural Network (HNN) against the original MLP-based HNN when predicting the dynamics of a spring-mass oscillator system. Ultimately, the KAN-based HNN outperformed the MLP-based HNN in all aspects. Additionally, the KAN-based HNN had the added benefit that it was able to be used to extract the exact symbolic solution of the Hamiltonian of the system, thus supporting the claim from the original KAN paper that one of the main benefits of KANs is their high level of interpretability [3].

I. INTRODUCTION

In "Hamiltonian Neural Networks" by Greydanus, S. et al. [1], the authors introduce a neural network architecture called a Hamiltonian Neural Network (HNN) to learn the dynamics of some physical system. Rather than learning the derivatives of position and momentum directly with a multilayer perceptron (MLP), an HNN first outputs a scalar quantity (the Hamiltonian of the system), and then computes the derivatives of position and momentum as the gradients of this scalar quantity according to Hamilton's equations [1]. In this way, the authors argue, the network learns to conserve the energy of the system, and this allows it to output more accurate calculations for the derivatives of these quantities than it would if it just learned the derivatives directly [1]. These derivatives can then be used to numerically predict the trajectory of the system [1]. The model learns purely from experimental measurements of position and momentum of the system and is robust to measurement noise [1]. This therefore has wide-ranging real world applications, all the way from robotics to even finetuning quantum computer gate operations.

In this project, I use pykan [4] to rewrite the code used in the HNN paper [2] to accommodate Kolmogorov-Arnold Networks (KANs). As argued in the original KAN paper [3], one of the central advantages of KANs is their interpretability. Elegant analytical solutions in scientific research applications are easily recovered using KANs. This is largely because, in contrast to MLPs, KANs learn activation functions directly, allowing for far more compact learned representations. In this project, I test these advantages by comparing the performance of a KAN-based HNN against the original MLP-based HNN used in [1] when predicting the dynamics of a spring-mass oscillator system.

A. Hamiltonians

The Hamiltonian of a system is the total energy of the system, expressed as the sum of kinetic and potential energies as functions of momentum and position, respectively:

$$H(q,p) = T(p) + V(q), \tag{1}$$

where q is a position variable, p is a momentum variable, T is the kinetic energy, and V is the potential energy. The equations of motion of a system can be derived from its Hamiltonian by using a set of differential equations called Hamilton's equations:

$$\frac{\mathrm{d}q}{\mathrm{d}t} = \frac{\partial H}{\partial p}, \tag{2}$$

$$\frac{\mathrm{d}p}{\mathrm{d}t} = -\frac{\partial H}{\partial q}. \tag{3}$$

Because equation (2) is just velocity, and equation (3) is just force, Newton's second law, $F = m\frac{\mathrm{d}^2q}{\mathrm{d}t^2}$, can be written using Hamilton's equations:

$$-\frac{\partial H}{\partial q} = m\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial H}{\partial p}\right). \tag{4}$$

Thus, we can obtain the equations of motion from the solution of this differential equation.

For this project, we are concerned with a simple spring-mass oscillator system. The Hamiltonian of this system is given by:

$$H = \frac{1}{2}kq^2 + \frac{p^2}{2m}, \tag{5}$$

where k is the spring constant, m the mass of the attached object, q the displacement from the spring's equilibrium position, and p the momentum of the attached object. Plugging this Hamiltonian into equations (2), (3), and (4), we obtain:

$$\frac{\partial H}{\partial p} = \frac{p}{m}, \tag{6}$$

$$-\frac{\partial H}{\partial q} = -kq, \tag{7}$$

$$-kq = m\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{p}{m}\right). \tag{8}$$

