Supplementary Material

Asif Khan
School of Informatics
University of Edinburgh
asif.khan@ed.ac.uk

Amos Storkey
School of Informatics
University of Edinburgh
a.storkey@ed.ac.uk

A Appendix

A .1 ELBO Derivation

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

$$\begin{aligned}\log p(\mathbf{x}_{1:T}|\mathbf{u}) &= \log \int p(\mathbf{x}_{1:T}, \mathbf{z}, \mathbf{s}_{1:T}|\mathbf{u}) d\mathbf{s}_{1:T} d\mathbf{z} \\ &= \log \int \frac{p(\mathbf{x}_{1:T}, \mathbf{z}, \mathbf{s}_{1:T}|\mathbf{u})}{q(\mathbf{z}, \mathbf{s}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u})} q(\mathbf{z}, \mathbf{s}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}) d\mathbf{s}_{1:T} d\mathbf{z} \\ &\geq \int \log \left[\frac{p(\mathbf{x}_{1:T}, \mathbf{z}, \mathbf{s}_{1:T}|\mathbf{u})}{q(\mathbf{z}, \mathbf{s}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u})} \right] q(\mathbf{z}, \mathbf{s}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}) d\mathbf{s}_{1:T} d\mathbf{z} \\ &\geq \mathbb{E}_{q(\mathbf{z}, \mathbf{s}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u})} \log \left[\frac{p(\mathbf{x}_{1:T}, \mathbf{z}, \mathbf{s}_{1:T}|\mathbf{u})}{q(\mathbf{z}, \mathbf{s}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u})} \right] \end{aligned} \tag{1}$$

where $\mathbf{s}_t = [\mathbf{q}_t, \mathbf{p}_t]$. The joint distribution is factorised as,

$$p(\mathbf{x}_{1:T}, \mathbf{z}, \mathbf{s}_{1:T}|\mathbf{u}) = p(\mathbf{z}) \prod_{t=1}^{T} p(\mathbf{x}_t|\mathbf{q}_t, \mathbf{z}) p(\mathbf{q}_t, \mathbf{p}_t|\mathbf{q}_{t-1}, \mathbf{p}_{t-1}, \mathbf{u}) \tag{2}$$

Since, we transform starting latent state $\mathbf{s}_1 = [\mathbf{q}_1, \mathbf{p}_1]$ using a deterministic transformation $f(\omega) = e^{t\mathbf{H}}$, we can write our transition distribution as,

$$p(\mathbf{s}_{t+1}|\mathbf{s}_t) = p(\mathbf{s}_t) \left| \frac{df}{d\mathbf{s}_t} \right| = p(\mathbf{s}_t) e^{\mathit{Trace}(\mathbf{H})} \tag{3}$$

Using the result we obtain the transition distribution over T steps as,

$$\prod_{t}^{T} p(\mathbf{q}_t, \mathbf{p}_t|\mathbf{q}_{t-1}, \mathbf{p}_{t-1}, \mathbf{u}) = p(\mathbf{q}_1, \mathbf{p}_1|\mathbf{u}) e^{T.\mathit{Trace}(\mathbf{H})} \tag{4}$$

The transition model is reversible; therefore, without loss of generality we can replace starting step 1 with any arbitrary t. We now equate (4) in the generative model that reduces the factorisation to,

$$p(\mathbf{x}_{1:T}, \mathbf{z}, \mathbf{s}_{1:T}|\mathbf{u}) = p(\mathbf{z}) p(\mathbf{s}_t|\mathbf{u}) e^{T.Trace(\mathbf{H})} \prod_{t}^{T} p(\mathbf{x}_t|\mathbf{q}_t) \tag{5}$$

We factorise the variational distribution as,

$$q(\mathbf{z}, \mathbf{s}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}) = q(\mathbf{z}|\mathbf{x}_{1:T}) q(\mathbf{q}_t|\mathbf{x}_t, \mathbf{u}) q(\mathbf{p}_t|\mathbf{x}_{t:t-w}, \mathbf{u}) \prod_{t} q(\mathbf{s}_{t+1}|\mathbf{s}_t), \quad \mathbf{s}_t = [\mathbf{q}_t, \mathbf{p}_t] \tag{6}$$

1st CtrlGen: Controllable Generative Modeling in Language and Vision Workshop at NeurIPS 2021.

