# [Re] Hamiltonian Generative Networks

**Carles Balsells Rodas**
carlesbr@kth.se

**Oleguer Canal Anton**
oleguer@kth.se

**Federico Taschin**
taschin@kth.se

## Abstract

This work re-implements and further develops Hamiltonian Generative Networks (HGN) [13]. HGN proposes a method to learn system dynamics within the Hamiltonian framework directly from RGB videos. We review the approach taken by the original authors and compare our results with theirs. Finally, we both present novel experiments and propose a modification of the architecture that addresses flaws we detected in the original one. Results show that we can learn the Hamiltonian of simple systems, getting similar outcomes of those from the original authors. We provide the first open-source code implementation of the paper, which can be found in this repo. Furthermore, we claimed [13] in the Papers With Code Reproducibility Challenge 2020, where we will submit our reproduction.

## 1 Introduction

Consider an isolated physical system with multiple bodies interacting with each other. Let $q \in \mathbf{R}^n$ be the position of the bodies, and $p \in \mathbf{R}^n$ the momentum. **The Hamiltonian formalism** [2] states that there exists a function $\mathcal{H}(q,p) : \mathbf{R}^{n+n} \to \mathbf{R}$ representing the energy of the system which relates $q$ and $p$ as:

$$\frac{\partial q}{\partial t} = \frac{\partial \mathcal{H}}{\partial p}, \qquad \frac{\partial p}{\partial t} = -\frac{\partial \mathcal{H}}{\partial q} \tag{1}$$

In this work $\mathcal{H}$ is modeled with an artificial neural network and property 1 is exploited to get the temporal derivatives of both $q$ and $p$. One can then use a numerical integrator 4.3 to solve the ODE and infer the system evolution both forward and backward in time given some initial conditions (see figure 2). These initial conditions are inferred from a natural image sequence of the system evolution (see figure 1). The authors propose a generative approach to learn low-dimensional representations of the position and momenta $(p_0, q_0)$. This allows us to sample new initial conditions and unroll previously unseen system evolutions according to the learned Hamiltonian dynamics.

We re-implement the proposed architecture and attempted to learn the dynamics of a pendulum, a mass-spring, an n-gravitational body system, and a double pendulum. Figure 3 shows video rollout examples for each environment we implemented. Experiments rely on a training dataset created by randomly sampling some valid initial conditions of the studied system and unrolling a 30-frame video sequence of its evolution (see 4). Once the model is trained, we can employ it to predict the evolution of the system from the given sequence; both forward and/or backward in time. Our system can produce rollouts at different speeds by simply modifying the time step of the integrator.

We achieve comparable results to those of [13] (see 4.2) and provide further insights by testing the approach on new environments and with principled architectural changes (see 4.3).

## 2 Related Work

The approach taken by the original authors can be understood as a temporally extended variational autoencoder [8]: The system learns to reconstruct further time-steps of the input natural-image sequence a natural-image sequence while ensuring that the latent representation of the initial conditions $(p_0, q_0)$

follows a Gaussian probability distribution. This sequence is generated using the learned abstract representations of position and momenta using the Hamilton formulas [2]. This shares a lot of similarities to the RNN [11] paradigm, where the latent momenta would be the "hidden" state and the latent position, the output. In the loss optimization, instead of having a constant Lagrange multiplier in the equation as proposed by [4], we use the GECO algorithm [10] to dynamically balance the weight of the KL divergence and the reconstruction error as suggested in [13].

Finally, the work which most relates to HGN [13] is Hamiltonian Neural Networks (HNN) [3]. The main differences between the two works are that HNN is mainly designed to work on non-image based inputs and that the model assumes that the dimensionality of the abstract position and momenta is known, and that HNN does not provide a generative model.