Remembering that $p = mv = m\frac{\mathrm{d}q}{\mathrm{d}t}$, we then obtain

$$-kq = m\frac{\mathrm{d}^2q}{\mathrm{d}t^2}. \tag{9}$$

The solution of the differential equation, (9), is of the form:

$$q(t) = A\cos(\omega t) + B\sin(\omega t), \tag{10}$$

where $\omega = \sqrt{k/m}$. Note that $q_0 = q(0) = A$, where q_0 is the initial position. Note also that $v_0 = \frac{\mathrm{d}q}{\mathrm{d}t}|_{t=0} = B\omega$, where v_0 is the initial velocity. We can then rewrite the position of the mass on the spring as a function of time as

$$q(t) = q_0\cos(\omega t) + \frac{v_0}{\omega}\sin(\omega t), \tag{11}$$

where q_0 and v_0 are the initial position and velocity of the system, respectively. Rewritten again in terms of position and momentum, this obviously becomes

$$q(t) = q_0\cos(\omega t) + \frac{p_0}{m\omega}\sin(\omega t), \tag{12}$$

where $p_0 = mv_0$ is the initial momentum.

The momentum of the system as a function of time is then trivial to derive:

$$p(t) = -m\omega q_0\sin(\omega t) + p_0\cos(\omega t). \tag{13}$$

If we set (in whatever consistent system of units you prefer) $k = 2$, $m = 0.5$, $p_0 = 0$, and $q_0 = 1$, we obtain

$$q(t) = \cos(2t), \tag{14}$$

$$p(t) = -\sin(2t) \tag{15}$$

as our equations of motion. Fig. 1 shows the phase space representation of the resulting trajectory, which can be described as a unit circle starting at the point (1, 0) and moving in a clockwise direction.

II. RESULTS

The data used to train and test the models were generated in largely the same way as in [1]. 50 random trajectories of the system were generated, and they were all sampled at 30 different points. For all trajectories, the spring constant k was fixed to $k = 2$, and mass m was fixed to $m = 0.5$. However, the initial position and momentum were allowed to vary randomly in each trajectory such that the total energy was a random number in the range [0.1, 1.0]. These 50 trajectories were randomly split into 25 training trajectories and 25 test trajectories. Note that noise was not added to the data points. All training was done in 20,000 epochs with the L-BFGS optimizer. Each training step used the entire training data set; no batching was used. The learning rate was $\alpha = 0.001$.

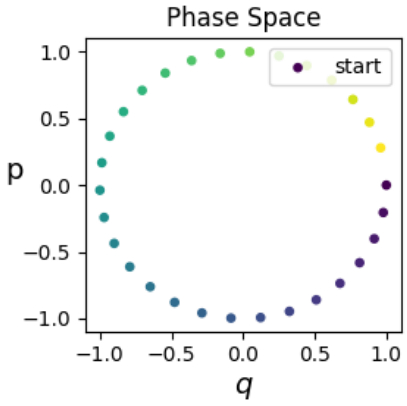

FIG. 1. Phase space trajectory of a spring-mass oscillator system for parameters $k = 2$, $m = 0.5$, $q_0 = 1$, $p_0 = 0$. Purple is the starting point, (1, 0).

The KAN had 2 input nodes. It had a single hidden layer with 1 node. It also had 2 output nodes. Note that because the HNN is learning a scalar value, only one output node is needed. However, one useless output was included to test the efficacy of an L1 regularization. All other variables of the KAN were kept as the default values as in pykan version 0.0.5 [4]. A graphical representation of one of the learned KANs is depicted in Fig. 2.

The MLP had an input dimension of 2. Though it also had an output dimension of 2, only one of these outputs was ultimately used as the scalar Hamiltonian output; the model was only given 2 outputs to maintain consistency with the KAN. There was only one hidden layer with a dimension of 200. The activation function was hyperbolic tangent.

Two different loss functions were tested: one with an L1 regularization, and one without an L1 regularization. Apart from the L1 regularization, the loss function used was the same as in [1]:

$$\mathcal{L} = \left\|\frac{\partial H_\theta}{\partial p} - \frac{\mathrm{d}q}{\mathrm{d}t}\right\|_2 + \left\|-\frac{\partial H_\theta}{\partial q} - \frac{\mathrm{d}p}{\mathrm{d}t}\right\|_2, \tag{16}$$

where H_θ is the model's Hamiltonian approximation, and $\frac{\mathrm{d}q}{\mathrm{d}t}$ and $\frac{\mathrm{d}p}{\mathrm{d}t}$ are the actual derivatives of position and momentum, respectively.