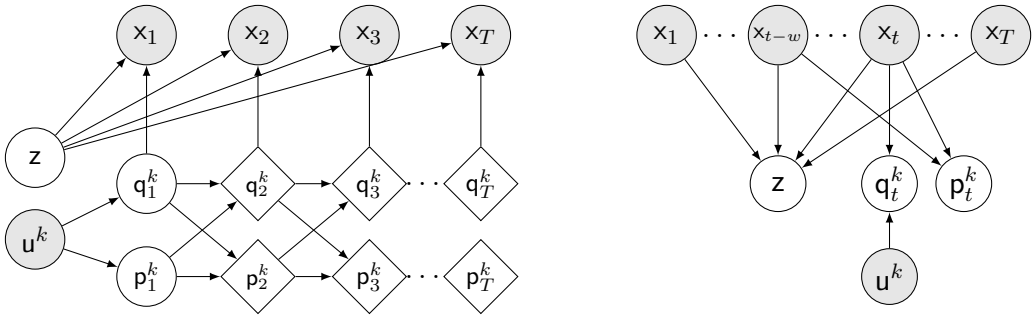


Figure 1: Probabilistic graph of our proposed model. On the left is a graph of our generative model and on the right is of our inference model.

$$q(\mathbf{s}_{t+1}|\mathbf{s}_t) = q(\mathbf{q}_t|\mathbf{x}_t, \mathbf{u})q(\mathbf{p}_t|\mathbf{x}_{t:t-w}, \mathbf{u}) \left| \frac{df}{d\mathbf{s}_t} \right| = q(\mathbf{q}_t|\mathbf{x}_t, \mathbf{u})q(\mathbf{p}_t|\mathbf{x}_{t:t-w}, \mathbf{u})e^{T.Trace(\mathbf{H})} \tag{7}$$

where w is a window size to condition momentum variable on previous steps. We can rewrite the variational distribution as,

$$q(\mathbf{z}, \mathbf{s}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}) = q(\mathbf{z}|\mathbf{x}_{1:T})q(\mathbf{q}_t|\mathbf{x}_t, \mathbf{u})q(\mathbf{p}_t|\mathbf{x}_{t:t-w}, \mathbf{u})e^{TTrace(\mathbf{H})} \tag{8}$$

We now use the equations (8) and (5) to rewrite the ELBO as,

$$\begin{aligned}
\log p(\mathbf{x}_{1:T}|\mathbf{u}) \geq \mathbb{E}_{q(\mathbf{z}|\mathbf{x}_{1:T}),q(\mathbf{q}_t|\mathbf{x}_t,\mathbf{u}),q(\mathbf{p}_t|\mathbf{x}_{t:t-w},\mathbf{u})} \log \left[\frac{p(\mathbf{z})p(\mathbf{q}_t, \mathbf{p}_t|\mathbf{u})e^{T.Trace(\mathbf{H})} \prod_t^T p(\mathbf{x}_t|\mathbf{q}_t, \mathbf{z})}{q(\mathbf{z}|\mathbf{x}_{1:T})q(\mathbf{q}_t|\mathbf{x}_t, \mathbf{u})q(\mathbf{p}_t|\mathbf{x}_{t:t-w}, \mathbf{u})e^{T.Trace(\mathbf{H})}} \right] \\
\mathbb{E}_{q(\mathbf{q}_t|\mathbf{x}_t,\mathbf{u})} \log \left[\frac{p(\mathbf{q}_t|\mathbf{u})}{q(\mathbf{q}_t|\mathbf{x}_t, \mathbf{u})} \right] + \mathbb{E}_{q(\mathbf{p}_t|\mathbf{x}_{t:t-w},\mathbf{u})} \log \left[\frac{p(\mathbf{p}_t|\mathbf{u})}{q(\mathbf{p}_t|\mathbf{x}_{t:t-w}, \mathbf{u})} \right] \\
+ \mathbb{E}_{q(\mathbf{z}|\mathbf{x}_{1:T})} \log \left[\frac{p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x}_{1:T})} \right] + \mathbb{E}_{q(\mathbf{q}_t|\mathbf{x}_t,\mathbf{u})} \left[\sum_{t'} \log p(\mathbf{x}_{t'}|\mathbf{q}_{t'}, \mathbf{z}) \right]
\end{aligned}$$