## 3 Method

### 3.1 Hamiltonian Generative Network (HGN)

The HGN [13] architecture can be splitted into two parts: first, a model that extracts the initial abstract position and momenta $(p_0, q_0)$ provided an environment rollout (fig. 1), and second, a recurrent model which performs steps of a fixed $\Delta t$ and reconstructs the encoded latent position information (fig 2). As figures 1, 2 depict, this model is composed by four main networks:

- **Encoder**: Parametrized by: $\phi$. 8-layer 64-filter Conv2D network that takes a sampled video rollout from the environment and outputs the mean and variance of a normally distributed latent variable $z \sim q_\phi(z)$. This hidden variable $z$ is sampled using the reparametrization trick [8].
- **Transformer**: Parametrized by: $\psi$. Takes in the sampled hidden variable $z$ and transforms it into a lower-dimensional initial state $s_0 = (q_0, p_0)$, by applying 3 Conv2D layers with a stride of 2 and 64 filters.
- **Hamiltonian**: Parametrized by: $\gamma$. 6-layer 64-filter Conv2D network which takes in the abstract momenta and position at a certain time-step $(q_t, p_t)$ and outputs the energy of the system $e_t$. This network is used by the integrator 4.3 to compute the system state at the next time-step $(q_{t+1}, p_{t+1})$.
- **Decoder**: Parametrized by: $\theta$. 3-residual block upsampling Conv2D network (as in [6]) which converts each abstract position into an image close to the source domain.

Given an input sequence: $(x_0, ..., x_T)$ the loss function to optimize is:

$$\mathcal{L}(\phi, \psi, \gamma, \theta; x_0, ..., x_T) = \frac{1}{T+1} \sum_{t=0}^{T} \left[ E_{q_\phi(z|x_0,..x_T)} \left[ \log p_{\psi,\gamma,\theta}(x_t \mid q_t) \right] - KL(q_\phi(z) \mid\mid p(z)) \right] \quad (2)$$

We use the same optimizer as in [13]: Adam [7] with a constant learning rate of $lr = 1.5\text{e}{-4}$. In addition we use the GECO algorithm presented in [10] to adapt the Lagrange multiplier during training.

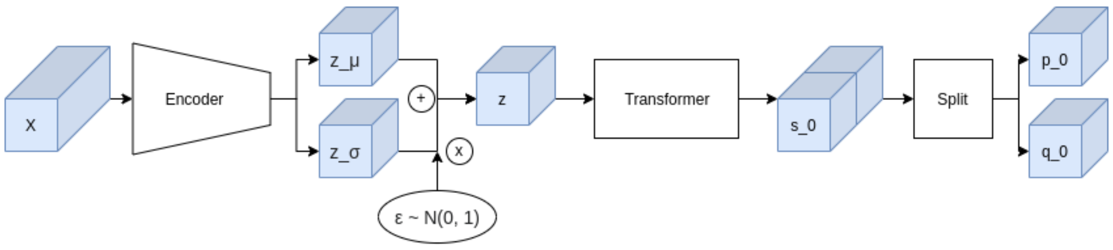


Figure 1: HGN network architecture to find initial abstract position and momentum $(q_0, p_0)$. Tensors are represented in blue and operations in black. A channel-axis concatenated rollout is inputed into the encoder which uses the reparameterization trick to sample the latent space variable $z$. The transformer converts this sample to the initial state, which is split into initial position and momentum by evenly dividing the tensor across the channels' dimension.

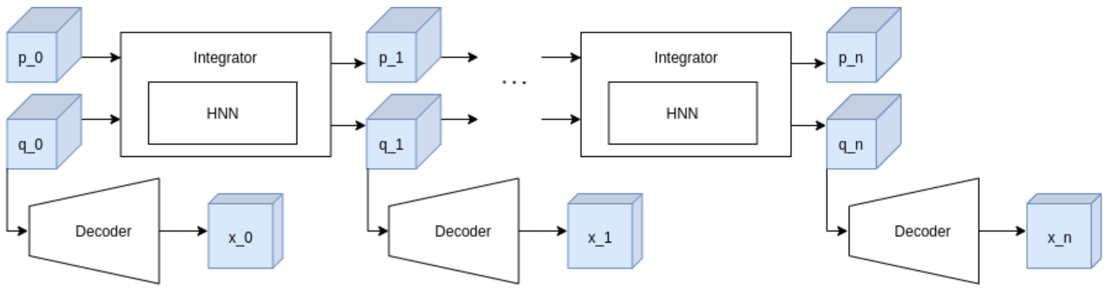


Figure 2: Recurrent part of the HGN architecture. Tensors are represented in blue and operations in black. The integrator takes the position and momentum for each time-step and provides the position and momentum of next time-step using the Hamiltonian Network. The decoder transforms the latent position information into the original input image in that particular time.

