APPENDICES FOR "ACTION-SUFFICIENT STATE REPRESENTATION LEARNING FOR CONTROL WITH STRUCTURAL CONSTRAINTS"

A PROOF OF PROPOSITION 1

We first give the definitions of the Markov condition and the faithfulness assumption, which will be used in the proof.

Definition 1 (Global Markov Condition (Spirtes et al., 1993; Pearl, 2000)). *The distribution p over $\mathbf{V}$ satisfies the global Markov property on graph G if for any partition (A, B, C) such that B d-separates A from C,*

$$p(A, C|B) = p(A|B)p(C|B).$$

Definition 2 (Faithfulness Assumption (Spirtes et al., 1993; Pearl, 2000)). *There are no independencies between variables that are not entailed by the Markov Condition.*

Below, we give the proof of Proposition 1.

Proof. We first show that if $s_{i,t} \in \vec{s}_t^{\,\text{ASR}}$, then it has a direct or indirect edge to $r_{t+\tau}$.

We prove it by contradiction. Suppose that $s_{i,t}$ does not have a direct or indirect edge to $r_{t+\tau}$. According to the Markov assumption, $s_{i,t}$ is independent of a_t given R. Hence, $s_{i,t}$ is not necessary for decision making, and thus $s_{i,t}$ is not a dimension in $\vec{s}_t^{\,\text{ASR}}$, which contradicts to the assumption. Since we have a contradiction, it must be that $s_{i,t}$ has a direct or indirect edge to $r_{t+\tau}$.

We next show that if $s_{i,t}$ has a direct or indirect edge to $r_{t+\tau}$, then $s_{i,t} \in \vec{s}_t^{\,\text{ASR}}$.

Similarly, by contradiction suppose that $s_{i,t}$ is not a dimension in $\vec{s}_t^{\,\text{ASR}}$. It means that $s_{i,t}$ is independent on a_t given R and some other variables. Then according to the faithfulness assumption, $s_{i,t}$ does not have a direct or indirect edge to $r_{t+\tau}$, which contradicts to the assumption.

□

B MINIMALITY OF THE REPRESENTATION

In this section, we give the detailed derivation of the minimality of the state representation given in Section 2.1.

We achieve minimality of the representation by minimizing conditional mutual information between observed high-dimensional signals $\mathbf{y}_t$, where $\mathbf{y}_t = \{o_t^T, r_t^T\}$, and the ASR $\tilde{\vec{s}}_t^{\,\text{ASR}}$ at time t given data at previous time instances, and meanwhile minimizing the dimensionality of ASRs with sparsity constraints:

$$\lambda_1 \sum_{t=2}^{T} I(\mathbf{y}_t; \tilde{\vec{s}}_t^{\,\text{ASR}} | \mathbf{y}_{1:t-1}, a_{1:t-1}, \tilde{\vec{s}}_{t-1}) + \lambda_2 \|\tilde{D}^{\,\text{ASR}}\|_1.$$

Note that in the above conditional mutual information, we need to conditional on the previous states $\tilde{\vec{s}}_{t-1}$, instead of $\tilde{\vec{s}}_{t-1}^{\,\text{ASR}}$, which two give different conditional mutual information. It can be shown by contradiction. Suppose $I(\mathbf{y}_t; \tilde{\vec{s}}_t^{\,\text{ASR}} | \mathbf{y}_{1:t-1}, a_{1:t-1}, \tilde{\vec{s}}_{t-1}) = I(\mathbf{y}_t; \tilde{\vec{s}}_t^{\,\text{ASR}} | \mathbf{y}_{1:t-1}, a_{1:t-1}, \tilde{\vec{s}}_{t-1}^{\,\text{ASR}})$, and denote $\tilde{\vec{s}}^{\,C} = \tilde{\vec{s}} \backslash \tilde{\vec{s}}^{\,\text{ASR}}$. Then the equivalence implies that $\tilde{\vec{s}}_{t-1}^{\,C}$ is independent of o_t (where $o_t \in \mathbf{y}_t$) given $\{\mathbf{y}_{1:t-1}, a_{1:t-1}, \tilde{\vec{s}}_{t-1}^{\,\text{ASR}}\}$. It is obviously violated for the example given in Figure 1, where $\tilde{\vec{s}}^{\,C} = s_1$ and $\tilde{\vec{s}}^{\,\text{ASR}} = \{s_2, s_3\}$, and $s_{1,t-1}$ is dependent on o_t given $\{\mathbf{y}_{1:t-1}, a_{1:t-1}, s_{2,t-1}, s_{3,t-1}\}$. Hence, conditioning on $\tilde{\vec{s}}_{t-1}$ and $\tilde{\vec{s}}_{t-1}^{\,\text{ASR}}$ give different conditional mutual information. Therefore, in the above conditional mutual information, we need to condition on the previous states $\tilde{\vec{s}}_{t-1}$.

Moreover, the conditional mutual information $I(\mathbf{y}_t; \tilde{\vec{s}}_t^{\text{ASR}} | \mathbf{y}_{1:t-1}, a_{1:t-1}, \tilde{\vec{s}}_{t-1})$ can be upper bound by a KL-divergence:

