A. Lower Bound Derivation

For brevity we omit conditioning on control inputs $u_{1:T}$.

$$
\begin{aligned}
\log p(x_{1:T}) &= \log \int_{z_{1:T}} \int_{s_{2:T}} q_\phi(z_{1:T}, s_{2:T} \mid x_{1:T}) \\
&\qquad \frac{p_\theta(x_{1:T} \mid z_{1:T}) p_\theta(z_{1:T}, s_{2:T})}{q_\phi(z_{1:T}, s_{2:T} \mid x_{1:T})} \\
&\geq \int_{z_{1:T}} \int_{s_{2:T}} q_\phi(z_{1:T}, s_{2:T} \mid x_{1:T}) \\
&\qquad \log \frac{p_\theta(x_{1:T} \mid z_{1:T}) p_\theta(z_{1:T}, s_{2:T})}{q_\phi(z_{1:T}, s_{2:T} \mid x_{1:T})} \\
&= \int_{z_{1:T}} \int_{s_{2:T}} q_\phi(z_{1:T}, s_{2:T} \mid x_{1:T}) \log p_\theta(x_t \mid z_t, s_t) \\
&\quad + \int_{z_{1:T}} \int_{s_{2:T}} q_\phi(z_{1:T}, s_{2:T} \mid x_{1:T}) \\
&\qquad \log \frac{p_\theta(z_{1:T}, s_{2:T})}{q_\phi(z_{1:T}, s_{2:T} \mid x_{1:T})} \\
&= \int_{z_1} \int_{s_2} \cdots \int_{s_T} \int_{z_T} q(z_1 \mid x_{1:T}) q(s_2 \mid z_1, x_{1:T}) \dots \\
&\quad \dots q(s_T \mid z_{T-1}, s_{T-1}, x_{1:T}) q(z_T \mid z_{T-1}, s_T, x_{1:T}) \\
&\quad \log p_\theta(x_1 \mid z_t, s_t) \dots p_\theta(x_T \mid z_T, s_T) \\
&\quad - \mathrm{KL}(q(z_{1:T}, s_{2:T} \mid x_{1:T}) \mid\mid p_\theta(z_{1:T}, s_{2:T})) \\
&= \mathbb{E}_{z_1 \sim q(z_1 \mid \cdot)}[p(x_1 \mid z_1)] \\
&\quad + \sum_{t=2}^{T} \mathbb{E}_{s_t \sim q(s_t \mid \cdot)} \left[\mathbb{E}_{z_t \sim q(z_t \mid \cdot)}[p(x_t \mid z_t, s_t)] \right] \\
&\quad - \mathrm{KL}(q(z_{1:T}, s_{2:T} \mid x_{1:T}) \mid\mid p(z_{1:T}, s_{2:T}))
\end{aligned}
$$

A.1. Factorization of the KL Divergence

The dependencies on data $x_{1:T}$ and $u_{1:T}$ as well as parameters ϕ and θ are omitted in the following for convenience.