### 3.2 Neural Hamiltonian Flow

Taking inspiration from the integrator structure (see fig. 2), the authors present a normalizing flow model. Notice that a property of Hamiltonian dynamics is that they are bijective, easily invertible and volume-preserving (determinant of the Jacobian is 1). This makes them computationally efficient single flow steps. In this case, $q$ is treated as the random variable to model. $p$ becomes a helper latent variable using the trick presented in the Hamiltonian Monte Carlo (HMC) literature [1]. They use a Neural Network to infer $p$ from $q$ which they call *encoder*. We coded this model but found several training issues due to a lack of implementation details in the original paper. Therefore, we will not include results in this report but we will continue to work on this approach for the Papers With Code Reproducibility Challenge 2020.

### 3.3 Our contribution: Integrator Modelling

We wondered to what extent it was beneficial for the system to model the Hamiltonian and use its partial derivatives in the integrator. First, computing the partial derivatives is an expensive operation, as it requires to perform backpropagation trough the Hamiltonian Network. Second, our results and [13] show that the resulting Hamiltonian has high variance. We therefore wondered whether the fully-hamiltonian approach is actually helpful, and whether we could achieve the same goal without it. Therefore, we decided to test an architecture almost identical to the HGN, where the Hamiltonian Network is replaced by a CNN that directly computes $\Delta q$ and $\Delta p$ from $q_t$ and $p_t$. Integration is then performed as an Euler step: $q_{t+1} = q_t + \Delta t \Delta q$ and $p_{t+1} = p_t + \Delta t \Delta p$. In this architecture therefore we do not learn Hamiltonian-like dynamics anymore, but we directly learn the system dynamics in the abstract space. This approach achieves lower reconstruction loss than HGN[13], and is still able to operate at different time-scales using a different $\Delta t$. Results are presented in the additional experiments section 4.3.

## 4 Experiments

### 4.1 Data

The datasets considered by the original authors consist of observations of the time evolution of four physical systems: mass-spring, simple pendulum, and two-/three-body [13]. In addition, we introduce two new physical systems to experiment with: damped harmonic oscillator and double pendulum (see Fig. 3).

The procedure for data generation is analogous to the one used by [3]. Given a physical system, we first randomly sample an initial state $(\mathbf{q}, \mathbf{p})$ and generate a 30 step rollout following the Hamiltonian dynamics. Once the trajectory is obtained, we add Gaussian noise with standard deviation $\sigma = 0.1$ to each phase-space coordinate at each step and render 32x32 image observations. The objects in the systems are represented as circles and we use different colors to represent different objects. We generate 50000 train samples and 10000 test samples for each physical system. To sample the initial conditions $(\mathbf{q}, \mathbf{p})$, we first sample the total energy denoted as a radius $\mathbf{r}$ in the phase space and then $(\mathbf{q}, \mathbf{p})$ are sampled uniformly on the circle of radius $\mathbf{r}$.

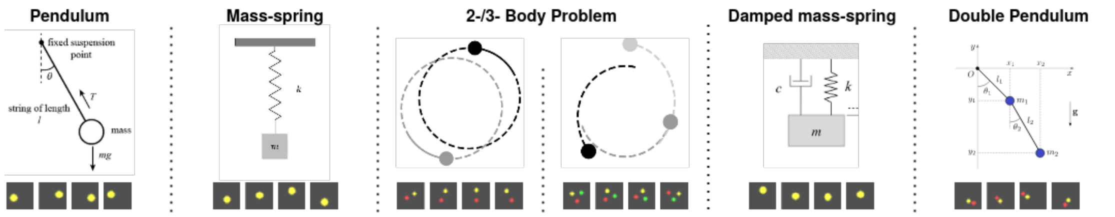


Figure 3: Representation and samples from the different physical systems considered in our experiments. Notice that differing from *Toth et al.* [13], we also consider a damped mass-spring system and a double pendulum.