$$
\begin{aligned}
& I(\mathbf{y}_t; \tilde{\vec{s}}_t^{\text{ASR}} | \mathbf{y}_{1:t-1}, a_{1:t-1}, \tilde{\vec{s}}_{t-1}) \\
& = I(\mathbf{y}_t; \tilde{\vec{s}}_t^{\text{ASR}}, \{\mathbf{y}_{1:t-1}, a_{1:t-1}, \tilde{\vec{s}}_{t-1}\}) - I(\mathbf{y}_t; \{\mathbf{y}_{1:t-1}, a_{1:t-1}, \tilde{\vec{s}}_{t-1}\}) \\
& = \big[H(\mathbf{y}_t) - H(\mathbf{y}_t | \tilde{\vec{s}}_t^{\text{ASR}}, \mathbf{y}_{1:t-1}, a_{1:t-1}, \tilde{\vec{s}}_{t-1})\big] - \big[H(\mathbf{y}_t) - H(\mathbf{y}_t | \mathbf{y}_{1:t-1}, a_{1:t-1}, \tilde{\vec{s}}_{t-1})\big] \\
& = H(\mathbf{y}_t | \mathbf{y}_{1:t-1}, a_{1:t-1}, \tilde{\vec{s}}_{t-1}) - H(\mathbf{y}_t | \tilde{\vec{s}}_t^{\text{ASR}}, \mathbf{y}_{1:t-1}, a_{1:t-1}, \tilde{\vec{s}}_{t-1}) \\
& = -p(\mathbf{y}_t | \mathbf{y}_{1:t-1}, a_{1:t-1}, \tilde{\vec{s}}_{t-1}) \log p(\mathbf{y}_t | \mathbf{y}_{1:t-1}, a_{1:t-1}, \tilde{\vec{s}}_{t-1}) \\
& \quad + p(\mathbf{y}_t | \tilde{\vec{s}}_t^{\text{ASR}}, \mathbf{y}_{1:t-1}, a_{1:t-1}, \tilde{\vec{s}}_{t-1}) \log p(\mathbf{y}_t | \tilde{\vec{s}}_t^{\text{ASR}}, \mathbf{y}_{1:t-1}, a_{1:t-1}, \tilde{\vec{s}}_{t-1}) \\
& \leq -p(\mathbf{y}_t | \tilde{\vec{s}}_t^{\text{ASR}}, \mathbf{y}_{1:t-1}, a_{1:t-1}, \tilde{\vec{s}}_{t-1}) \log p(\mathbf{y}_t | \mathbf{y}_{1:t-1}, a_{1:t-1}, \tilde{\vec{s}}_{t-1}) \\
& \quad + p(\mathbf{y}_t | \tilde{\vec{s}}_t^{\text{ASR}}, \mathbf{y}_{1:t-1}, a_{1:t-1}, \tilde{\vec{s}}_{t-1}) \log p(\mathbf{y}_t | \tilde{\vec{s}}_t^{\text{ASR}}, \mathbf{y}_{1:t-1}, a_{1:t-1}, \tilde{\vec{s}}_{t-1}) \\
& = \int q_\phi(\tilde{\vec{s}}_t^{\text{ASR}} | \tilde{\vec{s}}_{t-1}, \mathbf{y}_{1:t}, a_{1:t-1}) \log \frac{q_\phi(\tilde{\vec{s}}_t^{\text{ASR}} | \tilde{\vec{s}}_{t-1}, \mathbf{y}_{1:t}, a_{1:t-1})}{p_\gamma(\tilde{\vec{s}}_t^{\text{ASR}} | \tilde{\vec{s}}_{t-1}, a_{t-1}; D_{\vec{s}}, D_{a \to \vec{s}})} \, d\tilde{\vec{s}}_t^{\text{ASR}} \\
& = \text{KL}\big(q_\phi(\tilde{\vec{s}}_t^{\text{ASR}} | \tilde{\vec{s}}_{t-1}, \mathbf{y}_{1:t}, a_{1:t-1}) \| p_\gamma(\tilde{\vec{s}}_t^{\text{ASR}} | \tilde{\vec{s}}_{t-1}, a_{t-1}; D_{\vec{s}}, D_{a \to \vec{s}})\big),
\end{aligned}
$$

with p_γ being the transition dynamics of $\tilde{\vec{s}}_t$ with parameters γ.

C Assumptions of Proposition 2

To show the identifiability of the model in the linear case, we make the following assumptions:

A1. $d_o + d_r \geq d_s$, where $|o_t| = d_o$, $|r_t| = d_r$, and $|s_t| = d_s$.
A2. $(D_{\vec{s} \to o}^\top, D_{\vec{s} \to r}^\top)^\top$ is full column rank and $D_{\vec{s}}$ is full rank.
A3. The control signal a_t is i.i.d. and the state $\vec{s}_t$ is stationary.
A4. The process noise has a unit variance, i.e., var$(\eta_t) = I$.

D Proof of Proposition 2

Proof. The proof of the linear case without control signals has been shown in Zhang & Hyvärinen (2011). Below, we give the identifiability proof in the linear-Gaussian case with control signals:

$$
\begin{cases}
o_t = D_{\vec{s} \to o} \vec{s}_t + e_t, \\
r_{t+1} = D_{\vec{s} \to r} \vec{s}_t + D_{a \to r} a_t + \epsilon_{t+1}, \\
\vec{s}_t = D_{\vec{s}} \vec{s}_{t-1} + D_{a \to \vec{s}} a_{t-1} + \eta_t.
\end{cases}
\tag{5}
$$

Let $\mathbf{y}_{t+1} = [o_t^\top, r_{t+1}^\top]^\top$, $\ddot{D}_{\vec{s} \to o} = [D_{\vec{s} \to o}^\top, D_{\vec{s} \to r}^\top]^\top$, $\ddot{D}_{a \to r} = [\vec{0}^\top, D_{a \to r}^\top]^\top$, and $\ddot{e}_t = [e_t^\top, \epsilon_{t+1}^\top]^\top$. Then the above equation can be represented as:

$$
\begin{cases}
\mathbf{y}_t = \ddot{D}_{\vec{s} \to o} \vec{s}_t + \ddot{D}_{a \to r} a_t + \ddot{e}_t, \\
\vec{s}_t = D_{\vec{s}} \vec{s}_{t-1} + D_{a \to \vec{s}} a_{t-1} + \eta_t.
\end{cases}
\tag{6}
$$

Because the dynamic system is linear and Gaussian, we make use of the second-order statistics of the observed data to show the identifiability. We first consider the cross-covariance between $\mathbf{y}_{t+k}$ and a_t:

$$
\begin{cases}
\text{Cov}(\mathbf{y}_{t+k}, a_t) = \ddot{D}_{\vec{s} \to o} D_{\vec{s}}^{k-1} D_{a \to \vec{s}} \cdot \text{Var}(a_t), & \text{if } k > 0. \\
\text{Cov}(\mathbf{y}_{t+k}, a_t) = \ddot{D}_{a \to r} \cdot \text{Var}(a_t), & \text{if } k = 0.
\end{cases}
\tag{7}
$$

Thus, from the cross-covariance between $\mathbf{y}_{t+k}$ and a_t, we can identify $\ddot{D}_{\vec{s} \to o} D_{a \to \vec{s}}$, $\ddot{D}_{a \to r}$, and $\ddot{D}_{\vec{s} \to o} D_{\vec{s}}^k D_{a \to \vec{s}}$ for $k > 0$.