Since, for each motion $\mathbf{u}_k$ we associate a separate Hamiltonian $\mathbf{H}_k$ that acts on a subspace $\mathbf{S}^k$, we can view the full state space $\mathbf{S}$ as a partitions of symmetry groups $\mathbf{S} = \mathbf{S}_1 \oplus \cdots \oplus \mathbf{S}_K$ where the Hamiltonian $\mathbf{H}$ is in the block diagonal form $\mathbf{H} = diag(\mathbf{H}_1, \cdots, \mathbf{H}_K)$. We therefore, express the distributions in terms of the variables of their respective subspaces. to obtain the final ELBO,

$$\begin{aligned}
\log p(\mathbf{x}_{1:T}|\mathbf{u}) \geq -KL[q(\mathbf{q}_t^k|\mathbf{x}_t, \mathbf{u})||p(\mathbf{q}_t^k)] - KL[q(\mathbf{p}_{t:t-w}^k|\mathbf{x}_t, \mathbf{u})||p(\mathbf{p}_t^k)] \\
- KL[q(\mathbf{z}|\mathbf{x}_{1:T}, \mathbf{u})||p(\mathbf{z})] + \mathbb{E}_{q(\mathbf{q}_t^k|\mathbf{x}_t,\mathbf{u})} \left[\sum_{t'} \log p(\mathbf{x}_{t'}|\mathbf{q}_{t'}, \mathbf{z}) \right] \qquad (9)
\end{aligned}$$

(10)

The probabilistic graph of our generative and inference model is provided in

A .2 Background

In this section, we provide a small overview of the necessary definitions useful in the context of our work.

The symmetry of an object is a transformation that leaves some of its properties unchanged. E.g., translation, rotation, etc. 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 1. A group G is a set with a binary operation $*$ satisfying following conditions:

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

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 by an e identity, r rotation by π, 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 2. A Lie group G is a group which also forms a smooth manifold structure, where the group operations under multiplication $G \times G \to G$ and its inverse $G \to G$ are smooth maps.

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

Definition 3. 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 $exp : \mathfrak{g} \to G$ and a binary operation $\mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$.

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

Definition 5. The Lie algebra $\mathfrak{sp}$ of a symplectic group $Sp(2d)$ is a vector space defined by, $\mathfrak{sp} = \{\mathbf{H} \in \mathbb{R}^{2d\times 2d} | \quad \mathbf{JH} = (\mathbf{JH})^T\}$

Definition 6. A symplectic orthogonal group $SpO(2d)$ is defined by restricting the Hamiltonian to the orthogonal group.

Definition 7. A group action is a map $\circ : G \times \mathcal{X} \to \mathcal{X}$ 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.

A .3 Experiment and Results

A .3.1 Network Architecture

The architecture of the encoder and decoder network is based on [2] also outlined in Table 1 and 2. We use the same network architecture for both sprites and MUG dataset. The output of an encoder is given as an input to the content, position, and momentum network to get the variational distributions in $\mathbf{Z}$, $\mathbf{Q}$ and $\mathbf{P}$ space. The details of network are described in Table 3. For the position and momentum network, the input action k is represented by a one-hot representation $\mathbf{u}$ that takes one at index k and is zero everywhere else.