**Mass-spring.** Assuming no friction, the Hamiltonian of a mass-spring system is $\mathcal{H} = \frac{p^2}{2m} + \frac{1}{2}kq^2$, where $m$ is the object's mass and $k$ is the spring's elastic constant. We generate our data considering $m = 0.5$, $k = 2$ and $r \sim \mathbb{U}(0.1, 1.0)$.

**Pendulum.** An ideal pendulum is modelled by the Hamiltonian $\mathcal{H} = \frac{p^2}{2ml^2} + 2mgl(1 - \cos q)$, where $l$ is the length of the pendulum and $g$ is the gravity acceleration. The data is generated considering $m = 0.5$, $l = 1$, $g = 3$ and $r \sim \mathbb{U}(1.3, 2.3)$.

**Two-/three- body problem.** The n-body problem considers the gravitational interaction between $n$ bodies in space. Its Hamiltonian is $\mathcal{H} = \sum_i^n \frac{||\mathbf{p}_i||^2}{2m_i} - \sum_{i \neq j}^n \frac{gm_im_j}{||\mathbf{q}_i - \mathbf{q}_j||}$, where $m_i$ corresponds to the mass of object $i$. In this dataset, we set $\{m_i = 1\}_{i=1}^n$ and $g = 1$. For the two-body problem, we modify the observation noise to $\sigma = 0.05$ and set $r \sim \mathbb{U}(0.5, 1.5)$. We set $\sigma = 0.2$ and $r \sim \mathbb{U}(0.9, 1.2)$.

**Dobule pendulum** The double pendulum consists of a system where we attach a simple pendulum to the end of another simple pendulum. For simplicity, we conider both simple pendulums with identical properties (equal mass and length). The Hamiltonian of this system is $\mathcal{H} = \frac{1}{2ml^2}\frac{p_1^2 + p_2^2 + 2p_1p_2cos(q_1 - q_2)}{1 + sin^2(q_1 - q_2)} + mgl\Big(3 - 2\cos q_1 - \cos q_2\Big)$, where $\{q_1, p_1\}$ and $\{q_2, p_2\}$ refer to the phase state of the first and second pendulum respectively. Our data is generated by setting $m = 1$, $l = 1$, $g = 3$ and $r \sim \mathbb{U}(0.5, 1.3)$. In this scenario we consider a very low intense source of noise $\sigma = 0.05$.

**Damped oscillator** The damped mass-spring system is obtained by considering a dissipative term in the equations of motion of the ideal mass-spring system. For such systems, one can obtain its dynamics using the Caldirola-Kanai Hamiltonian $\mathcal{H} = e^{\gamma t}\Big(\frac{p^2}{2m} + \frac{1}{2}kq^2\Big)$ [12], where $\gamma$ is the damping factor of the oscillator. In our experiments, we consider an underdamped harmonic oscillator and set $\gamma = 0.3$, $m = 0.5$, $k = 2$, $r \sim \mathbb{U}(0.75, 1.4)$ and $\sigma = 0.1$.

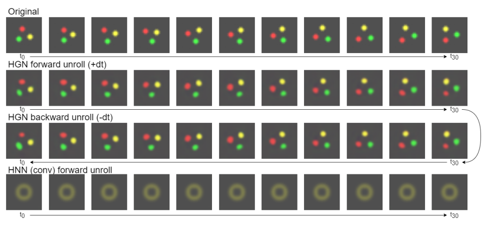
### 4.2 Reproduction

We first tested whether the HGN [13] and HNN [3] can learn the dynamics of the four presented physical systems by measuring the average mean squared error (MSE) of the pixel reconstructions. In addition, we test the original HGN architecture along with different modifications: a version trained with Euler integration rather than Leapfrog integration (HGN Euler), a version trained with no transformer network (HGN no $f_\psi$), and a version that does not include sampling from the posterior $q_\phi(\mathbf{z}|\mathbf{x}_0...\mathbf{x}_T)$ (HGN determ). To test HNN, we use the implementation provided by *Greydanus et al.*[3] known as pixelHNN. Similarly to [13], we test HNN with a modification of the architecture to closely match the HGN (HNN Conv).