The L1 regularization used is given by

$$L_1 = \lambda\sum_{\theta_i\in H_\theta}|\theta_i|, \tag{17}$$

where the θ_i are the parameters of the model, H_θ, and λ is a constant fixed at $\lambda = 0.00005$.

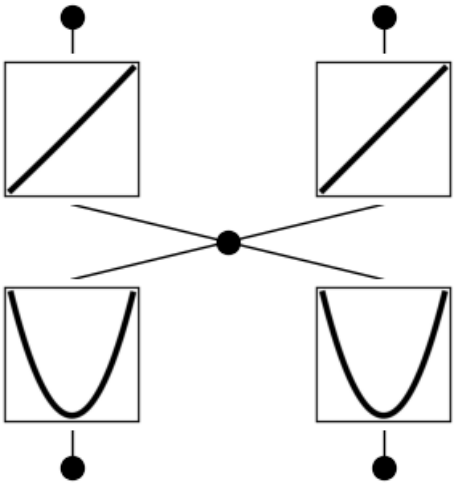

FIG. 2. KAN-HNN trained without L1 regularization. The top left node does not contribute to the Hamiltonian learned by the model. The position, q, and momentum, p, are fed into the bottom of the network. They then pass through a parabolic activation function, and finally to a linear activation function. This is exactly what we should expect the model to look like, as the exact Hamiltonian is $H = q^2 + p^2$. Indeed, the model fits to $H_\theta = q^2 + p^2 + 0.92$, where the constant offset, 0.92, is inconsequential under Hamilton's equations.

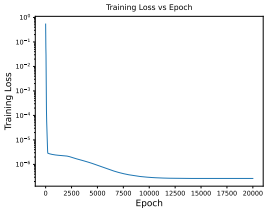

FIG. 3. Training loss of MLP-HNN without L1 regularization versus epoch. The final train loss was 2.6696e-07.

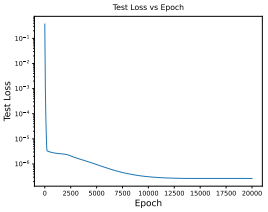

FIG. 4. Test loss of MLP-HNN without L1 regularization versus epoch. The final test loss was 2.6558e-07.

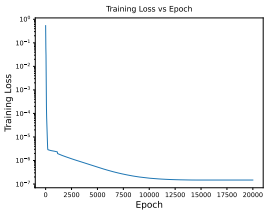

FIG. 5. Training loss of KAN-HNN without L1 regularization versus epoch. The final train loss was 1.4791e-07.

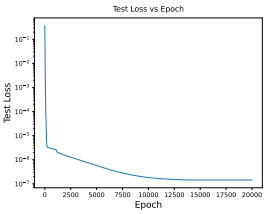

FIG. 6. Test loss of KAN-HNN without L1 regularization versus epoch. The final test loss was 1.4370e-07.

First, the MLP-HNN and the KAN-HNN were both trained using the loss function without the L1 regularization according to the aforementioned parameters and data generation method. Fig. 3 and Fig. 4 depict the training and test loss of the MLP-HNN versus epoch, respectively. The final train loss was 2.6696e-07, while the final test loss was 2.6558e-07. Fig. 5 and Fig. 6 depict the training and test loss of the KAN-HNN, respectively. The final train loss was 1.4791e-07, while the final test loss was 1.4370e-07.

These models were then used to predict the phase space trajectory and energy of the system over time starting from $q_0 = 1$, $p_0 = 0$ using the same numerical method used in [1]. Fig. 7 shows these trajectories, as well as the "baseline truth." This baseline truth was computed by calculating the derivatives of position and momentum via Hamilton's equations using the actual analytical Hamiltonian, and using the same numerical method as with the models [1]. Fig. 8 shows the mean squared error of the models' predictions of the phase space trajectory compared to the baseline truth's predicted trajectory. Fig. 9 shows the energy the HNNs predicted over the course of the predicted phase space trajectories, along with the baseline truth.