Encoder Architecture	
Conv2d	kernels=256, kernelSize=(5,5), stride=(1,1), padding=(2,2)
	BatchNorm2d -> LeakyReLU(0.2)
Conv2d	kernels=256, kernelSize=(5,5), stride=(2,2), padding=(2,2)
	BatchNorm2d -> LeakyReLU(0.2)
Conv2d	kernels=256, kernelSize=(5,5), stride=(2,2), padding=(2,2)
	BatchNorm2d -> LeakyReLU(0.2)
Conv2d	kernels=256, kernelSize=(5,5), stride=(2,2), padding=(2,2)
	BatchNorm2d -> LeakyReLU(0.2)
Conv2d	kernels=256, kernelSize=(5,5), stride=(1,1), padding=(2,2)
BatchNorm2d -> LeakyReLU(0.2) -> Rearrange('b c w h -> b (c w h)')	
Linear	inSize=(c w h), outSize=(4096)
	BatchNorm1d -> LeakyReLU(0.2)
Linear	inSize=(4096), outSize=(2048)
	BatchNorm1d -> LeakyReLU(0.2)
Linear	inSize=(2048), outSize=(h)
	BatchNorm1d -> LeakyReLU(0.2)

Table 1: Encoder network

Decoder Architecture	
Linear	inSize=(h), outSize=(4096)
	BatchNorm1d -> LeakyReLU(0.2)
Linear	inSize=(4096), outSize=(c w h)
BatchNorm1d -> LeakyReLU(0.2) -> Rearrange('b (c w h) -> b c w h')	
ConvTranspose2d	kernels=256, kernelSize=(5,5), stride=(2,2), padding=(2,2)
	BatchNorm2d -> LeakyReLU(0.2)
ConvTranspose2d	kernels=256, kernelSize=(5,5), stride=(2,2), padding=(2,2)
	BatchNorm2d -> LeakyReLU(0.2)
ConvTranspose2d	kernels=256, kernelSize=(5,5), stride=(2,2), padding=(2,2)
	BatchNorm2d -> LeakyReLU(0.2)
ConvTranspose2d	kernels=256, kernelSize=(5,5), stride=(2,2), padding=(2,2)
	BatchNorm2d -> LeakyReLU(0.2)
ConvTranspose2d	kernels=256, kernelSize=(5,5), stride=(1,1), padding=(2,2)
	BatchNorm2d -> Tanh()

Table 2: Decoder network

A .3.2 Training details

For MUG, we choose $|\mathbf{Z}| = 512$, $|\mathbf{Q}| = K \times 12$ and $|\mathbf{P}| = K \times 12$ and for sprites $|\mathbf{Z}| = 256$, $|\mathbf{Q}| = K \times 6$ and $|\mathbf{P}| = K \times 6$, 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 [3] optimiser with a learning rate of $2e^{-4}$ and a batch size of 24. We use Pytorch [4] for the implementation. The code will be made available on publication. We train all our models on Nvidia GeForce RTX 2080 GPUs.

A .3.3 Results

We provide extended qualitative samples of our model on MUG and sprites dataset. Figure (2) shows results of conditional sequence generation, Figure (3, 4) shows results of image to sequence generation. Here we generate 16 frames in future conditioned on an initial starting frame. The results demonstrate our model can generate long term sequences, Figure (5) shows results of motion swapping.

Content and Motion Architecture					
Content		Position		Momentum	
LSTM	in=h, out=Z	Linear	in=h+k, out=V	Linear	in=h+k, out=P
Linear_{μ}	in=Z, out=Z	BatchNorm1d -> LeakyReLU(0.2)		BatchNorm1d -> LeakyReLU(0.2)	
$\text{Linear}_{\log \sigma}$	in=Z, out=Z	Linear	in=V, out=V	Linear	in=P, out=P
		BatchNorm1d -> LeakyReLU(0.2)		BatchNorm1d -> LeakyReLU(0.2)	
		Linear_{μ}	in=V, out=V	TCN	kernelSize=4, pad=3, stride=1
		$\text{Linear}_{\log \sigma}$	in=V, out=V	Linear_{μ}	in=P, out=P
				$\text{Linear}_{\log \sigma}$	in=P, out=P

Table 3: Content and Motion network. TCN stands for temporal convolution network.

