Hamiltonian Operator Disentanglement of Content and Motion in Image Sequences

Asif Khan 1 Amos Storkey 1

Abstract

We introduce a deep generative model for image sequences that reliably factorise the latent space into content and motion variables. To model the diverse dynamics, we split the motion space into subspaces and introduce a unique Hamiltonian operator for each subspace. The Hamiltonian formulation provides reversible dynamics that constrain the evolution of the motion path along the low-dimensional manifold and conserves learnt invariant properties. The explicit split of the motion space decomposes the Hamiltonian into symmetry groups and gives long-term separability of the dynamics. This split also means we can learn content representations that are easy to interpret and control. We demonstrate the utility of our model by swapping the motion of two videos, generating long term sequences of various actions from a given image, unconditional sequence generation and image rotations.

1. Introduction

The ability to learn to generate artificial image sequences has diverse uses, from animation, key frame generation, summarisation to restoration and has been explored in previous work over many decades (Hogg, 1983; Hurri and Hyvärinen, 2003; Cremers and Yuille, 2003; Storkey and Williams, 2003; Kannan et al., 2005). However, learning to generate arbitrary sequences is not enough; to provide useful value, the user must be able to control aspects of the sequence generation, such as the motion being enacted, or the characteristics of the agent doing an action. To enable this, we must learn to decompose image sequences into content and motion characteristics such that we can apply learnt motions to new objects or vary the types of motions being applied.

Deep generative models (DGMs) such as variational autoencoders (VAEs) (Kingma and Welling, 2013) and Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) use neural networks (NNs) to transform the samples from a prior distribution over lower-dimensional latent factors to samples from the data distribution itself. Recent developments (Chung et al., 2015; Srivastava et al., 2015; Hsu et al., 2017; Yingzhen and Mandt, 2018) extend VAEs to sequences using Recurrent Neural Networks (RNNs) on the representation of temporal frames. Similar approaches have been taken for GAN models (Tulyakov et al., 2018; Yoon et al., 2019; Dandi et al., 2020).

The dynamical processes creating the evolution of image sequences are highly constrained. Consider the simplistic case of a person walking in a scene with a camera moving around that individual. The walking pose will return to similar positions periodically, and likewise, the revolving camera will revisit previous positions. Even without strict periodicity, many dynamical processes are reversible. Any time that a dynamic could conceivably return to an earlier state suggests an implicit conservation law—the conservation of information in the underlying scene generator—as it must be capable of returning to and regenerating the same scene with non-negligible probability. The critical observation we wish to capture in this paper is that understanding the conservation occurring in the context of a set of sequences is a vital ingredient to decomposing content from motion.

Given any conserved quantity, any motion must be modelled in a way that maintains the conserved quantity. In physics, such a motion is called a Hamiltonian motion; it keeps the corresponding Hamiltonian function constant. Hence we argue that a flexible latent Hamiltonian model provides a good inductive bias to learn a representation space that enables conservation of the right quantities (which are themselves learnt) and models the dynamic evolution. This is mathematically equivalent to saying that we can learn to represent the underlying motions as combinations of differentiable symmetry groups; all differentiable symmetry transformations follow a conservation law (Noether, 1918).

The existing sequential DGMs do not impose any structural prior to constrain the dynamics in motion space, and therefore, accumulate errors as the length of the sequence grows, quickly deviating from the relevant path (Karl et al., 2016;
Hamiltonian Operator Disentanglement of Content and Motion in Image Sequences

Fraccaro et al., 2017; Yildiz et al., 2019; Bird and Williams, 2019. The attractive property of Hamiltonian dynamics is that they are symplectic that is, the divergence of a vector field is zero, and the evolving dynamics preserves the infinitesimal volume element. Consequently, the motion paths are restricted to a low-dimensional manifold in the latent space, and the dynamics can be predicted forward and backwards.

To empirically demonstrate the benefits of a Hamiltonian model we refer to Figure 1. Here, we consider a data set of \((32, 32)\) image sequences of a ball rotating in a fixed orbit; the \((i, j)\) centre of a ball moves under constraint \(i^2 + j^2 = c\), where \(c\) is a constant. The dynamics of any such sequence takes the form of an \(S^1\) manifold embedded in 1024 dimensional space. We generated \(N\) sequences of equal number of time steps with different initial condition and trained a VAE with a Hamiltonian operator in latent space. That is, we map sequences to 2-dimensional phase space, then use a linear learnable Hamiltonian operator to unroll the dynamics in the phase space and map the unrolled trajectory back to the data space\(^1\). The left side of Figure shows a pair of an input and a reconstructed sequence (held out from training) and three sequences generated from random initial conditions. On the right, the coordinates in phase space are coloured by the Hamiltonian energy, plus the demonstration of a random sequence that conserves the energy. The constant energy reveals the relationship between latent components and introduces the regular structure in the latent space proving critical for generating novel manifold consistent sequences. Therefore, identifying symmetries is a suitable inductive bias for developing expressive DGMs that understand the motion constraints and generalise beyond the training data. (Higgins et al., 2018; Toth et al., 2019; Botev et al., 2021) have discussed the benefits of such inductive biases for learning disentangled representation.

In this paper, we intimate the more general applicability of latent Hamiltonian models; previous applications have been limited to fairly constrained physical systems. We propose a VAE framework to model the dynamics of sequences using a collection of Hamiltonian operators. Specifically, for any motion sequence, we model the transition from time step \(t\) to step \(t+1\) using a group action of a Hamiltonian operator. The evolution of the dynamics of a sequence leaves certain information unchanged, identified as content, and certain properties that evolve in conjunction (i.e. motion). Since the space of image sequences comprises of diverse actions, we split the motion space into subspaces where each subspace models a unique action and is unaffected by other actions. This paper focuses on a discrete, identified set of actions that we can then compose at generation time. We want to remark our method can also work without action labels; we provide the details in the appendix. The benefit of identified action is we can use it for a controlled generation. Our main contributions are as follows,

1. We propose learnable Hamiltonian operators which associate conserved quantities with latent dynamics. In contrast to existing Hamiltonian approaches, usually restricted to constrained physical systems, we extend the formulation to more natural image sequences. Furthermore, the explicit form of operator imposes the structure in the motion latent space. In this way, we simultaneously learn a representation space along with symmetry transformations that act on the space.
2. To model diverse motions, we introduce a subspace for each action that allows the separability of dynamics. It further reduces the computational cost since the Hamiltonian of the space is now in a block diagonal form where each block is a Hamiltonian of a symmetry subgroup.
3. We empirically demonstrate the benefits of our model through i) generation of diverse dynamics from a starting frame, ii) demonstrating successful disentanglement of the content and motion representation. Furthermore, each motion subspace is associated with a single action allowing us to perform controlled generation.

2. Related Work

Hamiltonian Neural Networks In recent years, several deep learning (DL) methods have been proposed to learn the...
Hamiltonian Operator Disentanglement of Content and Motion in Image Sequences

dynamics of physical systems using Hamiltonian mechanics. Gerydanus et al. (2019) use NNs to predict Hamiltonian from phase-space coordinates \( s = (p, q) \) and their derivatives. In similar work (Bondesan and Lamacraft, 2019) used NNs to discover symmetries of Hamiltonian mechanical systems. More recently, Hamiltonian NNs are used for simulating complex physical systems (Sanchez-Gonzalez et al., 2019; 2020). The key idea of this work is to represent the states of particles as a graph and use a graph neural network (GNN) to predict the change from the current state to the next state. In a follow-up Cranmer et al. (2020), introduce sparsity on the messages in a graph and use the symbolic regression method to search for physical laws that describe the messages in the graph. Recently Toth et al. (2019) developed the Hamiltonian generative network (HGN), where they proposed to learn a Hamiltonian from image sequences. HGN maps a sequence to a latent representation and then projects it to the phase space to unroll the dynamics using a symplectic ODE integrator with Hamilton’s equation. In another work Yildiz et al. (2019) use second-order ODE parameterised as a BNN for modelling dynamics of high dimensional sequence data in the latent space of VAE. Most of the developments are built on the neural ODE (Chen et al., 2018) an idea to view layers of NNs as internal states of an ODE. These methods rely on the numerical integration scheme and the stability of the ODE solver. A Hamiltonian formalism dictates an additional requirement that the dynamics of an ODE should be volume-preserving and reversible.