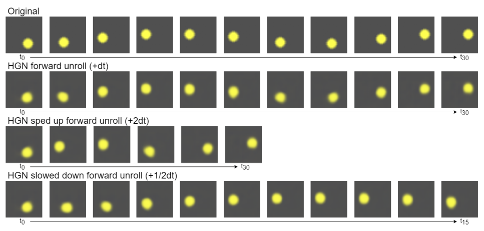
Table 1 shows the results of the experiment described previously along with the results of the original authors. As we can see, we achieved average pixel reconstruction errors that are similar to the ones reported by the original authors. However, our re-implementation is not able to achieve the same performance as the original work. This might be for several reasons that will be discussed in Sec. 6. In general, we can observe that our HGN and the proposed modifications learned well on the four physical systems. As we can observe, the results reported by both versions of the HNN are one order of magnitude higher than

| Model | Mass-spring |  | Pendulum |  | Two-body |  | Three-body |  |
|---|---|---|---|---|---|---|---|---|
|  | Train | Test | Train | Test | Train | Test | Train | Test |
| Orig. HGN (Euler) [13] | $3.67 \pm 1.09$ | $6.2 \pm 2.69$ | $5.43 \pm 2.53$ | $10.93 \pm 4.32$ | $6.62 \pm 3.93$ | $15.06 \pm 7.01$ | $7.51 \pm 3.49$ | $9.4 \pm 3.92$ |
| Orig. HGN (Determ) [13] | $0.23 \pm 0.23$ | $3.07 \pm 1.06$ | $0.79 \pm 1.24$ | $10.68 \pm 3.19$ | $2.34 \pm 2.3$ | $14.47 \pm 5.24$ | $4.1 \pm 2.05$ | $5.17 \pm 1.96$ |
| Orig. HGN (No $f_\psi$) [13] | $4.95 \pm 1.71$ | $7.04 \pm 2.55$ | $6.83 \pm 3.29$ | $13.98 \pm 4.94$ | $6.35 \pm 3.86$ | $16.49 \pm 6.6$ | $8.37 \pm 3.13$ | $10.41 \pm 3.72$ |
| Orig. HGN (Leapfrog) [13] | $3.84 \pm 1.07$ | $6.23 \pm 2.03$ | $4.9 \pm 1.86$ | $11.72 \pm 4.14$ | $6.36 \pm 3.29$ | $16.47 \pm 7.15$ | $7.88 \pm 3.55$ | $9.8 \pm 3.72$ |
| HNN | 69.62 | 69.61 | 125.12 | 125.07 | 48.50 | 48.53 | 57.77 | 57.79 |
| HNN (conv) | 69.25 | 69.25 | 125.96 | 125.93 | 48.53 | 48.55 | 57.72 | 57.74 |
| HGN (Euler) ours | 14.98 | 14.96 | 17.65 | 17.73 | 7.44 | 7.44 | 12.21 | 12.19 |
| HGN (Determ) ours | 10.13 | 10.12 | 17.91 | 17.96 | 4.67 | 4.68 | 6.75 | 6.76 |
| HGN (No $f_\psi$) ours | 12.62 | 12.60 | 15.57 | 15.59 | 8.62 | 8.63 | 8.78 | 8.78 |
| HGN (Leapfrog) ours | 11.17 | 11.18 | 18.78 | 18.84 | 6.31 | 6.31 | 9.16 | 9.18 |

Table 1: Average pixel MSE of the reconstruction of a 30-frame rollout sequence on the test and train datasets of the four physical systems presented by [13]. All the values are multiplied by $10^4$. We show our results (third section) along with the ones reported by the original authors (first and second sections).



(a)


Original
$t_0$
$t_{30}$
HGN forward unroll (+dt)
HGN sped up forward unroll (+2dt)
HGN slowed down forward unroll (+1/2dt)
$t_{15}$

(b)

Figure 4: (a) Reconstruction of a sequence of the 3-body system along with a backward unroll of the data from the final state; and a forward rollout of the HNN (conv). (b) Reconstruction of a sequence of the pendulum system along with a sped up and a slowed down forward rollout.

the four versions of the HGN. Visual inspections of the results provided by HNN show that the network simply learned to output a static image.