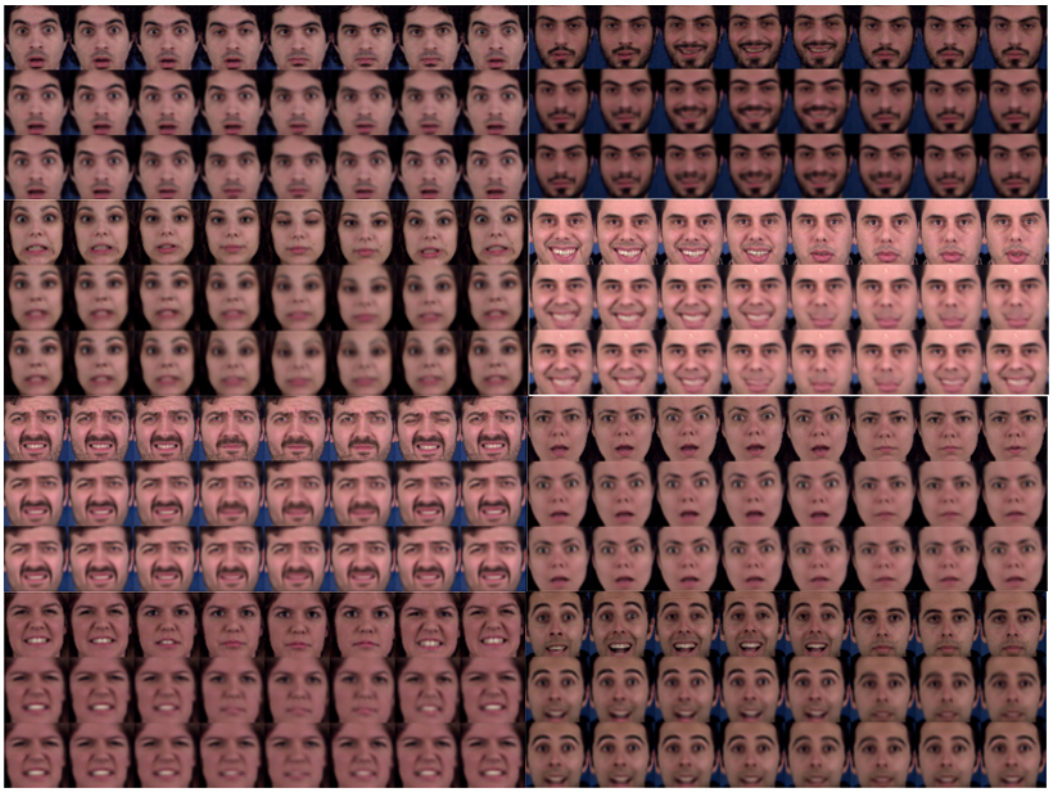
References

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

[2] Li Yingzhen and Stephan Mandt. Disentangled sequential autoencoder. In *International Conference on Machine Learning*, pages 5670–5679. PMLR, 2018.

[3] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[4] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.