Latent Space Models There is a long history of latent state space models for modelling sequences (Kalman, 1960; Starner and Pentland, 1997; Roweis and Ghahramani, 1999; Elliott and Krishnamurthy, 1999; Pavlovic et al., 2000). Much recently, these methods are combined with deep generative models for generating high dimensional sequences as well as learning a disentangled representation (Karl et al., 2016; Villegas et al., 2017; Tulyakov et al., 2018; Hsieh et al., 2018; Yingzhen and Mandt, 2018; Miladinovic et al., 2019; Minderer et al., 2019; Franceschi et al., 2020; Zhu et al., 2020). MoCoGAN (Tulyakov et al., 2018) developed an adversarial framework, combining a random content noise with a sequence of random motion noise to generate videos. More recently, DSVAE (Yingzhen and Mandt, 2018) proposed to split a latent space into time-variant and invariant representation and use LSTM ( Hochreiter and Schmidhuber, 1997) to model the prior on time-variant representation. S3VAE (Zhu et al., 2020) improves the disentanglement of DSVAE by minimising a mutual information loss between content and motion variables. Some Hamiltonian methods (Toth et al., 2019; Yildiz et al., 2019) also model the dynamics of high dimensional sequential data in a latent space. However, the focus in those cases is only on sequence generation, and to our knowledge, this has not been investigated for disentanglement.

Group Transformations in Latent Space Models Rao and Ruderman (1999) proposed the algorithm to model the infinitesimal movement on data manifold using learnable Lie group operators. Culpepper and Olshausen (2009), use the matrix exponents to learn the transport operators for modelling the manifold trajectory. Many other similar methods have investigated the use of geometric operators for learning the manifold representation from data (Rao and Ruderman, 1999; Culpepper and Olshausen, 2009; Memisevic, 2012; Sohl-Dickstein et al., 2010; Cohen and Welling, 2014). The use of symmetries for learning disentangled factor of variations has recently been considered. A disentanglement is generally identified as learning representations with independent latent factors. The main goal is that each latent factor should control a distinct data factor, and a single latent variable should control no two data factors. Various authors (Bengio et al., 2013; Lake et al., 2017; Eastwood and Williams, 2018). Higgins et al. (2018) have proposed a symmetry-based definition of disentanglement. The goal in these settings was to decompose a latent space into subspaces, and on each subspace, to learn a unique group transformation such that the subspace is unchanged by the action of other groups. Caselles-Dupré et al. (2019), build such a model using interaction with the environment. Some other similar approaches were recently proposed to learn group transformations in a latent space (Connor and Rozell, 2020; Quessard et al., 2020; Dupont et al., 2020). However, the applications were restricted to relatively toy problems and to our knowledge has not been investigated on higher dimensional videos.

3. Method

We propose a deep latent variable model for sets of sequential image data. Each set of sequences depicts the temporal evolution associated with one of a number of actions. In this context, an action is simply a label associated with a particular sequence set, but where it is understood the sequences within a set may have very different content, but the same form of dynamic. E.g. in the sprites data (discussed later) the actions are ‘walking’, ‘spell cast’ and ‘slash’ and the sequences within a set are different individuals performing the relevant action. In the following we assume the separation into actions sets are known, but that assumption is relaxed later.

Let \( x_{i,t} \) denote the \( i \)th image sequence, with \( x_{i,t} \) the \( t \)th image in the sequence. Let \( u_k \) be an indicator vector denoting the action associated with the \( i \)th sequence; i.e. \( u^i_k = 1 \) iff sequence \( i \) follows action \( k \) and \( u^i_{k'} = 0 \) for all other \( k' \neq k \). These sequences and corresponding actions are collected into a dataset \( \{(x_{i,t}, u_k)\}_{i=1}^N \) of size \( N \), where, for the sake of simplicity in description, we assume they all are of same length \( T \). In this paper, we use a latent space to
Hamiltonian Operator Disentanglement of Content and Motion in Image Sequences

Figure 2. The framework for our model. We first encode each time step of a sequence to a respective feature vector $h_{k:t}$. Next, to unroll the dynamics of an action $k$, we map to the phase space. Specifically, we sample a starting index $t$ and map $h_k$ to position coordinate $q_k^t$. For momentum $p_k^t$, we use temporal convolution with a kernel size $w$ on $h_{k:t−w}$. We then use the operator $H_k$ to trace out the forward and backward trajectory. At last we combine the position of all timesteps $q_{1:T}$ with the content $z$ and pass through the decoder network to generate the sequence.

aid modelling of each sequence and decompose that latent space into two parts, which we call a content space (denoted by $Z$) and a motion space (denoted by $S$). As the data comprises sequences of various actions that take different dynamical form, we further decompose the latent motion space $S = S^1 \oplus S^2 \oplus \ldots \oplus S^K$, with one part for each action. In modelling a sequence corresponding to action $k$, only the subspace $S^k$ will be allowed to change across the length of that sequence. Each motion subspace is further decomposed into generalised position and momentum parts: $S^k = (p^k, q^k)$. Only the position part of this latent space is used generatively to create an individual image. The momentum part only affects the dynamics.

Critical to this work, this decomposition makes it straightforward to define learnable Hamiltonian mechanics to model the dynamic process. This Hamiltonian model provides many advantages; it prevents the neural network from leaking constant content information via the motion representation, and it ensures the possibility of preserving key conservation quantities that we argue are implicit in the constraints of most motion dynamics. This is discussed further in Section 3.1.1. The full framework of our model is illustrated in Figure 2. In the next section, the generative model will be introduced, followed by the variational formalism for inference and the learning.

3.1. Generative Model

For completeness we first present the full probabilistic model in (1-4) before describing each component. Each dynamic is categorised by a particular action enumerated by $k$, encoded in an indicator vector $u$ (i.e. $u_k = 1$ for action $k$). The generative model is conditioned on this action vector. First, in (1), we sample the content variable $z$ from a prior $p(z)$. The content variable will describe the constant appearance characteristics expressed through the whole sequence. Next we sample a starting position from prior $p(q_1^0)$ (we initialise the actions not represented in the sequence to zero). The full state-space representation for the dynamic of action $k$ is then given by $s^k_t = (p^k_t, q^k_t)$. The dynamical model (3,6) then traces out the forward trajectory in the phase space. Finally, we combine the position trajectory with the content representation and use a decoder neural network to get the emission distribution of the data space sequence. In summary,

GIVEN: $k$ denoting action label for a sequence,

$$z \sim p(z), \quad q^k_1 \sim \mathcal{N}(0, I_d), \quad p^k_1 \sim \mathcal{N}(0, I_d), \quad (1)$$

$$s^k_0 = [p^k_0, q^k_0], \quad s^k_0 = 0, \quad \forall k' \neq k, \quad (2)$$

$$s^k_t = f(s^k_{t−1}; \omega_k, t), \quad s^k_{t−1} = s^k_{t−1}, \quad \forall t > 1, k' \neq k \quad (3)$$

$$q_t = [q_1^t, \ldots, q^K_t], \quad x_t \sim \mathcal{N}(x_t; \phi(z, q_t), \alpha^2 I_m), \quad \forall t \quad (4)$$

where $d$ is the dimensionality of $k^{th}$ subspace, $m$ is the dimensionality of data space, $f$ is a dynamical model (6) and $\omega_k$ are the parameters of $f$ to be used for the $k^{th}$ subspace. In this paper we use an emission distribution that is a spherical Gaussian, with a parameterised mean $\phi(\cdot, \cdot)$, and a spherical covariance $\alpha^2 I_m$. We have provided a probabilistic graph of the generative model in appendix Figure 5.
3.1.1. Dynamical Model

In image sequences, we can view each frame of a sequence as a point in some representation space; the temporal dynamics trace a path connecting the frames forming a 1-submanifold of the image manifold. Most dynamical models either try to capture this structure deterministically (Srivastava et al., 2015) or probabilistically (Chung et al., 2015; Hsu et al., 2017; Yingzhen and Mandt, 2018) via linear or non-linear state-space models. In either case, small errors in dynamical steps can accumulate and result in significant deviation from the manifold when unrolling long-term trajectories at inference time (Karl et al., 2016; Fraccaro et al., 2017). Interestingly, Hamiltonian systems alleviate some of these issues by constraining the dynamics to be reversible and volume-preserving. Reversibility is critical as it helps us understand how various actions could change the state of an underlying object in the world. By reversing the arrow of time, the object could return to its previous state, this awareness gives a sense of accountability to an object for its actions. In our work, without significant loss of generality, we propose a linear Hamiltonian system in the latent layer, relying on the nonlinear neural network mapping to data space to handle all nonlinear aspects. This also enhances the interpretability of the dynamics.

Definition 1. A matrix $H \in \mathbb{R}^{2d \times 2d}$ is an Hamiltonian matrix if $H^T J H = J$, where $J$ is a skew-symmetric matrix $J = \begin{pmatrix} 0 & I_d \\ -I_d & 0 \end{pmatrix}$ and $I_d$ is an identity matrix.