In Figure 4, we show some qualitative examples of the reconstructions obtained by the full version of HGN. The model can reconstruct the samples with high fidelity and its rollouts can be reversed in time, sped up, or slowed down by changing the value of the time step used in the integrator. Since the HGN is designed as a latent variable model, we can sample from the latent space to produce initial conditions and perform their time evolution. We show some rollouts obtained by sampling the latent space of the model in figure 5. We observe that our model is only able to generate plausible and diverse samples in the *mass-spring* dataset. This behavior is different than the one shown by [13] and might be caused by different hyper-parameter configurations in the training procedure. Later discussion with the authors revealed that they did not input the whole rollout to the encoder but just the first five frames, which was not explained in the paper. Since we received the information too late, we could not run all the experiments again. However, we tried this modification in the *mass-spring* dataset, achieving a reconstruction loss of $1.4 \times 10^{-3}$, which is very similar to authors results.

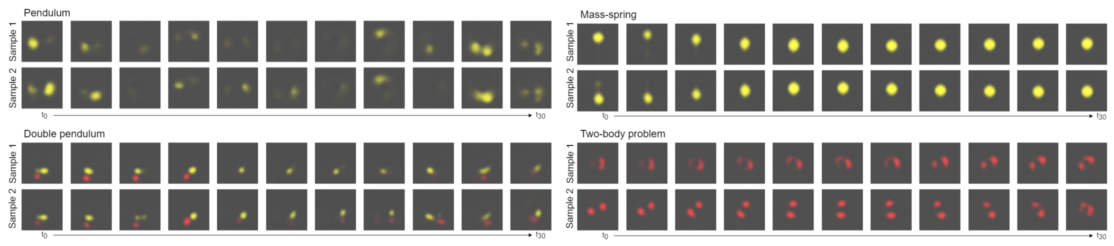


Figure 5: Examples of sample rollouts from the latent space for different physical systems.

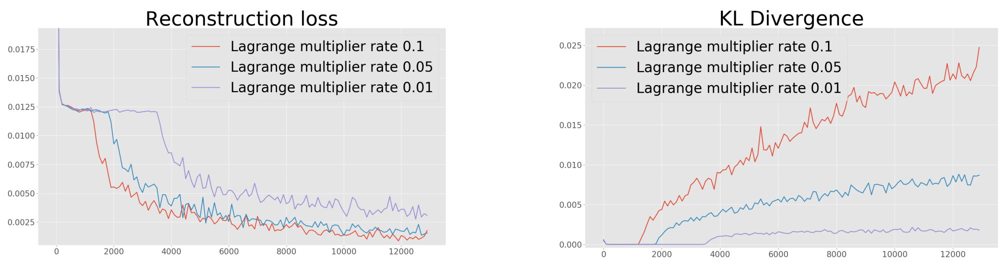


Figure 6: Reconstruction loss and KL divergence for different GECO parameters in the Pendulum environment.

| | EULER | RUNGE-KUTTA 4 | LEAPFROG | YOSHIDA |
|---|---|---|---|---|
| pixel MSE | 17.73 | 34.38 | 18.78 | 16.07 |
| $\mathcal{H}$ std | 23.09 | 4.69 | 10292 | 6049 |
| reconstr. time (s) | 0.28 | 1.61 | 0.71 | 1.26 |

Table 2: Comparison between four different integrators used to perform the time evolution in the HGN. The results are measured on the simple pendulum test set. The pixel MSE values have been multiplied by $10^4$.

### 4.3 Additional experiments

**Geco parameter search** The paper does not provide the values of GECO [10] used. In GECO the Lagrangian multiplier is optimized at each step with a rate $\gamma$. Figure 6 shows the behavior of GECO for $\gamma \in \{0.1, 0.05, 0.01\}$ in terms of reconstruction loss and KL divergence. Higher values of $\beta$ give a better reconstruction loss but greatly increase the KL divergence. In our experiments we generally use $\beta = 0.05$, which provides a good trade-off, or $\beta = 0.1$ for environments in which reconstruction is harder (e.g. pendulum).