$$
\begin{aligned}
&\mathrm{KL}(q(z_1, s_2, \dots, s_T, z_T) \mid\mid p(z_1, s_2, \dots, s_T, z_T)) \\
&\textit{(Factorization of the variational approximation)} \\
&= \int_{z_1} \int_{s_2} \cdots \int_{s_T} \int_{z_T} q(z_1) q(s_2 \mid z_1) \dots \\
&\qquad \dots q(s_T \mid z_{T-1}, s_{T-1}) q(z_T \mid z_{T-1}, s_T) \\
&\quad \log \frac{q(z_1) q(s_2 \mid z_1) \dots q(z_T \mid z_{T-1}, s_T)}{p(z_1, s_2, \dots, s_T, z_T)} \\
&\textit{(Factorization of the prior)} \\
&= \int_{z_1} \int_{s_2} \cdots \int_{s_T} \int_{z_T} q(z_1) q(s_2 \mid z_1) \dots \\
&\qquad \dots q(s_T \mid z_{T-1}, s_{T-1}) q(z_T \mid z_{T-1}, s_T) \\
&\quad \log \frac{q(z_1) q(s_2 \mid z_1) \dots q(z_T \mid z_{T-1}, s_T)}{p(z_1) p(s_2 \mid z_1) \dots p(z_T \mid z_{T-1}, s_T)} \\
&\textit{(Expanding the logarithm by the product rule)} \\
&= \int_{z_1} q(z_1) \log \frac{q(z_1)}{p(z_1)} \\
&\quad + \int_{z_1} \int_{s_2} q(z_1) q(s_2 \mid z_1) \log \frac{q(s_2 \mid z_1)}{p(s_2 \mid z_1)} \\
&\quad + \sum_{t=2}^{T} \int_{z_1} \int_{s_2} \cdots \int_{s_T} \int_{z_T} q(z_1) q(s_2 \mid z_1) \dots \\
&\qquad \dots q(z_T \mid z_{T-1}, s_T) \log \frac{q(z_t \mid z_{t-1}, s_t)}{p(z_t \mid z_{t-1}, s_t)} \\
&\quad + \sum_{t=3}^{T} \int_{z_1} \int_{s_2} \cdots \int_{s_T} \int_{z_T} q(z_1) q(s_2 \mid z_1) \dots \\
&\qquad \dots q(z_T \mid z_{T-1}, s_T) \log \frac{q(s_t \mid z_{t-1}, s_{t-1})}{p(s_t \mid z_{t-1}, s_{t-1})} \\
&\textit{(Ignoring constants)} \\
&= \mathrm{KL}(q(z_1) \mid\mid p(z_1)) \\
&\quad + \mathbb{E}_{z_1 \sim q(z_1)}[\mathrm{KL}(q(s_2 \mid z_1) \mid\mid p(s_2 \mid z_1))] \\
&\quad + \sum_{t=2}^{T-1} \mathbb{E}_{z_{t-1} \sim q(z_{t-1} \mid \cdot)} \Big[\mathbb{E}_{s_t \sim q(s_t \mid \cdot)} \big[\\
&\qquad \mathrm{KL}(q(z_t \mid z_{t-1}, s_t) \mid\mid p(z_t \mid z_{t-1}, s_t)) \big] \Big] \\
&\quad + \sum_{t=3}^{T-1} \mathbb{E}_{s_{t-1} \sim q(s_{t-1} \mid \cdot)} \Big[\mathbb{E}_{z_{t-1} \sim q(z_{t-1} \mid \cdot)} \big[\\
&\qquad \mathrm{KL}(q(s_t \mid z_{t-1}, s_{t-1}) \mid\mid p(s_t \mid z_{t-1}, s_{t-1})) \big] \Big]
\end{aligned}
$$

B. Comparison to Previous Models

We want to emphasize some subtle differences to previously proposed architectures that make an empirical difference, in particular for the case when s_t is chosen to be continuous. In Watter et al. (2015) and Karl et al. (2017a), the latent space is already used to draw transition matrices, however they do not extract features such as walls or joint constraints. There are a few key differences to our approach.

First, our latent switching variables s_t are only involved in predicting the current observation x_t through the transition selection process. The likelihood model therefore does not need to learn to ignore some input dimensions which are only helpful for reconstructing future observations but not the current one.

There is also a clearer restriction on how s_t and z_t may interact: s_t may now only influence z_t by determining the dynamics, while previously z_t influenced both the choice of transition function as well as acted inside the transition itself. These two opposing roles lead to conflicting gradients as to what should be improved. Furthermore, the learning signal for s_t is rather weak so that scaling down the KL-regularization was necessary to detect good features.

Table 2: Dimensionality of environments.

Dimensionality of	Observation Space	Control Input Space	Ground Truth State Space
Reacher	7	2	9
Hopper	8	3	15
Multi Agent Maze	6	6	12
Image Ball in Box	32×32	0	4
FitzHugh-Nagumo	2	1	2

Lastly, a locally linear transition may not be a good fit for variables determining dynamics as such variables may change very abruptly. Therefore, it might be beneficial to have part of the latent space evolve according to locally linear dynamics and other parts according to a general purpose neural network transition. Overall, our structure of choosing a transition gives a stronger bias towards learning such features when compared to other methods.

C. Training

Overall, training the Concrete distribution has given us the biggest challenge as it was very susceptible to various hyperparameters. We made use of the fact that we can use a different temperature for the prior and approximate posterior (Maddison et al., 2017) and we do independent hyperparameter search over both. For us, the best values were 0.75 for the posterior and 2 for the prior. Additionally, we employ an exponential annealing scheme for the temperature hyperparameter of the Concrete distribution. This leads to a more uniform combination of base matrices early in training which has two desirable effects. First, all matrices are scaled to a similar magnitude, making initialization less critical. Second, the model initially tries to fit a globally linear model, leading to a good starting state for optimization.
With regards to optimizing the KL-divergence, there is no closed-form analytical solution for two Concrete distributions. We therefore had to resort to a Monte Carlo estimation with n samples where we tried n between 1 and 1000. While using a single samples was (numerically) unstable, using a large number of samples also didn't result in observable performance improvements. We therefore settled on using 10 samples for all experiments.