Fig. 2 depicts the HNN-KAN. Note that the inputs, q and p, at the bottom, are fed into a parabolic activation function. They then meet in the center node and are passed into linear activation functions. Upon calling the "symbolic_formula" function in pykan [4], we find that the center node results in $q^2 + p^2$, as the parabolic functions both fit to $y = x^2$. The linear activation function feeding into the top node on the right, which is the only one actually used in Hamiltonian calculations (remember, one node is useless) is fit to $y = x + 0.92$. The one feeding into the top left is fit to $y = 0.64x + 0.61$. But, remember, its output is not used. Thus, we obtain $H_\theta = q^2 + p^2 + 0.92$ as our Hamiltonian approximation, while the actual Hamiltonian is $H = q^2 + p^2$. Because the constant offset (0.92) does not change the dynamics predicted by Hamilton's equations, this expression is functionally exact for a spring-mass Hamiltonian with parameters $k = 2$ and $m = 0.5$. This symbolic formula can therefore be used to exactly match the baseline truth trajectory, thus giving the KAN-HNN a massive advantage over the MLP-HNN both in terms of accuracy and compactness. Note that the reason the plotted trajectories and energies do not exactly match the baseline is because the actual model was used, and not the fitted function.

Finally, the L1 regularization was tested. Using the same training process and parameters, the same dataset, and the same tests as described before, a KAN-based HNN was trained with the L1 regularization and tested against the same MLP-based HNN as before. Fig. 10 and Fig. 11 depict the training and test loss of the KAN-HNN versus epoch (without the L1 regularization added), respectively. The final train loss was 2.9114e-07, while the final test loss was 3.4841e-07. Fig. 12 depicts the L1 regularization penalty versus epoch.

Fig. 13 depicts the phase space trajectories of the original MLP-HNN, the new KAN-HNN, and the baseline

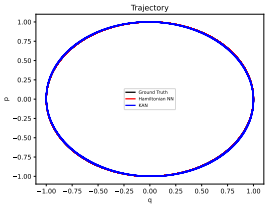

FIG. 7.
Phase space trajectory predictions of the MLP-HNN and KAN-HNN trained without L1 regularization, along with the baseline truth.

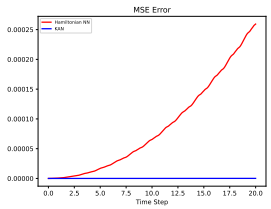

FIG. 8.
Mean squared error of the phase space trajectories predicted by the MLP-HNN and KAN-HNN trained without L1 regularization compared to the baseline truth versus time.

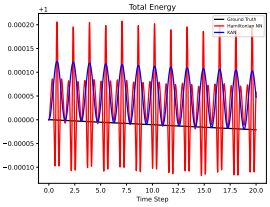

FIG. 9.
Energy of the predicted phase space trajectories as computed by the MLP-HNN and KAN-HNN trained without L1 regularization versus time. The baseline truth is also included.

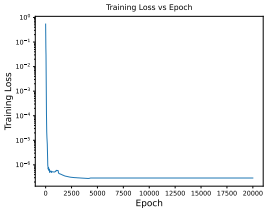

FIG. 10.
Training loss of KAN-HNN with L1 regularization versus epoch. The final train loss was 2.9114e-07.

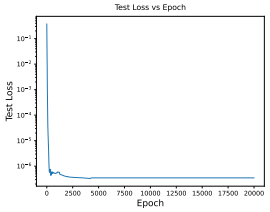

FIG. 11.
Test loss of KAN-HNN with L1 regularization versus epoch. The final test loss was 3.4841e-07.

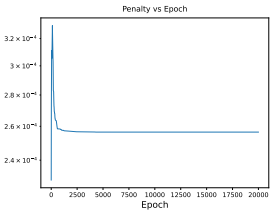

FIG. 12.
L1 regularization penalty of KAN-HNN versus epoch.

truth. Fig. 14 depicts the mean squared error of the model trajectories compared to the baseline truth. Fig. 15 shows the energy the HNNs predicted over the course of the predicted phase space trajectories, along with the baseline truth.