(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

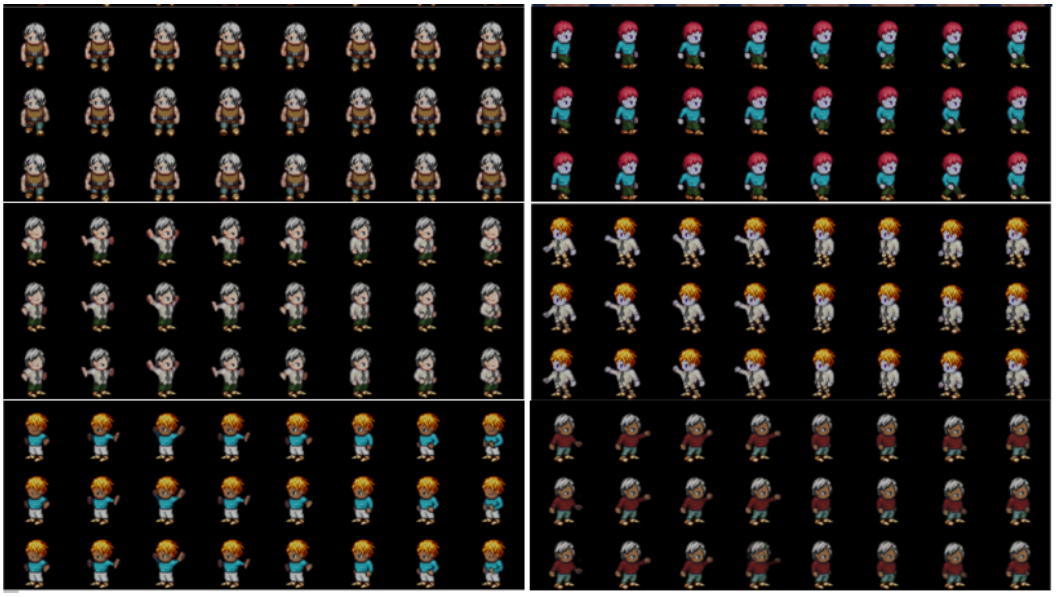


Figure 2: Conditional Sequence Generation. The first row is the original sequence, second row is a reconstructed sequence and third is generated by an action of dynamical model on the first time frame