Next, we consider the auto-covariance function of $\vec{s}$. Define the auto-covariance function of $\vec{s}$ at lag k as $\mathbf{R}_{\vec{s}}(k) = \mathbb{E}[\vec{s}_t \vec{s}_{t+k}^\top]$, and similarly for $\mathbf{R}_{\mathbf{y}}(k)$. Clearly, $\mathbf{R}_{\vec{s}}(-k) = \mathbf{R}_{\vec{s}}(k)^\top$ and $\mathbf{R}_{\mathbf{y}}(-k) = \mathbf{R}_{\mathbf{y}}(k)^\top$. Then we have

$$
\begin{cases}
\mathbf{R}_{\vec{s}}(k) = \mathbf{R}_{\vec{s}}(k-1) \cdot D_{\vec{s}}^\top, & \text{if } k > 0, \\
\mathbf{R}_{\vec{s}}(k) = \mathbf{R}_{\vec{s}}^\top(1) \cdot D_{\vec{s}}^\top + D_{a \to \vec{s}} \text{Var}(a_{t-1}) D_{a \to \vec{s}}^\top + I, & \text{if } k = 0.
\end{cases}
\tag{8}
$$

Below, we first consider the case where $d_o + d_r = d_s$. Let $\tilde{\mathbf{y}}_t = \ddot{D}_{\vec{s} \to o} \vec{s}_t$, so $\mathbf{y}_t = \tilde{\mathbf{y}}_t + \ddot{D}_{a \to r} a_{t-1} + \ddot{e}_t$ and $\mathbf{R}_{\tilde{\mathbf{y}}}(k) = \ddot{D}_{\vec{s} \to o} \mathbf{R}_{\vec{s}_t}(k) \ddot{D}_{\vec{s} \to o}^\top$. $\mathbf{R}_{\tilde{\mathbf{y}}}(k)$ satisfies the recursive property:

$$
\begin{cases}
\mathbf{R}_{\tilde{\mathbf{y}}}(k) = \mathbf{R}_{\tilde{\mathbf{y}}}(k-1) \cdot \Omega^\top, & \text{if } k > 0, \\
\mathbf{R}_{\tilde{\mathbf{y}}}(k) = \mathbf{R}_{\tilde{\mathbf{y}}}^\top(1) \cdot \Omega^\top + \ddot{D}_{\vec{s} \to o}(D_{a \to \vec{s}} \text{Var}(a_{t-1}) D_{a \to \vec{s}}^\top + I) \ddot{D}_{\vec{s} \to o}^\top, & \text{if } k = 0,
\end{cases}
\tag{9}
$$

where $\Omega = \ddot{D}_{\vec{s}\to o} D_{\vec{s}} \ddot{D}_{\vec{s}\to o}^{-1}$.

Denote $S_k = \ddot{D}_{\vec{s}\to o} D_{\vec{s}}^{k-1} D_{a\to\vec{s}} \cdot \text{Var}(a_t)$. Then we can derive the recursive property for $\mathbf{R}_{\mathbf{y}}(k)$:

$$\begin{cases} \mathbf{R}_{\mathbf{y}}(k) = \mathbf{R}_{\mathbf{y}}(k-1)\cdot\Omega^\top - \ddot{D}_{a\to r} S_{k-1}^\top \Omega^\top + \ddot{D}_{a\to r} S_k^\top, & \text{if } k>1, \\ \mathbf{R}_{\mathbf{y}}(k) = \mathbf{R}_{\mathbf{y}}(k-1)\cdot\Omega^\top - \ddot{D}_{a\to r}\text{Var}^\top(a_t)\ddot{D}_{a\to r}^\top\Omega^\top - \Sigma_e\Omega^\top + \ddot{D}_{a\to r} S_k^\top, & \text{if } k=1, \\ \mathbf{R}_{\mathbf{y}}(k) = \mathbf{R}_{\mathbf{y}}^\top(1)\cdot\Omega^\top + \big(\ddot{D}_{a\to r}\text{Var}(a_t)\ddot{D}_{a\to r}^\top + \Sigma_e\big) & \\ \qquad\qquad + \ddot{D}_{\vec{s}\to o}(D_{a\to\vec{s}}\text{Var}(a_t)D_{a\to\vec{s}}^\top + I)\ddot{D}_{\vec{s}\to o}^\top, & \text{if } k=0. \end{cases}$$

When $k=2$, we have

$$\mathbf{R}_{\mathbf{y}}(2) = \mathbf{R}_{\mathbf{y}}(1)\cdot\Omega^\top - \ddot{D}_{a\to r}S_1^\top\Omega^\top + \ddot{D}_{a\to r}S_2^\top.$$

The above equation can be re-organized as

$$\big(\mathbf{R}_{\mathbf{y}}(2) - \ddot{D}_{a\to r}\cdot S_2^\top\big) = \big(\mathbf{R}_{\mathbf{y}}(1) - \ddot{D}_{a\to r}\cdot S_1^\top\big)\cdot\Omega^\top.$$

Because $\ddot{D}_{a\to r}$ and S_k are identifiable, and suppose $\big(\mathbf{R}_{\mathbf{y}}(1) - \ddot{D}_{a\to r}\cdot S_1^\top\big)$ is invertible, $\Omega = \ddot{D}_{\vec{s}\to o} D_{\vec{s}} \ddot{D}_{\vec{s}\to o}^{-1}$ is identifiable.