Consider a coordinate vector $s \in \mathbb{R}^{2d}$ in the phase space $S$ that evolves under constant Hamiltonian energy $E$,

$$ E = \frac{1}{2} s^T M(t) s $$

where $M(t)$ is a symmetric matrix and $s$ is a coordinate in phase space at a time $t$. In Hamiltonian mechanics, the coordinates are specified in terms of position $q$ and momentum $p$ variables as $s = (q, p)$. Using the fact energy $E$ is constant over time we can express equation of motion as,\[ \frac{d s(t)}{d t} = J M(t) s. \] Let $H(t) = J M(t)$, we can rewrite the equation of motion as,\[ \frac{d s(t)}{d t} = H(t) s. \]

We can obtain the closed-form solution of the above system using matrix exponential $s(t) = e^{tH} s(0)$. The matrix exponent has a connection to Lie algebras, and for small $t$ we can interpret $e^{tH}$ as an infinitesimal transformation of $s(0)$ under the action of a Lie group of $H$. We discuss this further in Appendix B. For a detailed introduction to the topic we refer to (Chevalley, 2016).

In this work, we consider $K$ Hamiltonians $H_1, \ldots, H_K$, each acting on separate parts of the phase space $S^1, S^2, \ldots, S^K$. To unroll the trajectory of motion $k$, we use the group action defined by the matrix exponent of the operator $H_k$ on a starting phase space representation $s^k_0 \in S^k$ given by,

$$ s^k_t = f(s^k_{t-1}; \omega_k, t) = e^{tH_k} s^k_{t-1}, \forall t > 1 \quad (6) $$

$$ s^k_t = 0, \quad \forall t, k' \neq k. \quad (7) $$

The backward dynamics can simply be obtained by negating time; i.e. replacing $t$ with $-t$ in the above. We assume all time steps are equally spaced. The above formulation provides an explicit disentanglement of the motion space. It further allows us to parallelise the computation of matrix exponential by leveraging the block diagonal form of $H$.

Specific to our work, we parameterise a symmetric matrix $M_k$ and obtain its Hamiltonian matrix as $H_k = J M_k$. By restricting the parameters of $M_k$ we can make use of different structure of Hamiltonians. We consider the group of real Hamiltonian matrices that form a Symplectic Lie group under multiplication $Sp(2d)$ with $2d^2 + d$ independent elements. We also look at the symplectic orthogonal group $SpO(2d)$ that further restricts Hamiltonians to skew-symmetric form with $(d^2 - d)/2$ independent elements. We briefly introduce the details in Appendix B. For a more comprehensive overview, we refer readers to Easton (1993).

3.2. Inference

In order to learn the model parameters, we need to infer the distribution over latent variables; we follow a variational formalism that provides following evidence lower bound (ELBO),

$$ \max_{q} \mathbb{E}_{q(x,t;\tau|x,t;u)} \log \frac{p(x_{1:T}, z, s_{1:T}|u)}{q(z|x_1:T|x_{1:T}, u)} \quad (8) $$

where $q(\cdot | \cdot)$ is the approximate posterior distribution and $s_t = [q_t, p_t]$. It remains to define the approximate posterior we use. Since the Hamiltonian dynamics are reversible, at inference time, we randomly sample a choice of frame $t$ and use forward and backward action to trace the trajectory of states after and before that frame. For a sequence $x_{1:T}$, we use the process in (9)-(12) to draw samples from a variational distribution $q(z, s_{1:T}|x_{1:T}, u)$. Simply, we sample the content variable $z$ conditioned on the observed data and independently sample the motion states $s^k_t = [q^k_t, p^k_t]$ for the reference frame $t$ conditioned on the observed data and the relevant action $k$. Motion states corresponding to other actions are set to zero. The motion states for all the other frames are then created from the forward and backward application of the Hamiltonian of the relevant action. In equations, this is,

$$ z \sim q(z|x_{1:T}), \quad t \sim U(\{1, \ldots, T\}) \quad (9) $$

$$ q^k_t \sim q(q^k_t|x_t, u), \quad p^k_t \sim q(p^k_t|x_{t:w}, u), \quad s^k_t = [q^k_t, p^k_t] \quad (10) $$

$$ s^k_{t+1} = f(s^k_t; \omega_k, t), \quad s^k_{t-1} = f(s^k_t; \omega_k, -t), \quad \forall t \quad (11) $$

$$ s^k_t = 0, \quad \forall t, k' \neq k. \quad (12) $$
Hamiltonian Operator Disentanglement of Content and Motion in Image Sequences

where \( t \) is a starting index, \( q(q^k_t|x_t, u) \) is the posterior distribution of \( k^{th} \) position subspace conditioned on the frame \( x_t \) and action variable \( u \), \( q(p^k_t|x_{t+w}, u) \) is the posterior distributions of \( k^{th} \) momentum subspace conditioned on \( w \) previous frames and action variable \( u \) and \( q(z|x_{1:T}) \) is the posterior distribution of the content space conditioned on the entire sequence. We parameterise the factorised posterior as a spherical Gaussian distribution,

\[
q(z|x_{1:T}) = \mathcal{N}(z|\mu_z, \sigma_z^2I), \tag{13}
\]
\[
q(q^k_t|x_t, u) = \mathcal{N}(q^k_t|\mu_{q^k_t}, \sigma_{q^k_t}^2I), \tag{14}
\]
\[
q(p^k_t|x_{t+w}, u) = \mathcal{N}(p^k_t|\mu_{p^k_t}, \sigma_{p^k_t}^2I), \tag{15}
\]

where we parameterise (13), (14) and (15) as content, position and momentum encoder neural network. We use reparameterisation trick (Kingma and Welling, 2013) to sample from latent distribution \( z = \mu + \sigma \odot \epsilon \) where \( \epsilon \sim \mathcal{N}(0, I) \).

3.3. Learning Objective

The final learning problem reduces to the optimisation of the following objective function,

\[
\max -KL[q(q^k_t|x_t, u)||p(q^k_t)] - KL[q(p^k_t|x_{t+w}, u)||p(p^k_t)]
\]
\[
-KL[q(z|x_{1:T})||p(z)] \mathbb{E}_{q(q^k_t|x_t, u)} \left[ \sum \log p(x_t|q^k_t, z) \right].
\]

We have provided the derivation in the Appendix A.

4. Experiments

We conduct experiments on the following video datasets, i) Sprites, a sequence of animated character performing different actions as per sprites.\(^2\) It comprises three actions: ‘walking’, ‘spell cast’ and ‘slashing’ from three viewing angles: ‘left’, ‘right’ and ‘straight’. The sequences are of length 8 with each frame as an RGB image of size 64 \times 64 \times 3. The appearance of each character has four attributes: colour of skin, hairstyle, tops and pants. Each attribute can take six values resulting in 1296 unique characters. We used 1000 characters for training and the rest for evaluation. ii) MUG (Aifanti et al., 2010), which is a dataset of 52 individuals performing six facial expressions: anger, disgust, fear, happiness, sadness and surprise. The dataset is made available by signing the license agreement available. The dataset consists of sequences of variable length ranging from 50 to 160 frames. For training purpose, we downsample the sequences by a factor of two and use random subsequences of length 8, crop the face region and resize it to 64 \times 64. The training and evaluation splits are based on (Tulyakov et al., 2018). We also demonstrate the benefit of our model in predicting rotations on MNIST digits. The symplectic structure allows us to generate long term trajectories in the latent space. For results and discussion on rotating MNIST we refer readers to Appendix C.5.

4.1. Results and Discussion

We perform the qualitative and quantitative analysis of all the models. We investigate two choices of structure for the Hamiltonian matrices. We refer to the symplectic group structure as \( H \) and the symplectic orthogonal group as skew-\( H \). To compute a matrix exponential, we use fast Taylor approximation (Bader et al., 2019) that provides a stable solution under various matrix norms.

Quantitative Evaluation We evaluate our model on sequence generation as well as the disentanglement of representations. As a first step we evaluate the generation capability of the two operators \( H \) and skew-\( H \) using the per-frame structural similarity index measure (SSIM) peak signal to noise ratio (PSNR) and mean squared error (MSE). We generate a longer sequence of length 16 (twice the length used for training purpose) conditioned on the starting frame. The sprites consists of periodic sequences of length 8 where the start and end frames are identical, in this case we duplicate the sequence to get a ground truth of length 16. For MUG, we draw sequence of length 16 from the evaluation set. The SSIM scores are between \(-1\) and \(1\), with a more significant score indicating more similarity between the ground truth and generated sequence. Likewise, higher PSNR and lower MSE implies better generation. Table 1 describes the performance of our model on generation task under different scores. The scores show our model can generate high-quality sequences from an input image. We observe \( H \) performs better than skew-\( H \) at sequence generation. The skew-\( H \) are interpretable because they restrict Hamiltonian to rotation matrix. We hypothesise the superior performance of \( H \) can be attributed to the fact that skew-\( H \) introduces additional restrictions on the parameters of \( H \) that might be reducing the expressiveness of the model. This is an interesting finding. The question we wish to investigate it in the future work. For further discussion and analysis we only consider \( H \). We compare its performance with the state-of-the-art baselines for sequence disentanglement, namely DSVAE (Yingzhen and Mandt, 2018), MoCoGAN (Tulyakov et al., 2018) and S3VAE (Zhu et al., 2020).