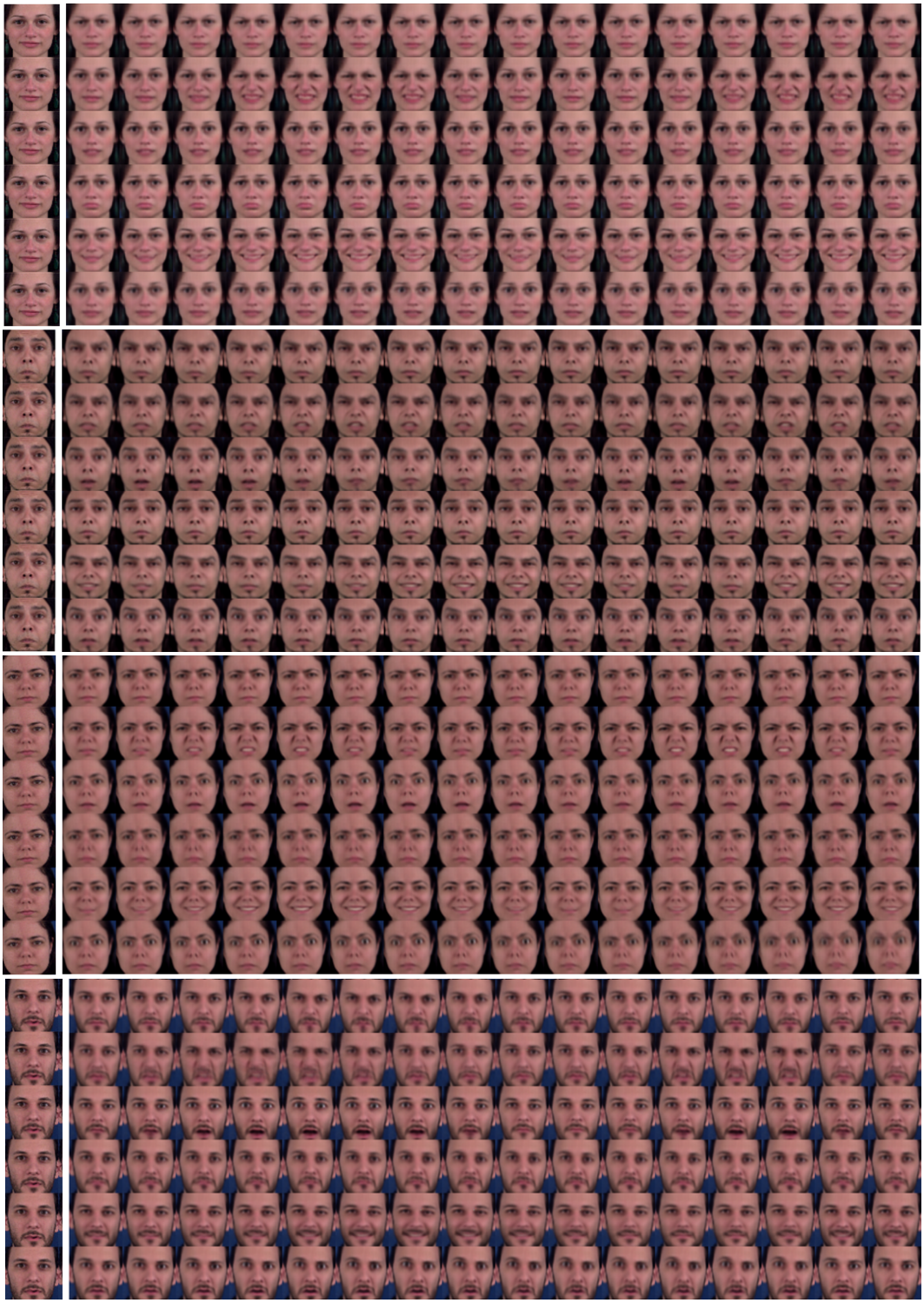


Figure 3: Image to Sequence generation. We generate dynamics of different action from a given image. Each row is a unique action generated by the operator associated with that action.

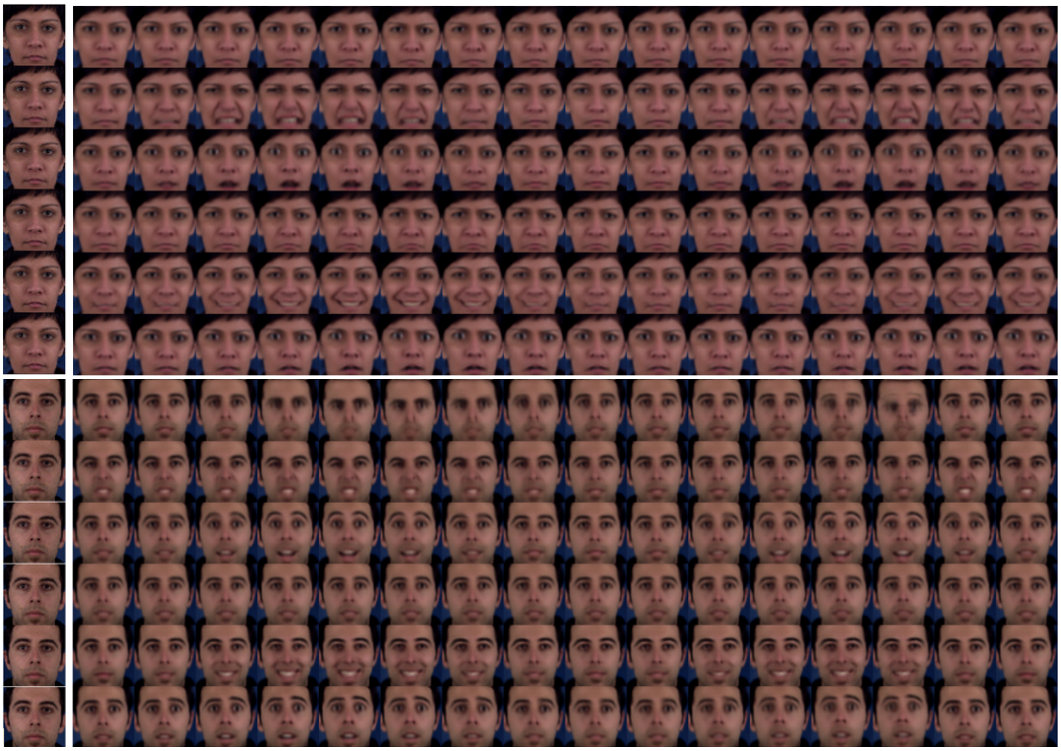


Figure 4: Image to Sequence generation. We generate dynamics of different action from a given image. Each row is a unique action generated by the operator associated with that action.