In all experiments, we train everything end-to-end with the ADAM optimizer (Kingma & Ba, 2015). We start with learning rate of $5\mathrm{e}{-4}$ and use an exponential decay schedule with rate $\lambda \in \{0.95, 0.97, 0.98\}$ (see table 3) every 2000 iterations.

D. Experimental Setup

D.1. Data Set Generation

D.1.1. ROBOSCHOOL REACHER

To generate data, we follow a Uniform distribution $\mathcal{U} \sim [-1, 1]$ as the exploration policy. Before we record data, we take 20 warm-up steps in the environment to randomize our starting state. We take the data as is without any other preprocessing.

D.1.2. MULTI AGENT MAZE

Observations are normalized to be in $[-1, 1]$. Both position and velocity is randomized for the starting state. We again follow a Uniform distribution $\mathcal{U} \sim [-1, 1]$ as the exploration policy.

D.2. Network Architecture

For most networks, we use MLPs implemented as residual nets (He et al., 2016) with ReLU activations.

Networks used for the reacher and maze experiments.

- $q_{\text{meas}}(z_t \mid x_{\geq t}, u_{\geq t})$: MLP consisting of two residual blocks with 256 neurons each. We only condition on the current observation x_t although we could condition on the entire sequence. This decision was taken based on empirical results.

- $q_{\text{trans}}(z_t \mid z_{t-1}, u_{t-1}, s_t)$: In the case of Concrete random variables, we just combine the base matrices and apply the transition dynamics to z_{t-1}. For the Normal case, the combination of matrices is preceded by a linear combination with softmax activation. (see equation 15)

- $q_{\text{meas}}(s_t \mid x_{\geq t}, u_{\geq t})$: is implemented by a backward LSTM with 256 hidden units. We reuse the preprocessing of $q_{\text{meas}}(z_t \mid x_t)$ and take the last hidden layer of that network as the input to the LSTM.

- $q_{\text{trans}}(s_t \mid s_{t-1}, z_{t-1}, u_{t-1})$: MLP consisting of one residual block with 256 neurons.

Table 3: Overview of hyperparameters.

	Multi Agent Maze	Reacher	Image Ball in Box	FitzHugh-Nagumo
# episodes	50000	20000	5000	100
episode length	20	30	20	400
batch size	256	128	256	32
dimension of z	32	16	8	4
dimension of s	16	8	8	8
posterior temperature	0.75	0.75	0.67	0.67
prior temperature	2	2	2	2
temperature annealing steps	100	100	100	100
temperature annealing rate	0.97	0.97	0.98	0.95
β (KL-scaling of switching variables)	0.1	0.1	0.1	0.1

- $q_{\text{initial}}(w \mid x_{1:T}, u_{1:T})$: MLP consisting of two residual block with 256 neurons optionally followed by a backward LSTM. We only condition on the first 3 or 4 observations for our experiments.
- $q_{\text{initial}}(s_2 \mid x_{1:T}, u_{1:T})$: The first switching variable in the sequence has no predecessor. We therefore require a replacement for $q_{\text{trans}}(s_t \mid s_{t-1}, z_{t-1}, u_{t-1})$ in the first time step, which we achieve by independently parameterizing another MLP.
- $p(x_t \mid z_t)$: MLP consisting of two residual block with 256 neurons.
- $p(z_t \mid z_{t-1}, u_{t-1}, s_t)$: Shared parameters with $q_{\text{trans}}(z_t \mid z_{t-1}, u_{t-1}, s_t)$.
- $p(s_t \mid s_{t-1}, z_{t-1}, u_{t-1})$: Shared parameters with $q_{\text{trans}}(s_t \mid s_{t-1}, z_{t-1}, u_{t-1})$.

We use the same architecture for the image ball in a box experiment, however we increase number of neurons of $q_{\text{meas}}(z_t \mid x_{\geq t}, u_{\geq t})$ to 1024.

For the FitzHugh-Nagumo model we downsize our model and restrict all networks to a single hidden layer with 128 neurons.

E. On Scaling Issues of Switching Linear Dynamical Systems

Let's consider a simple representation of a ball in a rectangular box where its state is represented by its position and velocity. Given a small enough Δt, we can approximate the dynamics decently by just 3 systems: no interaction with the wall, interaction with a vertical or horizontal wall (ignoring the corner case of interacting with two walls at the same time). Now consider the growth of required base systems if we increase the number of balls in the box (even if these balls cannot interact with each other). We would require a system for all combinations of a single ball's possible states: 3^2. This will grow exponentially with the number of balls in the environment.