To evaluate the disentanglement, we use a classifier pretrained on the task of action prediction, to evaluate the generated sequences. To begin with, we draw a starting position and momentum from a prior distribution and use a dynamical model to unroll the trajectory in the phase space. Next, we sample the content variable \( z \) from real sequences and combine it with position variables to generate the sequence. We report the performance of the classifier in predicting the action from these generated sequences. This score gives us a measure of how well a model can keep the

\[^3\]https://github.com/jrconway3/Universal-LPC-spritesheet
Hamiltonian Operator Disentanglement of Content and Motion in Image Sequences

Table 1. Evaluation for sequence generation. For SSIM and PSNR higher is better; for MSE lower is better. We can see both choices of operator can generate sequences close to the ground truth.

| Method | Data | Accuracy\(\uparrow\) | H\(y|x)\(\downarrow\) | H\(y)\(\uparrow\) |
|--------|------|----------------------|-------------------|-------------------|
| Ours H | MUG  | 0.992                | 0.108             | 1.778             |
| Ours H (unconditional) | MUG  | 0.750                | 0.187             | 1.762             |
| DSVAE (Yingzhen and Mandt, 2018) | MUG  | 0.543                | 0.374             | 1.657             |
| MoCoGAN (Tulyakov et al., 2018) | S3VAE (Zhu et al., 2020) | 0.631                | 0.183             | 1.721             |
| S3VAE (Zhu et al., 2020) | Sprites | 0.705                | 0.135             | 1.760             |
| Ours H | Sprites | 0.994                | 0.011             | 2.009             |
| DSVAE (Yingzhen and Mandt, 2018) | Sprites | 0.907                | 0.072             | 2.192             |
| MoCoGAN (Tulyakov et al., 2018) | Sprites | 0.928                | 0.090             | 2.192             |
| S3VAE (Zhu et al., 2020) | Sprites | 0.994                | 0.041             | 2.197             |

Table 2. Results of classifier on MUG and sprites data. The high score of accuracy and Inter-Entropy H\(y|x)\) while low scores of Intra-Entropy H\(y|x)\) are expected from a better model. Our model performs best across all three scores on MUG. On sprites we are comparable to S3VAE. This is due to simplicity of classes in sprites.

We want to remark that our formulation is not constrained by using action variables \(u\). Table 2 also describes the results for an unconditional version on the MUG (Aifanti et al., 2010). For details on the method and a more comprehensive discussion we refer readers to Appendix C.2. The benefit of incorporating action variables \(u\) is that our model can be used for the controlled generation of sequences, as demonstrated in Figure 4. We further evaluate our model in preserving the identity of sequences. The identity for sprites is described in terms of four different attributes, and for MUG (Aifanti et al., 2010) it refers to the identity of the person. We pre-train a classifier on the task of identity prediction and use it for evaluating the generated sequences. This evaluation gives us a measure of the model’s ability to keep the identity when the motion is changed. For sprites, we report the accuracy of individual attributes. Results are outlined in Table 3. We can see on MUG our model can preserve the identity with high accuracy. We can make a similar observation for different attributes of sprites sequences. Thus, the good performance indicates that the content is preserved when traversing the motion subspace, and the motion space is invariant when changing the content variables. This is also reflected in the qualitative results.

Qualitative Evaluation For the qualitative analysis, we report results with the Hamiltonian \(H\). We observed the skew-\(H\) resulted in similar performance; we omit it here due to limited space. To begin with, Figure 3a gives an example of original, reconstruction and generated sequences. We generate a sequence by applying the motion operator on the latent encoding of the first time step. On the left are the results for sprites and on the right of the MUG video sequences. To evaluate disentanglement, we use the model for the task of motion swapping. We start by encoding two sequences \(x^{1:T}_{1}, x^{1:T}_{2}\) to their latent representations \((z^{1}, q^{1}_{1:T}), (z^{2}, q^{2}_{1:T})\) and \((z^{1}, q^{1}_{1:T}), (z^{2}, q^{2}_{1:T})\) between the two representation spaces, and then pass the resulting representations through the decoder to generate the sequences \(x^{1:T}_{1}\) and \(x^{1:T}_{2}\). Figure 3b shows the result of this. On the left, the pair of consecutive rows are of original sequences and on the right of the sequences generated by swapping the motion representations. We can see swapping the motion part does not affect the identity of the sequences. To further investigate the generation quality of different motion operators, we use our model for an image to sequence generation. We first encode the image to its latent space representation. Next, we obtain its representation in the different motion spaces and use the respective operators to unroll the trajectories in phase space for each different motion, which are then combined with content and transformed to the image space using a decoder network. The left side of Figure 4 shows examples of decoding different motion from the same input image. For sprites, the actions are in order ‘walk’, ‘spell card’, ‘slash’ and for MUG they are ordered ‘anger’, ‘disgust’, ‘fear’, ‘happiness’, ‘anger’, ‘disgust’, ‘fear’, ‘happiness’, ‘anger’, ‘disgust’, ‘fear’, ‘happiness’, ‘anger’, ‘disgust’, ‘fear’, ‘happiness’.
Hamiltonian Operator Disentanglement of Content and Motion in Image Sequences

(a) In each patch, the first row is the original sequence, the second row is its reconstruction, and third row is a sequence generated by an action of the operator on the phase space representation of the starting time step. On left are results on sprites and on right on MUG. The reconstruction shows our model can learn good representations and the generation shows the dynamical operator can generate realistic motions from a starting frame.

(b) Left: original sequence pairs; right: reconstructions after swapping the motion variables in row pairs. The content space is disentangled from the motion space.

Figure 3. On left, generation vs reconstruction and on right motion transfer between two sequences. More examples are in Appendix C.2

Figure 4. On left (a) and (c) are examples of image to sequence generated by an action of Hamiltonian operator on the phase space representation of the starting frame. We can observe all dynamics are well separated that demonstrates the disentanglement of various actions. On right (b) is an example of unconditional generation. ‘sadness’ and ‘surprise’. We can see the visual dynamics associated with all the operators are well separated from one another. Further, we use our model to generate long-term sequences; the results are presented in Appendix C.2, where it is apparent that longer-term sequences maintain the consistency associated with the content-motion pair.

Next, we carry out ablation experiments to demonstrate the benefits of Hamiltonians over other dynamical methods. The quantitative and qualitative results are discussed in Appendix C.4. We summarise the key findings in Table 4. The Hamiltonian modelling approach outperforms other approaches and works best for generating sequences. We attribute the performance to the fact that it imposes a symplectic geometry in the phase space, which helps in the generation of long term sequences. The quadratic form of energy provides a relationship among latent components, which we speculate is critical for the interpretability of dynamics.

| Dynamics      | Image-To-Seq | Motion Swap | Structure  |
|---------------|--------------|-------------|------------|
| H             | ✓            | ✓           | Symplectic |
| Linear        | x            | ✓           | x          |
| RNN           | x            | ✓           | x          |
| Positional Encoding | x | ✓ | x |

Table 4. Properties compared to other dynamical methods. The Hamiltonian formulation impose a symplectic structure in motion space that is helpful for the image to sequence generation.

5. Conclusions and Future Work

We introduced a deep generative model that uses Hamiltonian operators in the latent space to model the manifold of various motions. We decompose the latent space into content and motion subspace, and for each action, associate a unique partition in the motion subspace. The dynamics in each partition are unrolled using a Hamiltonian operator. Our formulation, associates conserved quantities with the dynamics; we empirically show it provides a helpful notion of disentanglement for image sequences. We demonstrate the performance on several tasks such as image-to-sequence, motion swapping and unconditional generation. The main advantage of our model is we can generate long term trajectories and traverse the motion manifolds of different actions in the latent space. We look forward to future applications to other sequential data types, including music and speech.