**Integrators** Performing the integration step is key to generate the time evolution of a rollout given the initial state. In the HGN paper [13] the system is tested using Euler and Leapfrog integration. We wonder if using higher order integration methods might boost the performance of the rollout generation process. Therefore, we implement and test the HGN architecture with two additional numerical integration methods: the Runge-Kutta's 4th-order integrator [5] and the 4th-order Leapfrog integrator (Yoshida's algorithm [14]). Table 2 shows a comparison of all four integrators on the Pendulum dataset. Both Leapfrog and Yoshida are *symplectic* integrators: they guarantee to preserve the special form of the Hamiltonian over time [9].

Table 2 shows the average pixel MSE, the averaged standard deviation of the output of the Hamiltonian network during testing, and the reconstruction time of a single batch ($\texttt{batch} = 16$) using the different integration methods that we have described previously. The model has been trained on the simple pendulum dataset. As we can see, the reconstruction time increases when using higher-order integration methods, since they require more integration steps. In general, we see that Euler integration offers a fast and sufficiently reliable reconstruction of the rollouts. Moreover, we observe that the fourth-order symplectic integrator (Yoshida) achieves the best performance. Surprisingly, the symplectic integration methods show more variance in the output of the Hamiltonian networks throughout a single rollout. This behavior is unexpected since using a symplectic integration method should ideally keep the value of the Hamiltonian invariant. We conclude that more experiments need to be performed to guarantee that the implementation of both Leapfrog and Yoshida integration methods are faithful to their formulation.

**Integrator modelling** We trained the modified architecture of Section 3.3 on the Pendulum dataset for 5 epochs. The architecture is the same as HGN, but the Hamiltonian Network now outputs $\Delta q$ and $\Delta p$. The average MSE error over the whole Pendulum dataset is $1.485 \times 10^{-3}$, while in the test set it is $1.493 \times 10^{-3}$, which are both lower than the previous four versions of HGN (see Table 1). The modified architecture is still capable of performing forward slow-motion rollouts by modifying $\Delta t$. We set $\Delta t' = \frac{\Delta t}{2}$ and we compute the average MSE of the slow-motion reconstruction over 100 rollouts. The

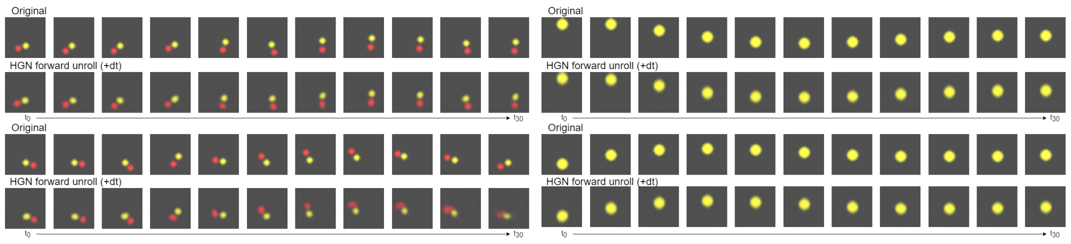


Figure 7: Examples of reconstructions of the double pendulum (left) and the damped harmonic oscillator (right).

modified architecture achieved an error of $8\text{x}10^{-4}$, while the standard HGN achieved $9\text{x}10^{-4}$. Note that reconstruction losses are smaller for slow-motion as the images change less between timesteps.

**Extra environments** Apart from the four physical systems presented by [13] we decided to test our re-implementation of the HGN with physical systems that do not have a simple Hamiltonian expression. As described previously, these are the damped harmonic oscillator and the double pendulum. On one hand, we are interested in a damped system since it introduces a dissipative term to the equations of motion; a feature that differs from the previous systems. On the other hand, the double pendulum is modelled by a non separable Hamiltonian: $\mathcal{H}(\mathbf{q}, \mathbf{p}) \neq K(\mathbf{p}) + V(\mathbf{q})$ as described previously. In figure 7 we show some visual examples of the reconstructions provided by the HGN trained on the two systems. As we can see, HGN is able to reconstruct the damped oscillator with high reliability. Regarding the double pendulum, we observe that the model reconstructs well small oscillations, but fails when the trajectory is too chaotic. The average pixel MSE of the reconstructions of the damped oscillator and the double pendulum are 0.0011 and 0.0012 respectively. The HGN is able to provide better reconstructions for these systems in comparison to the mass-spring and simple pendulum systems.