One way to alleviate this problem that requires only a linear growth in base systems is to independently turn individual systems on and off and let the resulting system be the sum of all activated systems. A base system may then represent solely the transition for a single ball being in specific state, while the complete system is then a combination of N such systems where N is the number of balls. Practically, this can be achieved by replacing the *softmax* by a *sigmoid* activation function or by replacing the categorical variable s of dimension M by M Bernoulli variables indicating whether a single system is active or not. We make use of this parameterization in the multiple bouncing balls in a maze environment.

Theoretically, a preferred approach would be to disentangle multiple systems (like balls, joints) and apply transitions only to their respective states. This, however, would require a proper and unsupervised separation of (mostly) independent components. We defer this to future work.

F. Further Results

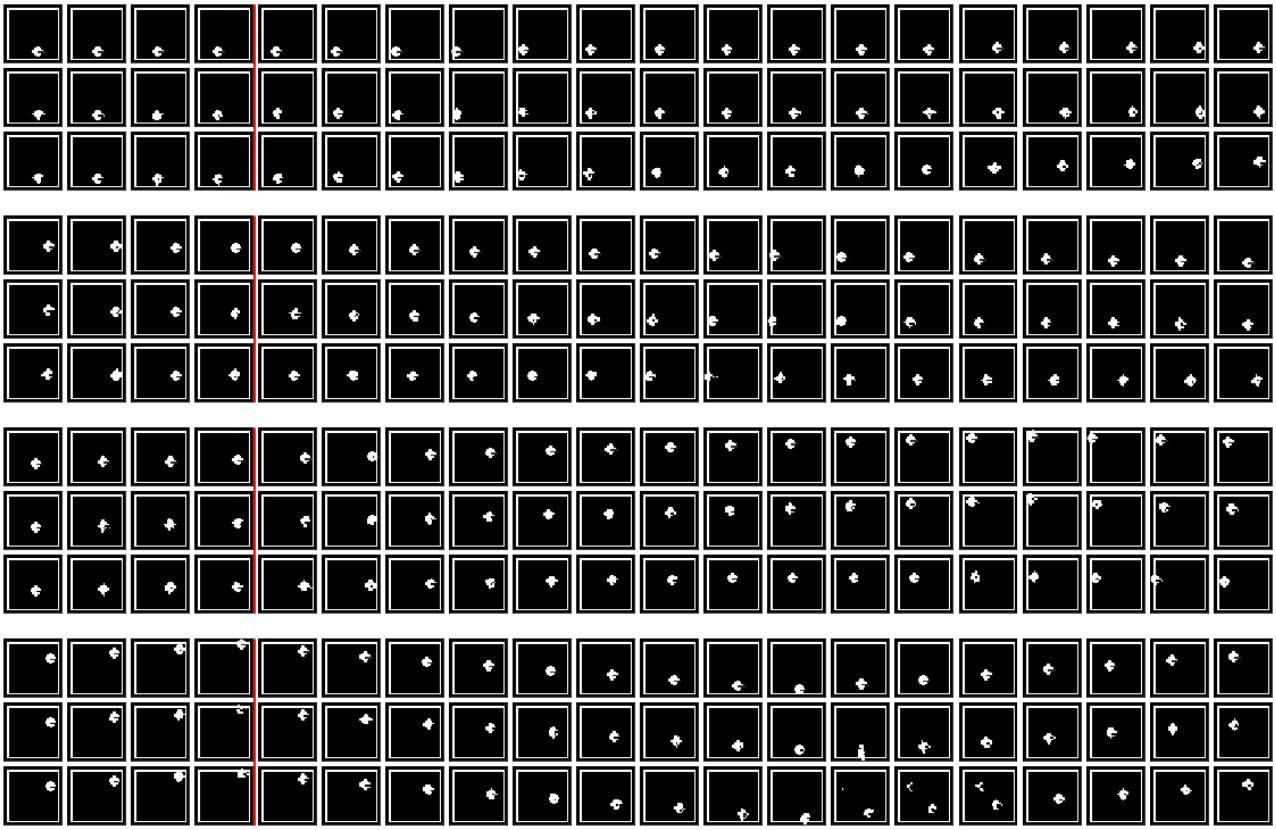

Figure 8: First row: data, second row: reconstructions, third row: predictions. The first 4 steps are used to find a stable starting state, predictions start with step 5 (after the red line). These results have been produced with Gaussian distributed switching variables.