A potential current limitation of our model is that it is less able to deal with irregularly sampled sequences, changes in tempo or reversals. In future work, this will be addressed by allowing a more flexible prior on the spacing between time steps. Nevertheless, all such approaches depend critically on the decomposition demonstrated here, and we argue that this work has demonstrated the significant promise of content and Hamiltonian motion decomposition of image sequences.
ACKNOWLEDGEMENTS

The authors would like to thank Joseph Mellor and William Toner for the helpful discussion during the project and Elliot J. Crowley for the valuable feedback on the paper. This research was funded in part by an unconditional gift from Huawei Noah’s Ark Lab, London.

References

Aifanti, N., Papachristou, C., and Delopoulos, A. (2010). The MUG facial expression database. In 11th International Workshop on Image Analysis for Multimedia Interactive Services WIAMIS 10, pages 1–4. IEEE.

Bader, P., Blanes, S., and Casas, F. (2019). Computing the matrix exponential with an optimized Taylor polynomial approximation. Mathematics, 7(12):1174.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. IEEE transactions on pattern analysis and machine intelligence, 35(8):1798–1828.

Bird, A. and Williams, C. K. (2019). Customizing sequence generation with multi-task dynamical systems. arXiv preprint arXiv:1910.05026.

Bondesan, R. and Lamacraft, A. (2019). Learning symmetries of classical integrable systems. arXiv preprint arXiv:1906.04645.

Botev, A., Jaegle, A., Wirnsberger, P., Hennes, D., and Higgins, I. (2021). Which priors matter? benchmarking models for learning latent dynamics.

Casale, F. P., Dalca, A. V., Saglietti, L., Listgarten, J., and Fusi, N. (2018). Gaussian process prior variational autoencoders. arXiv preprint arXiv:1810.11738.

Caselles-Dupré, H., Garcia-Ortiz, M., and Filliat, D. (2019). Symmetry-based disentangled representation learning requires interaction with environments. arXiv preprint arXiv:1904.00243.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. (2018). Neural ordinary differential equations. arXiv preprint arXiv:1806.07366.

Chevalley, C. (2016). Theory of Lie Groups (PMS-8), Volume 8. Princeton University Press.

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A., and Bengio, Y. (2015). A recurrent latent variable model for sequential data. arXiv preprint arXiv:1506.02216.

Cohen, T. and Welling, M. (2014). Learning the irreducible representations of commutative Lie groups. In International Conference on Machine Learning, pages 1755–1763. PMLR.

Connor, M. and Rozell, C. (2020). Representing closed transformation paths in encoded network latent space. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 3666–3675.
Cranmer, M., Sanchez-Gonzalez, A., Battaglia, P., Xu, R., Cranmer, K., Spergel, D., and Ho, S. (2020). Discovering symbolic models from deep learning with inductive biases. arXiv preprint arXiv:2006.11287.

Cremers, D. and Yuille, A. (2003). A generative model based approach to motion segmentation. In Joint Pattern Recognition Symposium, pages 313–320. Springer.

Culpepper, B. J. and Olshausen, B. A. (2009). Learning transport operators for image manifolds. In NIPS, pages 423–431.

Dandi, Y., Das, A., Singhal, S., Namboodiri, V., and Rai, P. (2020). Jointly trained image and video generation using residual vectors. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pages 3028–3042.

Denton, E. and Birodkar, V. (2017). Unsupervised learning of disentangled representations from video. In Proceedings of the 31st International Conference on Neural Information Processing Systems, pages 4417–4426.

Dupont, E., Martin, M. B., Colburn, A., Sankar, A., Susskind, J., and Shan, Q. (2020). Equivariant neural rendering. In International Conference on Machine Learning, pages 2761–2770. PMLR.

Easton, R. W. (1993). Introduction to Hamiltonian dynamical systems and the N-body problem (KR Meyer and GR Hall). SIAM Review, 35(4):659–659.

Eastwood, C. and Williams, C. K. (2018). A framework for the quantitative evaluation of disentangled representations. In International Conference on Learning Representations.

Elliott, R. J. and Krishnamurthy, V. (1999). New finite-dimensional filters for parameter estimation of discrete-time linear Gaussian models. IEEE Transactions on Automatic Control, 44(5):938–951.

Fraccaro, M., Kamronn, S., Paquet, U., and Winther, O. (2017). A disentangled recognition and nonlinear dynamics model for unsupervised learning. arXiv preprint arXiv:1710.05741.

Franceschi, J.-Y., Delasalles, E., Chen, M., Lamprier, S., and Gallinari, P. (2020). Stochastic latent residual video prediction. In International Conference on Machine Learning, pages 3233–3246. PMLR.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks. arXiv preprint arXiv:1406.2661.

Greydanus, S., Dzamba, M., and Yosinski, J. (2019). Hamiltonian neural networks. arXiv preprint arXiv:1906.01563.

He, J., Lehrmann, A., Marino, J., Mori, G., and Sigal, L. (2018). Probabilistic video generation using holistic attribute control. In Proceedings of the European Conference on Computer Vision (ECCV), pages 452–467.

Higgins, I., Amos, D., Pfau, D., Racaniere, S., Matthey, L., Rezende, D., and Lerchner, A. (2018). Towards a definition of disentangled representations. arXiv preprint arXiv:1812.02230.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8):1735–1780.

Hogg, D. (1983). Model-based vision: A program to see a walking person. Image and Vision Computing, 1(1):5–20.

Hsieh, J.-T., Liu, B., Huang, D.-A., Fei-Fei, L., and Niebles, J. C. (2018). Learning to decompose and disentangle representations for video prediction. arXiv preprint arXiv:1806.04166.

Hsu, W.-N., Zhang, Y., and Glass, J. (2017). Unsupervised learning of disentangled and interpretable representations from sequential data. arXiv preprint arXiv:1709.07902.

Hurri, J. and Hyvärinen, A. (2003). Temporal and spatiotemporal coherence in simple-cell responses: a generative model of natural image sequences. Network: Computation in Neural Systems, 14(3):527–551. PMID: 12938770.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems.

Kannan, A., Jojic, N., and Frey, B. J. (2005). Generative model for layers of appearance and deformation. In AISTATS, volume 2005.

Karl, M., Soelch, M., Bayer, J., and Van der Smagt, P. (2016). Deep variational Bayes filters: Unsupervised learning of state space models from raw data. arXiv preprint arXiv:1605.06432.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational Bayes. arXiv preprint arXiv:1312.6114.

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. (2017). Building machines that learn and think like people. Behavioral and brain sciences, 40.
Memisevic, R. (2012). On multi-view feature learning. arXiv preprint arXiv:1206.4609.

Miladinović, D., Gondal, W., Schölkopf, B., Buhmann, J. M., and Bauer, S. (2019). Disentangled state space models: Unsupervised learning of dynamics across heterogeneous environments.

Minderer, M., Sun, C., Villegas, R., Cole, F., Murphy, K., and Lee, H. (2019). Unsupervised learning of object structure and dynamics from videos. arXiv preprint arXiv:1906.07889.

Noether, E. (1918). Invariant variation problems, gott.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d ’Alché-Buc, F., Fox, E., and Garnett, R., editors, Advances in Neural Information Processing Systems 32, pages 8024–8035. Curran Associates, Inc.

Pavlovic, V., Rehg, J. M., and MacCormick, J. (2000). Learning switching linear models of human motion. In NIPS, volume 2, page 4.

Quesada, R., Barrett, T., and Clements, W. (2020). Learning disentangled representations and group structure of dynamical environments. Advances in Neural Information Processing Systems, 33.

Rao, R. P. and Ruderman, D. L. (1999). Learning Lie groups for invariant visual perception. Advances in neural information processing systems, pages 810–816.

Roweis, S. and Ghahramani, Z. (1999). A unifying review of linear Gaussian models. Neural computation, 11(2):305–345.

Sanchez-Gonzalez, A., Bapst, V., Cranmer, K., and Battaglia, P. (2019). Hamiltonian graph networks with ODE integrators. arXiv preprint arXiv:1909.12790.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. (2020). Learning to simulate complex physics with graph networks. In International Conference on Machine Learning, pages 8459–8468. PMLR.

Sohl-Dickstein, J., Wang, C. M., and Olshausen, B. A. (2010). An unsupervised algorithm for learning Lie group transformations. arXiv preprint arXiv:1001.1027.

Srivastava, N., Mansimov, E., and Salakhudinov, R. (2015). Unsupervised learning of video representations using LSTMs. In International conference on machine learning, pages 843–852. PMLR.

Starner, T. and Pentland, A. (1997). Real-time American sign language recognition from video using hidden Markov models. In Motion-based recognition, pages 227–243. Springer.

Storkey, A. and Williams, C. (2003). Image modeling with position-encoding dynamic trees. IEEE Transactions on Pattern Analysis and Machine Intelligence, 25(7):859–871.