We further consider $\mathbf{R}_{\mathbf{y}}(0)$ and $\mathbf{R}_{\mathbf{y}}(1)$ and write down the following form:

$$\begin{aligned} & \begin{bmatrix} \mathbf{R}_{\mathbf{y}}(0) - \ddot{D}_{\vec{s}\to o}(D_{a\to\vec{s}}\text{Var}(a_{t-1})D_{a\to\vec{s}}^\top + I)\ddot{D}_{\vec{s}\to o}^\top \\ \mathbf{R}_{\mathbf{y}}(1) \end{bmatrix} \\ = \; & \begin{bmatrix} \mathbf{R}_{\mathbf{y}}^\top(1) \\ \mathbf{R}_{\mathbf{y}}(0) \end{bmatrix}\cdot\Omega^\top + \begin{bmatrix} \ddot{D}_{a\to r}\text{Var}(a_t)\ddot{D}_{a\to r}^\top \\ -\ddot{D}_{a\to r}\text{Var}^\top(a_t)\ddot{D}_{a\to r}^\top\Omega^\top + \ddot{D}_{a\to r}S_1^\top \end{bmatrix} + \Sigma_e\begin{bmatrix} I \\ -\Omega^\top \end{bmatrix}. \end{aligned}$$

From the above two equations we can then identify Σ_e and $\ddot{D}_{\vec{s}\to o}(D_{a\to\vec{s}}\text{Var}(a_{t-1})D_{a\to\vec{s}}^\top + I)\ddot{D}_{\vec{s}\to o}^\top$, and because $\ddot{D}_{\vec{s}\to o}D_{a\to\vec{s}}$ is identifiable, $\ddot{D}_{\vec{s}\to o}\ddot{D}_{\vec{s}\to o}^\top$ is identifiable.

In summary, we have shown the identifiability of $\ddot{D}_{a\to r}$, $\ddot{D}_{\vec{s}\to o}D_{a\to\vec{s}}$, $\ddot{D}_{\vec{s}\to o}D_{\vec{s}}^k D_{a\to\vec{s}}$, $\ddot{D}_{\vec{s}\to o}\ddot{D}_{\vec{s}\to o}^\top$, and Σ_e. Furthermore, $\ddot{D}_{\vec{s}\to o}$, $D_{\vec{s}}$, and $D_{a\to\vec{s}}$ are identified up to some orthogonal transformations. That is, suppose the model in Eq. (3) with parameters $(D_{\vec{s}\to o}, D_{\vec{s}\to r}, D_{a\to r}, D_{\vec{s}}, D_{a\to\vec{s}}, \Sigma_e, \Sigma_\epsilon)$ and that with $(\tilde{D}_{\vec{s}\to o}, \tilde{D}_{\vec{s}\to r}, \tilde{D}_{a\to r}, \tilde{D}_{\vec{s}}, \tilde{D}_{a\to\vec{s}}, \tilde{\Sigma}_{\tilde{e}}, \tilde{\Sigma}_{\tilde{\epsilon}})$ are observationally equivalent, we then have $\tilde{\ddot{D}}_{\vec{s}\to o} = \ddot{D}_{\vec{s}\to o}U$, $\tilde{D}_{a\to r} = D_{a\to r}$, $\tilde{D}_{\vec{s}} = U^\top D_{\vec{s}}U$, $\tilde{D}_{a\to\vec{s}} = D_{a\to\vec{s}}U$, $\tilde{\Sigma}_{\tilde{e}} = \Sigma_e$, and $\tilde{\Sigma}_{\tilde{\epsilon}} = \Sigma_\epsilon$, where U is an orthogonal matrix.

Next, we extend the above results to the case where $d_o + d_r > d_s$. Let $\ddot{D}_{\vec{s}\to o(i,\cdot)}^\top$ be the i-th row of $\ddot{D}_{\vec{s}\to o}$. Recall that $\ddot{D}_{\vec{s}\to o}$ is of full column rank. Then for any i, one can show that there always exist $d_s - 1$ rows of $\ddot{D}_{\vec{s}\to o}$, such that they, together with $\ddot{D}_{\vec{s}\to o(i,\cdot)}^\top$, form a $d_s \times d_s$ full-rank matrix, denoted by $\bar{\ddot{D}}_{\vec{s}\to o(i,\cdot)}$. Then from the observed data corresponding to $\bar{\ddot{D}}_{\vec{s}\to o(i,\cdot)}$, $\bar{\ddot{D}}_{\vec{s}\to o(i,\cdot)}$ is determined up to orthogonal transformations. Thus, $\ddot{D}_{\vec{s}\to o}$ is identified up to orthogonal transformations. Similarly, $D_{a\to r}$, $D_{\vec{s}}$, and $D_{a\to\vec{s}}$ are identified up to orthogonal transformations. Furthermore, $\text{Cov}(\ddot{D}_{\vec{s}\to o}\vec{s}_t + D_{a\to r}a_t)$ is determined by $\ddot{D}_{\vec{s}\to o}$, $\ddot{D}_{a\to r}$, $D_{\vec{s}}$, and $D_{a\to\vec{s}}$. Because $\text{Cov}(\mathbf{y}_t) = \text{Cov}(\ddot{D}_{\vec{s}\to o}\vec{s}_t + D_{a\to r}a_t) + \Sigma_{\ddot{e}}$, $\Sigma_{\ddot{e}}$ is identifiable.

One may further add sparsity constraints on $D_{\vec{s}\to o}$, $D_{\vec{s}\to r}$, $D_{\vec{s}}$, and $D_{a\to\vec{s}}$, to select more sparse structures among the equivalent ones. For example, one may add sparsity constraints on the columns of $D_{\vec{s}\to o}$. Note this corresponds to the mask on the elements of $\vec{s}_t$ in Eq. 2; if the full column is 0, then the corresponding dimension of $\vec{s}_t$ is not selected.