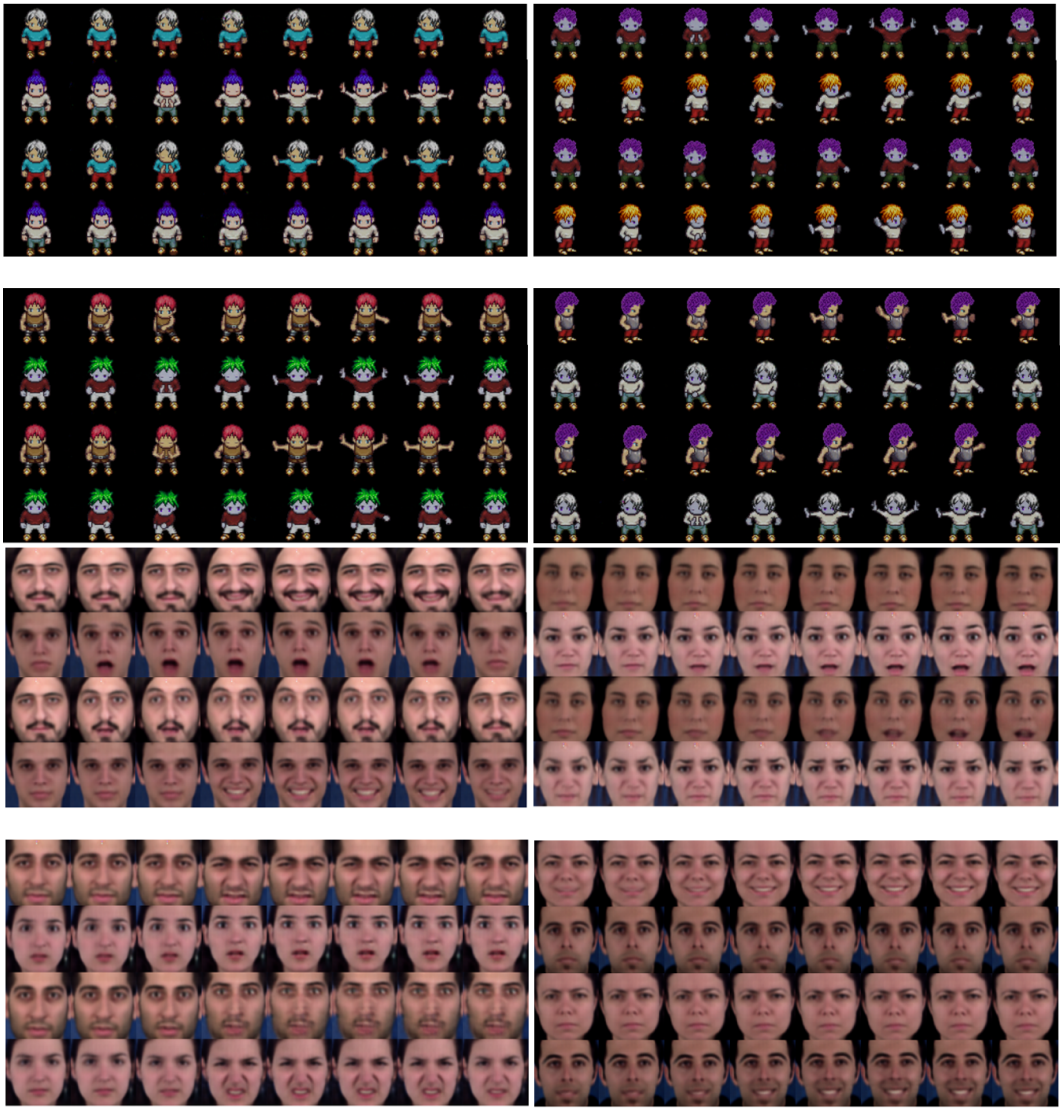


Figure 5: Motion Swapping. In each patch the first two rows are original sequence and the next two rows obtained by swapping motion variables of two sequences.