Toth, P., Rezende, D. J., Jaegle, A., Racanière, S., Botve, A., and Higgins, I. (2019). Hamiltonian generative networks. arXiv preprint arXiv:1909.13789.

Tulyakov, S., Liu, M.-Y., Yang, X., and Kautz, J. (2018). MoCoGAN: Decomposing motion and content for video generation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1526–1535.

Villegas, R., Yang, J., Hong, S., Lin, X., and Lee, H. (2017). Decomposing motion and content for natural video sequence prediction. arXiv preprint arXiv:1706.08033.

Yildiz, C., Heinonen, M., and Lähdesmäki, H. (2019). ODE2VAE: Deep generative second order ODEs with Bayesian neural networks.

Yingzhen, L. and Mandt, S. (2018). Disentangled sequential autoencoder. In International Conference on Machine Learning, pages 5670–5679. PMLR.

Yoon, J., Jarrett, D., and van der Schaar, M. (2019). Time-series generative adversarial networks.

Zhu, Y., Min, M. R., Kadav, A., and Graf, H. P. (2020). S3VAE: Self-supervised sequential VAE for representation disentanglement and data generation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 6538–6547.
A. Derivation of ELBO

We use maximum loglikelihood on sequence variables to derive the evidence lower bound (ELBO),

\[
\log p(x_{1:T}|u) = \log \int p(x_{1:T}, z, s_{1:T}|u) ds_{1:T} dz \\
= \log \int \frac{p(x_{1:T}, z, s_{1:T}|u)}{q(z, s_{1:T}|x_{1:T}, u)} q(z, s_{1:T}|x_{1:T}, u) ds_{1:T} dz \\
\geq \int \log \left( \frac{p(x_{1:T}, z, s_{1:T}|u)}{q(z, s_{1:T}|x_{1:T}, u)} \right) q(z, s_{1:T}|x_{1:T}, u) ds_{1:T} dz \\
= \mathbb{E}_{q(z, s_{1:T}|x_{1:T}, u)} \log \left( \frac{p(x_{1:T}, z, s_{1:T}|u)}{q(z, s_{1:T}|x_{1:T}, u)} \right) \\
\text{(16)}
\]

where \( s_t = [q_t, p_t] \). The joint distribution is factorised as,

\[
p(x_{1:T}, z, s_{1:T}|u) = p(z)p(x_1|q_1, z) \prod_{t=1}^{T-1} p(x_{t+1}|q_{t+1}, z)p(q_{t+1}, p_{t+1}|q_t, p_t, u) \\
\text{(17)}
\]

Since, we transform the starting latent state \( s_1 = [q_1, p_1] \) using a deterministic transformation \( f(t, H; \omega) = e^H \) (where \( \omega \) are the parameters of \( H \) matrix), we can write our transition distribution as,

\[
p(s_{t+1}|s_t, u) = p(s_t|s_{t-1}, u) \frac{df}{ds_t} = p(s_t|s_{t-1}, u)e^{\text{Tr}(H)} = p(s_t|u) \prod_{\ell=1, \neq t}^T e^{\text{Tr}(H)} \text{(18)}
\]

where \( \text{Tr} \) is the trace operator and \( p(s_1|u) = p(q_1|u)p(p_1|u) \). The transition model is reversible; therefore, without loss of generality we can replace starting step 1 with any arbitrary \( t \) and unroll both forward and backward. We next equate (18) in the generative model defined in (17) that reduces the factorisation to,

\[
p(x_{1:T}, z, s_{1:T}|u) = p(z)p(x_1|q_1, z)p(q_1|u)p(p_1|u) \prod_{t=1, \neq t}^{T-1} p(x_{t+1}|q_{t+1}, z)p(q_{t+1}, p_{t+1}|q_t, p_t, u)e^{\text{Tr}(H)} \\
\text{(19)}
\]

We factorise the variational distribution \( q(z, s_{1:T}|x_{1:T}, u) \) as,

\[
q(z, s_{1:T}|x_{1:T}, u) = q(z|x_{1:T}) \prod_{t} q(p_t|x_{1:t-1}, u)q(p_t|x_{t:t-1}, u), \quad s_t = [q_t, p_t] \\
\text{(20)}
\]

We now use the equations (20) and (19) to rewrite the ELBO as,

\[
\mathbb{E}_q(z|x_{1:T}, q(q_1|x_1,u), q(p_1|x_1-u), q(p_{1:t-1,u})) \log \left( \frac{p(z)p(q_1|x_1)q(p_1|u)}{q(z|x_1,T) \prod_{t=1, \neq t} q(p_t|x_{1:t-1}, u)} \right) \\
\mathbb{E}_q(q_1|x_1,u) \log \left( \frac{p(q_1|x_1)}{q(q_1|x_1,u)} \right) + \mathbb{E}_q(p_1|x_{1:t-1}, u) \log \left( \frac{p(p_1|u)}{q(p_1|x_{t:t-1}, u)} \right) \\
+ \mathbb{E}_q(z|x_{1:T}) \log \left( \frac{p(z)}{q(z|x_{1:T})} \right) + \mathbb{E}_q(q|x_{1:T}) \log \left( \sum_{x_{t+1}} p(x_{t+1}|x_t, z) \right) \\
\text{(21)}
\]

The trace of real-Hamiltonian matrix is zero we can therefore omit the term \( \text{Tr}(H) \). Since, for each motion \( u_k \) we associate a separate Hamiltonian \( H_k \) that acts on a subspace \( S^k \), we can view the full state space \( S \) as a partitions of symmetry groups \( S = S_1 \oplus \cdots \oplus S_K \) where the Hamiltonian \( H \) is in the block diagonal form \( H = \text{diag}(H_1, \cdots, H_K) \). We therefore, express
the distributions in terms of the variables of their respective subspaces to obtain the final ELBO,

$$
- KL[q_t(x_t, u_t)||p(q_t)] - KL[q_t(x_{t-1}, u_t)||p(p_t)] - KL[q(z|x_{1:T}, u)||p(z)] + E_{q(z|x_{1:T}, u)} \left[ \sum_t \log p(x_t|q_t, z) \right]
$$

(23)

The probabilistic graph of our generative and inference model is described in Figure 5.

B. Background

In this section, we provide a short overview of the definitions relevant in the context of our work.

The study of symmetries plays a fundamental role in discovering the constants of the physical systems. For instance, the space translation symmetry means the conservation of linear momentum, and the rotation symmetry implies the conservation of angular momentum. Groups are fundamental tools used for studying symmetry transformations. Formally we say,

**Definition 2.** A group $G$ is a set with a binary operation $\ast$ satisfying following conditions:

- closure under $\ast$, i.e., $x \ast y \in G$ for all $x, y \in G$
- there is an identity element $e \in G$, satisfying $x \ast e = e \ast x = e$ for all $x \in G$
- for each element $x \in G$ there exist an inverse $x^{-1} \in G$ such that $x \ast x^{-1} = x^{-1} \ast x = e$
- for all $x, y, z \in G$ the associative law holds i.e. $x \ast (y \ast z) = (x \ast y) \ast z$

The nature of the symmetry present in a system decides whether a group is discrete or continuous. A group is discrete if it has a finite number of elements. For e.g., a dihedral group $D2$ generated using an $e$ identity, $r$ rotation by $\pi$, and $f$ reflection along x-axis consists of finite elements $\{e, r, f, rf\}$. The group generators are a set of elements that can generate other group elements using the group multiplication rule. For $D2$ the generators are $\{e, r, f\}$. A continuous group is characterised by the notion of infinitesimal transformation and are generally known as Lie groups.

**Definition 3.** A Lie group $G$ is a group which also forms a smooth manifold structure, where the group operations under multiplication $G \times G \rightarrow G$ and its inverse $G \rightarrow G$ are smooth maps.

A group of 2D rotations in a plane is one common example of Lie group given by, $SO(2) = \{ R \in \mathbb{R}^{2 \times 2} | R^T R = I, det(R) = 1 \}$. The $SO(2)$ a single parameter $\theta$ group simply given by a 2D rotation matrix $R(\theta) = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$.

**Definition 4.** A Lie algebra $\mathfrak{g}$ of a Lie group $G$ is the tangent space to a group defined at its identity element $I$ with an exponential map $exp : \mathfrak{g} \rightarrow G$ and a binary operation $\mathfrak{g} \times \mathfrak{g} \rightarrow \mathfrak{g}$.
The structure of Lie groups are of much interest due to the Noether’s theorem that states for any differentiable symmetry there exists a conservation law. In physics such conservation laws are studied by identifying the Hamiltonian of the physical system (Easton, 1993). In this work, we look at two choice of Hamiltonians that form a symplectic group $Sp(2d)$ and symplectic orthogonal group $SpO(2d)$ structure.