□

E More Estimation Details for General Nonlinear Models

The generative model p_θ can be further factorized as follows:

$$\begin{aligned} & \log p_\theta(\mathbf{y}_{1:T}|\tilde{\vec{s}}_{1:T}, a_{1:T-1}; D_{\vec{s}\to o}, D_{\vec{s}\to r}, D_{a\to r}) \\ = \; & \log p_\theta(o_{1:T}|\tilde{\vec{s}}_{1:T}; D_{\vec{s}\to o}) + \log p_\theta(r_{1:T}|\tilde{\vec{s}}_{1:T}, a_{1:T-1}; D_{\vec{s}\to r}, D_{a\to r}) \\ = \; & \textstyle\sum_{t=1}^T \log p_\theta(o_t|\tilde{\vec{s}}_t; D_{\vec{s}\to o}) + \log p_\theta(r_t|\tilde{\vec{s}}_{t-1}, a_{t-1}; D_{\vec{s}\to r}, D_{a\to r}), \end{aligned} \tag{10}$$

where both $p_\theta(o_t|\tilde{\vec{s}}_t; D_{\vec{s}\to o})$ and $p_\theta(r_t|\tilde{\vec{s}}_{t-1}, a_{t-1}; D_{\vec{s}\to r}, D_{a\to r})$ are modelled by mixture of Gaussians, with $D_{\vec{s}\to o}$ indicating the existence of edges from $\tilde{\vec{s}}_t$ to o_t and $D_{\vec{s}\to r}$ indicating the existence of edges from $\tilde{\vec{s}}_{t-1}$ to r_t.

The inference model $q_\phi(\tilde{\vec{s}}_{1:T}|\mathbf{y}_{1:T}, a_{1:T-1})$ is factorized as

$$\begin{aligned} & \log q_\phi(\tilde{\vec{s}}_{1:T}|\mathbf{y}_{1:T}, a_{1:T-1}) \\ = \quad & \log q_\phi(\tilde{\vec{s}}_1|\mathbf{y}_1, a_0) + \sum_{t=2}^{T} \log q_\phi(\tilde{\vec{s}}_t|\tilde{\vec{s}}_{t-1}, \mathbf{y}_{1:t}, a_{1:t-1}), \end{aligned}$$

where both $q_\phi(\tilde{\vec{s}}_1|\mathbf{y}_1, a_0)$ and $q_\phi(\tilde{\vec{s}}_t|\tilde{\vec{s}}_{t-1}, \mathbf{y}_{1:t}, a_{1:t-1})$ are modelled with mixture of Gaussians.

The transition dynamics p_γ is factorized as

$$\log p_\gamma(\tilde{\vec{s}}_{1:T}|a_{1:T-1}; D_{\vec{s}(\cdot,i)}, D_{a\to\vec{s}(\cdot,i)}) = \sum_{t=1}^{T} \log p_\gamma(\tilde{\vec{s}}_t|\tilde{\vec{s}}_{t-1}, a_{t-1}; D_{\vec{s}(\cdot,i)}, D_{a\to\vec{s}(\cdot,i)}), \tag{11}$$

with $\tilde{\vec{s}}_t|\tilde{\vec{s}}_{t-1}$ modelled with mixture of Gaussians.

Thus, the KL divergence can be represented as follows:

$$\begin{aligned} & \mathrm{KL}\big(q_\phi(\tilde{\vec{s}}_{1:T}|\mathbf{y}_{1:T}, a_{1:T-1})\|p_\gamma(\tilde{\vec{s}}_{1:T})\big) \\ = \quad & \mathrm{KL}\big(q_\phi(\tilde{\vec{s}}_1|\mathbf{y}_1, a_0)\|p_\gamma(\tilde{\vec{s}}_1)\big) + \sum_{t=2}^{T} \mathbb{E}_{q_\phi}\left[\mathrm{KL}\big(q_\phi(\tilde{\vec{s}}_t|\tilde{\vec{s}}_{t-1}, \mathbf{y}_{1:t}, a_{1:t-1})\|p_\gamma(\tilde{\vec{s}}_t|\tilde{\vec{s}}_{t-1})\big)\right]. \end{aligned} \tag{12}$$

In practice, KL divergence with mixture of Gaussians is hard to implement, so instead, we used the following objective function:

$$\begin{aligned} & \mathrm{KL}\big(q_\phi(\tilde{\vec{s}}_1|\mathbf{y}_1, a_0)\|p_{\gamma'}(\tilde{\vec{s}}_1)\big) + \sum_{t=2}^{T} \mathbb{E}_{q_\phi}\left[\mathrm{KL}\big(q_\phi(\tilde{\vec{s}}_t|\tilde{\vec{s}}_{t-1}, \mathbf{y}_{1:t}, a_{1:t-1})\|p_{\gamma'}(\tilde{\vec{s}}_t|\tilde{\vec{s}}_{t-1})\big)\right] \\ & +\lambda \sum_{t=1}^{T} \log p_\gamma(\tilde{\vec{s}}_t|\tilde{\vec{s}}_{t-1}, a_{t-1}; D_{\vec{s}(\cdot,i)}, D_{a\to\vec{s}(\cdot,i)}) \end{aligned} \tag{13}$$

where $p_{\gamma'}$ is a standard multivariate Gaussian $\mathcal{N}(\vec{0}, I_d)$.