## 5 Challenges

The main challenge we encountered is finding the correct tools to debug a model composed of so many interconnected networks. The fact that it has a variational component with a dynamic Lagrange multiplier term makes it especially tricky to train. Furthermore, no public implementation existed and some details and parameters were missing in the original paper leading to some necessary assumptions or parameter searches.

## 6 Conclusions

We were able to design and train a Hamiltonian Generative Network with similar performance of the one of the original paper. Moreover, we provided further experiments giving a better grasp of the possibilities the Hamiltonian paradigm opens when using ANNs to learn about dynamical systems. Future work could include further testing on each network architecture, probably smaller networks would also be able to encode the needed information. Another clear next step is to try the approach on more challenging (and realistic-looking) environments. In addition, it would be interesting to tackle the transfer learning capabilities of such architecture between different environments. How re-usable each network is? How much faster the system is able to learn the new dynamics? Finally, another field which could benefit from this research is model-based reinforcement learning. A generative approach from which to sample example rollouts can be very useful for training agents without the need of directly interacting with the environment.

## 7 Ethical consideration, societal impact, alignment with UN SDG targets

One could ask until what point this research is helping anyone in any way. Specially considering that training these models leaves a carbon footprint and it is unlikely they get used in any practical setup. Nevertheless, we consider it to be a necessary investment to help us understand similar networks which can lead to more meaningful contributions.

## 8 Self Assessment

We believe our project qualifies for an A for the following reasons:

- Our repo is the first and only publicly available implementation of the studied paper.
- The implementation presents high technical difficulty which makes debugging challenging: We are training a variational model on several interconnected components, that involve partial derivatives w.r.t. latent variables.
- We test not only the HGN, but also the HNN architecture using their official code, therefore reproducing a meaningful comparison of the two methods. Moreover, we test HGN on two additional environments that add non-trivial complexity to the task (energy dissipation and non separable hamiltonian).
- We implement and test two additional integration methods, and we compare them with those used in the paper.
- We perform parameter search for parameters not specified in the paper.
- We propose a modification of the architecture based on extensive analysis of the results as well as the pros and cons of the Hamiltonian framework.

## Acknowledgements

We thank Stathi Fotiadis for voluntarily contributing with a GECO [10] implementation draft to the public repo and his useful feedback on code structuring. In addition, we would like to thank the original authors for providing further details on the implementation.

## References

[1] Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. *Handbook of Markov Chain Monte Carlo*. CRC press, 2011.

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

[3] Sam Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks, 2019.

[4] I. Higgins, Loïc Matthey, A. Pal, C. Burgess, Xavier Glorot, M. Botvinick, S. Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.

[5] TE Hull, WH Enright, BM Fellen, and AE Sedgwick. Comparing numerical methods for ordinary differential equations. *SIAM Journal on Numerical Analysis*, 9(4):603–637, 1972.

[6] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2018.

[7] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[8] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2014.

[9] Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

[10] Danilo Jimenez Rezende and Fabio Viola. Taming vaes, 2018.

[11] D. E. Rumelhart and J. L. McClelland. *Learning Internal Representations by Error Propagation*, pages 318–362. 1987.

[12] Francis Segovia-Chaves. The one-dimensional harmonic oscillator damped with caldirola-kanai hamiltonian. *Revista mexicana de física E*, 64(1):47–51, 2018.

[13] Peter Toth, Danilo Jimenez Rezende, Andrew Jaegle, Sébastien Racanière, Aleksandar Botev, and Irina Higgins. Hamiltonian generative networks, 2020.

[14] Haruo Yoshida. Symplectic integrators for hamiltonian systems: basic theory. In *Symposium-International Astronomical Union*, volume 152, pages 407–411. Cambridge University Press, 1992.