**Definition 5.** A symplectic group $Sp(2d)$ is a Lie group formed by the set of real symplectic matrices defined as $Sp(2d) = \{H \in \mathbb{R}^{2d \times 2d} | H^T J H = J\}$, where $J = \begin{pmatrix} 0 & I_d \\ -I_d & 0 \end{pmatrix}$.

**Definition 6.** The Lie algebra $\mathfrak{sp}$ of a symplectic group $Sp(2d)$ is a vector space defined by, $\mathfrak{sp} = \{H \in \mathbb{R}^{2d \times 2d} | JH = (JH)^T \}$

**Definition 7.** A symplectic orthogonal group $SpO(2d)$ is defined by restricting the Hamiltonian to the orthogonal group.

**Definition 8.** A group action is a map $\circ : G \times \mathcal{X} \rightarrow \mathcal{X}$ iff (i) $e \circ x = x$, $\forall x \in \mathcal{X}$, where $e$ is the identity element of $G$, (ii) $(g_1 \cdot g_2) \circ x = g_1 \cdot (g_2 \circ x)$, $g_1,g_2 \in G, \forall x \in \mathcal{X}$ where $\cdot$ is a group operation.

## C. Experiment and Results

### C.1. Network Architecture

The architecture of the encoder and decoder network is based on (Yingzhen and Mandt, 2018) also outlined in Table 7 and 8. We use the same network architecture for both sprites and the MUG dataset. The output of an encoder is fed to the content, position, and momentum network to get the variational distributions in $Z, Q$ and $P$ space. Table 9 describes the architecture of the network. For the position and momentum network, the input action $k$ is represented by a one-hot vector $u$ that takes one at index $k$ and is zero elsewhere.

#### C.1.1. Training Details

For MUG, we choose $|Z| = 512$, $|Q| = K \times 12$ and $|P| = K \times 12$ and for sprites $|Z| = 256$, $|Q| = K \times 6$ and $|P| = K \times 6$, where $K$ is the number of actions. For sprites, $K = 3$ and for MUG $K = 6$. To train all our models we use an Adam (Kingma and Ba, 2014) optimiser with a learning rate of $2 \times 10^{-4}$ and a batch size of 24. We use Pytorch (Paszke et al., 2019) for the implementation. The code will be made available on publication. We train all our models on Nvidia GeForce RTX 2080 GPUs.

### C.2. Results and Discussion

We further provide extended qualitative samples of our model on the MUG and sprites dataset. Figure (12) shows results of conditional sequence generation, Figure (15) shows results of motion swapping. Figure (13, 14) further shows examples of image to sequence generation. We generate 16 frames in future conditioned on an initial starting frame. Next, we adapt our model to the scenarios where action variables are $u$ not available.

**Unconditional Dynamics** In our formulation introduced in Section 3.1.1, we use the action variable $u$ to map the sequence to its respective phase space that allows the separability of dynamics and controlled generation of motion sequences. The choice to use action variables do not restrict the Hamiltonian dynamics; in this section, we adapt our formulation to sequences where action variables are not available. Specifically, we factorise the phase space into $K$ symmetry groups where the Hamiltonian takes the form $H = \text{block-diagonal}(H_1, \ldots, H_K)$. To unroll the trajectory for any arbitrary sequence $x_{1:T}$ we evolve all the operators simultaneously as,

$$s_t = f(s_{t-1}; \omega_k, t) = \text{block-diagonal}(e^{iH_1s_{1,t}}, \ldots, e^{iH_Ks_{K,t-1}}) \quad \forall t > 1$$

We want to remark that in such a formulation, we don’t have direct control over the action generated by dynamics. The type of motion generated depends on the initial position and momenta variable. Furthermore, the operators $H_k$ may not necessarily correspond to specific action instead could describe a more general property that is conserved and shared across motions. For instance, different operators could capture varying magnitude of action movement like smiling, surprise, etc. To investigate it empirically we map a starting frame to phase space and generate a sequence using individual $H_k$ as well as $H$. Figure 7 describes the generated motion sequences. The first six rows are sequences generated by individual $H_k$, and the combined $H$ generates the last row. We can observe the operators capture the varying extent of motion. Figure 6 further shows the performance of a model on sequence generation and motion transfer.
C.3. Ablation

To investigate the effectiveness of our dynamical model, we perform the following ablation studies,

**What is the benefit of Constant Energy?** The Hamiltonian formulation maintains the constant energy along time. Such a choice is beneficial for generating long term sequences. In this part, we generate long sequences using our dynamical model and look at the evaluation of energy over time.

The total Hamiltonian energy in the phase space is given by,

$$ E = \frac{1}{2} s'^TMs $$  \tag{25} 

where \( s = (q, p) \), and \( M \) is a symmetric matrix. Let \( M \) be a 2 \times 2 block matrix \( M = \begin{pmatrix} A & B \\ B & C \end{pmatrix} \). We can expand the energy term as,

$$ E = \frac{1}{2} q'^T A q + \frac{1}{2} p'^T C p + \frac{1}{2} q'^T A p + \frac{1}{2} p'^T B q $$  \tag{26} 

The first term is the potential energy (PE), the second is kinetic energy (KE), and the last two combined are a non-separable term. When \( B \) is zero, the energy is entirely separable into KE and PE terms. The non-separable Hamiltonian is common in many physical problems, for instance, rigid body dynamics and many others appearing in quantum mechanics. For details, we refer to (Easton, 1993). The choice of the unconstrained linear form of Hamiltonian was motivated to allow more flexibility to the model to learn in a data-driven way.

In Figure 8, we report the plot of energy over time for an image under different motion dynamics. The change in the individual energy shows the dynamics are not constant; this is also evident from the corresponding image sequences shown in the plot. As dynamics evolve, the total energy is strictly conserved, demonstrating that the trajectory cannot diverge from the learned symplectic structure. The results demonstrate the benefit of our model in generating long term sequences. We want to add a remark that the energy terms should be interpreted with care. It might not have any equivalence to the energy of a physical system; what it does is that it provides constraints to use the time translation symmetry of the dynamics.

**Linear Model** A linear dynamical is defined as,

$$ h_t = A_{t-1} s_{t-1} + B_{t-1} h_{t-1} + b $$  \tag{27} 

where \( h_t \) is a hidden state, and \( \{A, B, b\} \) are learnable parameters. To generate the trajectory \( x_{1:T} \), we combine the state coordinates \( h_{1:T} = \{h_1, \ldots, h_T\} \) with the content variable \( z \) and pass the joint representation through the decoder network.
Supplementary Material

Figure 7. Unconditional Hamiltonian approach. An example of image to sequence generation. The first column is the starting frame, first six rows correspond to sequence generated by the action of $k-th$ block of $H$, and the last row is the sequence generated by the full $H$.

We report the performance of a linear model in conditional as well as unconditional setting.

**Positional Encoding** We generate a simplistic baseline using a fixed Fourier encoding representation. Specifically, for a sequence of frames $x_{1:T} = \{x_1, \ldots, x_T\}$ we map it to a content variable $z$ and a frame $x_t$ to a phase $\phi_t \in [-1, 1]$. We then generate $T-t$ linearly separated phase coordinates $\{\phi_t, \ldots, \phi_T\} \in [\phi_t, 1 + \phi_t]$ and define the motion space representation as,

$$s_t = \{\sin(\phi_t 2^k), \cos(\phi_t 2^k)\}_{k=1}^{k=\lfloor d/2 \rfloor}$$

(28)

where $d$ is the size of motion space. We impose a Gaussian prior on the phase coordinates $p(\phi_t) = \mathcal{N}(0, 1)$.

C.4. Discussion

In this section, we compare the results of our formulation with other choices of dynamical models. We describe the qualitative results in Table 5. The Hamiltonian model achieves the best performance across all scores. We observe all models except the position encoding achieve comparable performance on identity prediction. We speculate this could be due to the non-changing dynamics, which makes predicting the identity from a sequence of static images a much easier task for a classifier. Due to the failure of positional encoding, we omit it from the rest of the discussion.

We restrict the qualitative analysis to the conditional models. Figure 10 shows the comparison of the models on the task of motion swap. The motion variables under Hamiltonian dynamics are better disentangled from the content variables. Figure 9 describes the results on image to sequence generation, further demonstrate that the Hamiltonian dynamics are consistent in long term prediction and prevent the flow of constant information to motion variables. Overall the Hamiltonian formulation outperforms other approaches and works best across all tasks.