F More Details for Policy Learning with ASRs

Algorithm 1 gives the procedure of model-free policy learning with ASRs in partially observable environments. Specifically, it starts from model initialization (line 1) and data collection with a random policy (line 2). Then it updates the environment model and identifies the set of ASRs with the collected data (line 3), after which, the main procedure of policy optimization follows. In particular, because we do not directly observe the states $\vec{s}_t$, on lines 8 and 12, we infer $q_\phi(\vec{s}^{\text{ASR}}_{t+1}|o_{\leq t+1}, r_{\leq t+1}, a_{\leq t})$ and sample $\vec{s}^{\text{ASR}}_{t+1}$ from the posterior. The sampled ASRs are then stored in the buffer (line 13). Furthermore, we randomly sample a minibatch of N transitions to optimize the policy (lines 14 and 15). One may perform various RL algorithms on the ASRs, such as deep deterministic policy gradient (DDPG (Lillicrap et al., 2015)) or Q-learning (Mnih et al., 2015).

Algorithm 2 presents the procedure of the classic Dyna algorithm with ASRs. Lines 17-22 make use of the learned environment model to predict the next step, including $\vec{s}^{\text{ASR}}_{t+1}$ and r_{t+1}, and update the Q function n times. Specifically, in our implementation, the hyper-parameter n is 20. Based on the learned model, the agent learns behaviors from imagined outcomes in the compact latent space, which helps to increase sample efficiency.

G Additional Experiments and Details

G.1 CarRacing Experiment

CarRacing (with an illustration in Figure 8) is a continuous control task with three continuous actions: steering left/right, acceleration, and brake. Reward is -0.1 every frame and $+1000/N$ for every track tile visited, where N is the total number of tiles in track. It is obvious that the CarRacing environment

Algorithm 1 Model-Free Policy Learning with ASRs in Partially Observable Environments

```
1:  Randomly initialize neural networks and initialize replay buffer $\mathcal{B}$.
2:  Apply random control signals and record multiple rollouts.
3:  Estimate the model given in (2) with the recorded data (according to Section 3).
4:  Identify indices of ASRs according to the learned graph structure and the criteria in Prop. 1.
5:  for episode = 1, ..., M do
6:      Initialize a random process $\mathcal{N}$ for action exploration.
7:      Receive initial observations $o_1$ and $r_1$.
8:      Infer the posterior $q_\phi(\vec{s}_1^{\text{ASR}}|o_1, r_1)$ and sample $\vec{s}_1^{\text{ASR}}$.
9:      for t = 1, ..., T do
10:         Select action $a_t = \pi(\vec{s}_t^{\text{ASR}}) + \mathcal{N}_t$ according to the current policy and exploration noise.
11:         Execute action $a_t$ and receive reward $r_{t+1}$ and observation $o_{t+1}$.
12:         Infer the posterior $q_\phi(\vec{s}_{t+1}^{\text{ASR}}|o_{\leq t+1}, r_{\leq t+1}, a_{\leq t})$ and sample $\vec{s}_{t+1}^{\text{ASR}}$.
13:         Store transition $(\vec{s}_t^{\text{ASR}}, a_t, r_{t+1}, \vec{s}_{t+1}^{\text{ASR}})$ in $\mathcal{B}$.
14:         Sample a random minibatch of $N$ transitions $(\vec{s}_i^{\text{ASR}}, a_i, r_{i+1}, \vec{s}_{i+1}^{\text{ASR}})$ from $\mathcal{B}$.
15:         Update network parameters using a specified RL algorithm (e.g., DQN or DDPG).
16:     end for
17: end for
```

Algorithm 2 Model-Based Policy Learning with ASRs in Partially Observable Environments

```
1:  Randomly initialize neural networks and initialize replay buffer $\mathcal{B}$.
2:  Apply random control signals and record multiple rollouts.
3:  Estimate the model given in (2) with the recorded data (according to Section 3).
4:  Identify indices of ASRs according to the learned graph structure and the criteria in Prop. 1.
5:  for episode = 1, ..., M do
6:      Initialize a random process $\mathcal{N}$ for action exploration.
7:      Receive initial observations $o_1$ and $r_1$.
8:      Infer the posterior $q_\phi(\vec{s}_1^{\text{ASR}}|o_1, r_1)$ and sample $\vec{s}_1^{\text{ASR}}$.
9:      for t = 1, ..., T do
10:         Select action $a_t = \pi(\vec{s}_t^{\text{ASR}}) + \mathcal{N}_t$ according to the current policy and exploration noise.
11:         Execute action $a_t$ and receive reward $r_{t+1}$ and observation $o_{t+1}$.
12:         Infer the posterior $q_\phi(\vec{s}_{t+1}^{\text{ASR}}|o_{\leq t+1}, r_{\leq t+1}, a_{\leq t})$ and sample $\vec{s}_{t+1}^{\text{ASR}}$.
13:         Store transition $(\vec{s}_t^{\text{ASR}}, \vec{s}_t, a_t, r_{t+1}, \vec{s}_{t+1}^{\text{ASR}}, \vec{s}_{t+1}, o_{t+1})$ in $\mathcal{B}$.
14:         Sample a random minibatch of $N$ transitions $(\vec{s}_i^{\text{ASR}}, a_i, r_{i+1}, \vec{s}_{i+1}^{\text{ASR}})$ from $\mathcal{B}$.
15:         Update network parameters using a specified RL algorithm (e.g., DQN or DDPG).
16:         Update the model given in (2) with the recorded data from $\mathcal{B}$ (according to Section 3).
17:         for p = 1, ..., n do
18:             Sample a random minibatch of pairs of $(\vec{s}_t, a_t)$ from $\mathcal{B}$.
19:             Predict $(\vec{s}_{t+1}^{\text{ASR}}, r_{t+1})$ according to the model given in (2).
20:             Update network parameters using a specified RL algorithm (e.g., DQN or DDPG).
21:         end for
22:     end for
23: end for
```

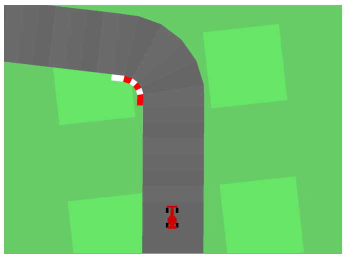

Figure 8: An illustration of Car Racing environment.

is partially observable: by just looking at the current frame, although we can tell the position of the car, we know neither its direction nor velocity that are essential for controlling the car.