Upon calling the "symbolic_formula" function from pykan [4], the predicted Hamiltonian, as output by the top right node of the KAN, is fit to $H_\theta = q^2 + p^2 + 1.69$. Again, this is functionally exact. The inconsequential top left node is fit to 0. Thus, the L1 regularization has erased an inconsequential section of the KAN.

III. DISCUSSION

In conclusion, the KAN-HNN outperforms the MLP-HNN. For training without L1 regularization, the KAN obtains lower training and test losses, more accurate trajectory predictions, and better energy conservation. Additionally, the learned activation functions are easily interpreted and fit to a functionally exact Hamiltonian, and this fit can be used to derive exact equations of motion. When introducing L1 regularization to the KAN training, the KAN obtains larger training and test losses than an MLP trained without regularization. However, it still predicts the trajectory and energy more accurately than the MLP. Additionally, a functionally exact Hamiltonian fit is still recoverable.

A. Future Work

In the future, I plan to try using both MLPs and KANs to train a single HNN on multiple Hamiltonians. I am curious if, by training this network on many simple scenarios, it could easily learn more complex physical scenarios that are effectively combinations of the base scenarios. This would have interesting applications to robotics: a robot could be trained in a simulation on identifying many simple Hamiltonians, such as springs, pendulums, and free fall motion, and then, when it encounters a previously unseen and more complex scenario, it can quickly finetune its weights to handle it. For example, it could be argued that playing tennis or ping pong is a combination of these base scenarios: the ball is a spring, and the paddle in hand is a pendulum. Indeed, in the original paper [1], the authors do actually train an HNN on a video of a pendulum and it does accurately predict its motion, demonstrating that it is possible for these HNNs to learn using computer vision.

Of course, while combining simple scenarios can be straightforward, such as combining free fall motion with a simple spring-mass oscillator, it can also result in chaotic systems, such as the elastic pendulum, where the rod in

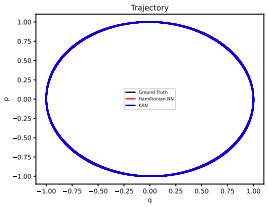

FIG. 13. Phase space trajectory predictions of the MLP-HNN trained without L1 regularization and the KAN-HNN trained with L1 regularization, along with the baseline truth.

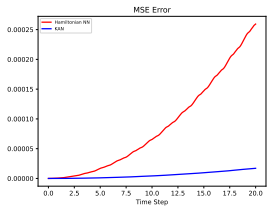

FIG. 14. Mean squared error of the phase space trajectories predicted by the MLP-HNN trained without L1 regularization and the KAN-HNN trained with L1 regularization compared to the baseline truth versus time.

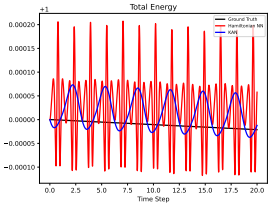

FIG. 15. Energy of the predicted phase space trajectories as computed by the MLP-HNN trained without L1 regularization and the KAN-HNN trained with L1 regularization versus time. The baseline truth is also included.

a pendulum is replaced with a spring. The authors in the original paper [1] do show that MLP-based HNNs perform well at predicting three-body problems, though it is unclear to me that using a KAN would be advantageous in these types of nonanalytical scenarios.

[1] S. Greydanus, M. Dzamba, and J. Yosinski, Hamiltonian neural networks (2019), arXiv:1906.01563 [cs.NE].
[2] S. Greydanus, M. Dzamba, and J. Yosinski, hamiltonian-nn, `https://github.com/greydanus/hamiltonian-nn.git` (2019).
[3] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Y. Hou, and M. Tegmark, Kan: Kolmogorov-arnold networks (2024), arXiv:2404.19756 [cs.LG].
[4] Z. Liu, pykan, `https://github.com/KindXiaoming/pykan.git` (2024).