C.5. Rotating MNIST

In this section, we investigate the performance of our approach in predicting the rotations of MNIST digits. We use an unconditional version of our model for this part. Following the procedure of (Casale et al., 2018), we generated sequences of 16 time steps by rotating the images of digit “3”. We followed the same training procedure. In Figure 11, part (a) first row is the input sequence, and the second row is a reconstruction. In part (b), we show three sequences generates by random initial phase space coordinates. The network architecture for MNIST experiments is outlined in the Table (10, 11, 12).

Next, in Table 6, we compare the mean squared error (MSE) of our model with the other related methods (Yildiz et al., 2019; Casale et al., 2018). Our model achieves comparable performance to GPPVAE. The ODE2VAE performs best in terms of MSE; this can be attributed to using a second-order latent ODE model. In contrast, our formulation only uses the first-order dynamics, which provides extra computational efficiency. Furthermore, compared to GPPVAE, we don’t have costly kernel computations.
Figure 8. We map a starting frame to the phase space and use the operators $H$ to generate the phase space trajectory, which is then mapped to data space using the decoder network. At the top is the plot of energy vs time of the operators $H_k$ ($E$ is the total energy, $KE$ is the kinetic energy term, $PE$ is the potential energy, and $NonSep$ is the non-separable term). Below, each row is the sequence generated by the action of $H_k$.

We want to remark datasets such as stochastic movingMNIST (Denton and Birodkar, 2017) used in a few disentanglement papers is not a good application of our model. This is due to the nature of dynamics generated by an action of a random transformation. The Hamiltonian model relies on a dataset of data sequences where dynamics follow a conserved quantity and can be associated with constant energy. This assumption may or may not hold for SMNIST data due to random movements.
Table 5. Quantitative evaluation of disentanglement and diversity of generated samples

(a) Results of classifier on MUG for different dynamical model. The high score of accuracy and Inter-Entropy $H(y)$ while low scores of Intra-Entropy $H(y|x)$ are expected from a better model.

(b) Comparison to other baseline in terms of accuracy of identity of sequences. This shows our model can preserve content when the motion representation is changed.

Table 6. Mean squared error on test set of rotating MNIST.

Table 7. Encoder network
Figure 9. Results on Image to sequence generation. On the left, is the starting frame and on the right are different motions generated the dynamical models.
Figure 10. Results of motion transfer. On the top are pair of original sequence. On bottom, left side are the results of Linear Model, center of RNN and on right of our Hamiltonian model. The first two rows are reconstruction of sequence and the last two rows are results of swapping the motion variables of the two sequences.

Figure 11. Results on Rotating MNIST with a learnable Hamiltonian operator. On the top left, we have four input sequences, and on the right, their reconstruction; on the bottom, we have four sequences generated by an action of Hamiltonian on the state space coordinate of the frame in the first column.

Decoder Architecture of Sprites and MUG

| Layer          | Description                                      |
|----------------|--------------------------------------------------|
| Linear         | in: \( h \), out: 4096                          |
| BatchNorm1d    | \( \rightarrow \) LeakyReLU(0.2)                 |
| Linear         | in: 4096, out: \( c \times w \times h \)         |
| BatchNorm1d    | \( \rightarrow \) LeakyReLU(0.2) \( \rightarrow \) Rearrange\('b (c \times w \times h) -> b \times c \times w \times h'\) |
| ConvTranspose2d| kernels: 256, kernelSize: \( (5, 5) \), stride: \( (2, 2) \), padding: \( (2, 2) \) \( \rightarrow \) LeakyReLU(0.2) |
| ConvTranspose2d| kernels: 256, kernelSize: \( (5, 5) \), stride: \( (2, 2) \), padding: \( (2, 2) \) \( \rightarrow \) LeakyReLU(0.2) |
| ConvTranspose2d| kernels: 256, kernelSize: \( (5, 5) \), stride: \( (2, 2) \), padding: \( (2, 2) \) \( \rightarrow \) LeakyReLU(0.2) |
| ConvTranspose2d| kernels: 256, kernelSize: \( (5, 5) \), stride: \( (2, 2) \), padding: \( (2, 2) \) \( \rightarrow \) LeakyReLU(0.2) |
| ConvTranspose2d| kernels: 256, kernelSize: \( (5, 5) \), stride: \( (1, 1) \), padding: \( (2, 2) \) \( \rightarrow \) Tanh() |

Table 8. Decoder network
(a) Conditional Sequence Generation. The first row is the original sequence, second row is a reconstructed sequence and third is generated by an action of dynamical model on the first time frame.

*Figure 12.* Conditional Sequence Generation. The first row is the original sequence, second row is a reconstructed sequence and third is generated by an action of dynamical model on the first time frame.
Figure 13. Image to Sequence generation. We generate dynamics of different action from a given image. Each row is a unique action generated by the operator associated with that action.
Figure 14. Image to Sequence generation. We generate dynamics of different action from a given image. Each row is a unique action generated by the operator associated with that action.

| Content   | Position          | Momentum         |
|-----------|-------------------|------------------|
| LSTM      | in: $h$, out: $z$ | Linear in: $h+k$, out: $v$ | Linear in: $h+k$, out: $v$ |
| Linear$_\mu$ | in: $z$, out: $z$ | BatchNorm1d $\rightarrow$ LeakyReLU(0.2) | BatchNorm1d $\rightarrow$ LeakyReLU(0.2) |
| Linear$_{log\sigma}$ | in: $z$, out: $z$ | Linear in: $v$, out: $v$ | Linear in: $v$, out: $v$ |
|           |                   | BatchNorm1d $\rightarrow$ LeakyReLU(0.2) | BatchNorm1d $\rightarrow$ LeakyReLU(0.2) |
| Linear$_\mu$ | in: $v$, out: $q$ | TCN kernelSize: 3, pad: 2, stride: 1 | Linear$_\mu$ in: $v$, out: $p$ |
| Linear$_{log\sigma}$ | in: $v$, out: $q$ | Linear$_{log\sigma}$ in: $v$, out: $p$ | Linear$_{log\sigma}$ in: $v$, out: $p$ |

Table 9. Content and Motion network. TCN stands for temporal convolution network.
Figure 15. Motion Swapping. In each patch the first two rows are original sequence and the next two rows obtained by swapping motion variables of two sequences.
**Supplementary Material**

---

### Encoder Architecture MNIST

| Layer | Description |
|-------|-------------|
| Conv2d | kernels: 32, kernelSize: (5, 5), stride: (2, 2), padding: (2, 2)  
      | BatchNorm2d → ReLU() |
| Conv2d | kernels: 64, kernelSize: (5, 5), stride: (2, 2), padding: (2, 2)  
      | BatchNorm2d → ReLU() |
| Conv2d | kernels: 128, kernelSize: (5, 5), stride: (2, 2), padding: (2, 2)  
      | BatchNorm2d → ReLU() |
| Linear | in: \((c \times w \times h)\), out: 4096  
        | BatchNorm1d → ReLU() |
| Linear | in: 4096, out: 256  
        | BatchNorm1d → ReLU() |

*Table 10. Encoder network MNIST*

---

### Decoder Architecture MNIST

| Layer | Description |
|-------|-------------|
| Linear | in: 20, out: 4096  
      | BatchNorm1d → ReLU() |
| Linear | in: 4096, out: \((c \times w \times h)\)  
      | BatchNorm1d → ReLU() → Rearrange(‘b (c w h) -> b c w h’)  
      | BatchNorm1d → ReLU() |
| ConvTranspose2d | kernels: 128, kernelSize: (3, 3), stride: (1, 1), padding: (0, 0)  
        | BatchNorm2d → ReLU() |
| ConvTranspose2d | kernels: 64, kernelSize: (5, 5), stride: (2, 2), padding: (1, 1)  
        | BatchNorm2d → ReLU() |
| ConvTranspose2d | kernels: 32, kernelSize: (5, 5), stride: (2, 2), padding: (1, 1)  
        | BatchNorm2d → ReLU() |
| ConvTranspose2d | kernels: 1, kernelSize: (5, 5), stride: (1, 1), padding: (2, 2)  
        | BatchNorm2d → Sigmoid() |

*Table 11. Decoder network MNIST*

---

### Motion Network MNIST

| Layer | Description |
|-------|-------------|
| Linear | in: 256, out: 320  
        | BatchNorm1d → LeakyReLU(0.2) |
| Linear | in: 320, out: 20  
        | BatchNorm1d → LeakyReLU(0.2) |
| Linear | in: 20, out: 20  
        | TCN kernelSize: 4, pad: 3, stride: 1 |
| Linear | in: 20, out: 20  
        | Linear<br>µ in: 20, out: 20  
        | Linear<br>log σ in: 20, out: 20 |

*Table 12. Motion network MNIST. TCN stands for temporal convolution network.*