For a fair comparison, we followed the same setting as in Ha & Schmidhuber (2018). Specifically, we collected a dataset of $10k$ random rollouts of the environment, each consisting of 1000 time steps, for model estimation. The dimensionality of latent states $\tilde{\vec{s}}_t$ was set to $\tilde{d} = 32$, and regularization parameters was set to $\lambda_1 = 1$, $\lambda_2 = 1$, $\lambda_3 = 1$, $\lambda_4 = 1$, $\lambda_5 = 1$, $\lambda_6 = 6$, $\lambda_7 = 10$, $\lambda_8 = 0.1$, which are determined by hyperparameter turning.

Analysis of ASRs. To demonstrate the structures over observed frames, latent states, actions, and rewards, we visualized the learned $D_{\vec{s}\to o}$, $D_{\vec{s}\to r}$, $D_{\vec{s}}$, and $D_{a\to\vec{s}}$, as shown in Figure 9. Intuitively, we can see that $D_{\vec{s}\to r}$ and $D_{a\to\vec{s}}$ have many values close to zero, meaning that the reward is only influenced by a small number of state dimensions, and not many state dimensions are influenced by the action. Furthermore, from $D_{\vec{s}}$, we found that there are influences from $\tilde{\vec{s}}_{i,t}$ to $\tilde{\vec{s}}_{i,t+1}$ (diagonal values) for most state dimensions, which is reasonable because we want to learn an MDP over the underlying states, while the connections across states (off-diagonal values) are much sparser. Compared to the original 32-dim latent states, ASRs have only 21 dimensions. Below, we empirically showed that the low-dimensional ASRs significantly improve the policy learning performance in terms of both efficiency and efficacy.

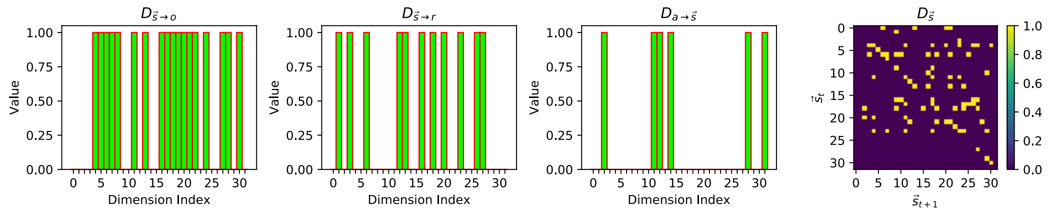

Figure 9: Visualization of estimated structural matrices $D_{\vec{s}\to o}$, $D_{\vec{s}\to r}$, $D_{a\to\vec{s}}$, and $D_{\vec{s}}$ in Car Racing.

Ablation Study. We further performed ablation studies on latent dynamics prediction; that is, we compared with the case when the transition dynamics in (4) are not explicitly involved. Figure 10 shows that by explicitly modelling the transition dynamics (denoted by *with LDP*), the cumulative reward has an obvious improvement over the one without modelling the transition dynamics (denoted by *without LDP*).

Difference between our SS-VAE and Planet, Dreamer. Both our method and Planet (Hafner et al., 2018) and Dreamer (Hafner et al., 2019) are world model-based methods. The differences are mainly in two aspects: (1) our method explicitly considers the structural relationships among variables in the RL system, and (2) it guarantees minimal sufficient state representations for policy learning. Previous approaches usually fail to take into account whether the extracted state representations are sufficient and necessary for downstream policy learning. Moreover, as for the component of recurrent networks, SS-VAE uses LSTM that only contains the stochastic part, while PlaNet and Dreamer use RSSM that contains both deterministic and stochastic components.

G.2 VizDoom Experiment

We also applied the proposed method to VizDoom (Kempka et al., 2016). VizDoom provides many scenarios and we chose the *take cover* scenario (Figure 11). Unlike CarRacing, *take cover* is a discrete control problem with two actions: move left and move right. Reward is +1 at each time step while alive, and the cumulative reward is defined to be the number of time steps the agent manages to stay alive during a episode. Therefore, in order to survive as long as possible, the agent has to learn how to avoid fireballs shot by monsters from the other side of the room. In this task, *solving* is defined as attaining the average survival time of greater than 750 time steps over 100 consecutive episodes, each running for a maximum of 2100 time steps.

Following the same setting as in Ha & Schmidhuber (2018), we collected a dataset of 10k random rollouts of the environment, each consisting of 500 time steps. The dimensionality of latent state $\tilde{\vec{s}}_t$

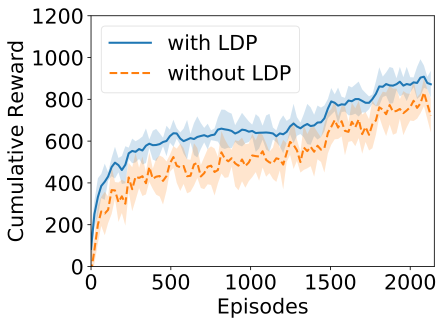

Figure 10: Ablation study of latent dynamics prediction (LDP) evaluated on Car Racing with model-free ASR.

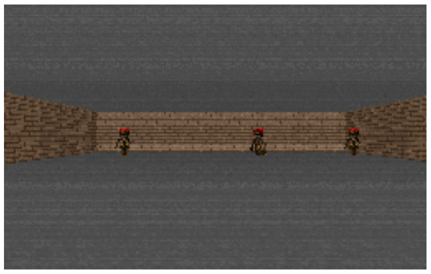

Figure 11: An illustration of VizDoom *take cover* scenario.

is set to $\tilde{d} = 32$. We also set $\lambda_1 = 1$, $\lambda_2 = 1$, $\lambda_3 = 1$, $\lambda_4 = 1$, $\lambda_5 = 1$, $\lambda_6 = 6$, $\lambda_7 = 10$, $\lambda_8 = 0.1$. By tuning thresholds, we finally reported all the results on the 21-dim ASRs, which achieved the best results in all the experiments.

H Detailed Model Architectures

In the car racing experiment, the original screen images were resized to $64 \times 64 \times 3$ pixels. The encoder consists of three components: a preprocessor, an LSTM, and an MDN. The preprocessor architecture is presented in Figure 12, which takes as input the images, actions and rewards, and its output acts as the input to LSTM. We used 256 hidden units in the LSTM and used a five-component Gaussian mixture in the MDN. The decoder also consists of three components: a current observation reconstructor (Figure 13), a next observation predictor (Figure 14), and a reward predictor (Figure 15). The architecture of the transition/dynamics is shown in Figure 16, and its output is also modelled by an MDN with a five-component Gaussian mixture. The architecture of the action prediction is given in Figure 17, which is a two-layer MLP taking states and rewards as input and predicted action as output. In the VizDoom experiment, we used the same image size and the same architectures except that the LSTM has 512 hidden units and the action has one dimension. It is worth emphasising that we applied weight normalization to all the parameters of the architectures above except for the structural matrices $D_{(\cdot)}$.

In DDPG, both actor network and critic network are modelled by two fully connected layers of size 300 with ReLU and batch normalisation. Similarly, in DQN (Mnih et al., 2013) on both ASRs and SSSs, the Q network is also modelled by two fully connected layers of size 300 with ReLU and batch normalisation. However, in DQN on observations, it is modelled by three convolutional layers (i.e., relu conv $32 \times 8 \times 8 \longrightarrow$ relu conv $64 \times 4 \times 4 \longrightarrow$ relu conv $64 \times 3 \times 3$) followed by two additional fully connected layers of size 64. In DRQN (Hausknecht & Stone, 2015) on observations, we used the same architecture as in DQN on observations but padded an extra LSTM layer with 256 hidden units as the final layer.

I Platform and License

We run all the experiments on the servers with 4 NVidia V100 GPUs. We used the Car Racing in OpenAI gym and VizDoom environments and we have cited the creators. In our code, we have used the following libraries: Tensorflow (Apache License 2.0), OpenAI Gym (MIT License), VizDoom (MIT License), OpenCV (Apache 2 License), Numpy (BSD 3-Clause "New" or "Revised" License) and NVIDIA-DALI libraries (Apache 2 License).

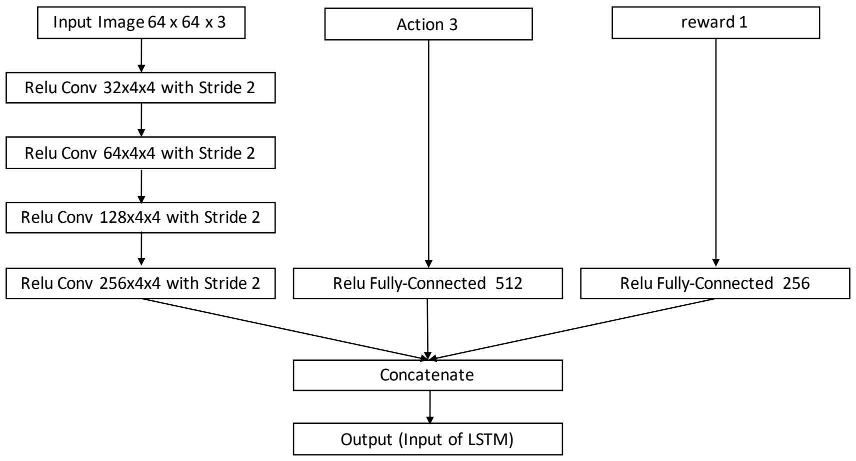
Input Image 64 x 64 x 3
Action 3
reward 1
Relu Conv 32x4x4 with Stride 2
Relu Conv 64x4x4 with Stride 2
Relu Conv 128x4x4 with Stride 2
Relu Conv 256x4x4 with Stride 2
Relu Fully-Connected 512
Relu Fully-Connected 256
Concatenate
Output (Input of LSTM)

Figure 12: Network architecture of preprocessor.

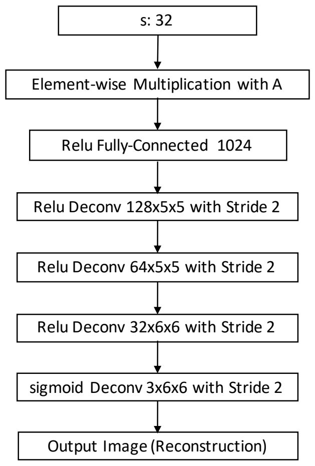
s: 32
Element-wise Multiplication with A
Relu Fully-Connected 1024
Relu Deconv 128x5x5 with Stride 2
Relu Deconv 64x5x5 with Stride 2
Relu Deconv 32x6x6 with Stride 2
sigmoid Deconv 3x6x6 with Stride 2
Output Image (Reconstruction)

Figure 13: Network architecture of observation reconstruction.

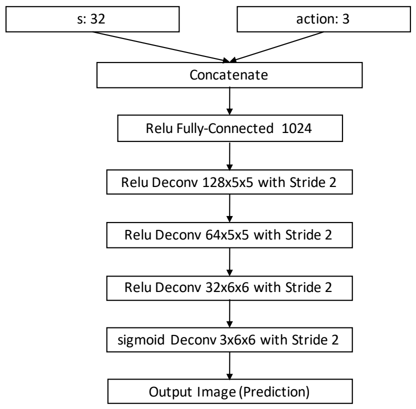
s: 32
action: 3
Concatenate
Relu Fully-Connected 1024
Relu Deconv 128x5x5 with Stride 2
Relu Deconv 64x5x5 with Stride 2
Relu Deconv 32x6x6 with Stride 2
sigmoid Deconv 3x6x6 with Stride 2
Output Image (Prediction)

Figure 14: Network architecture of observation prediction.

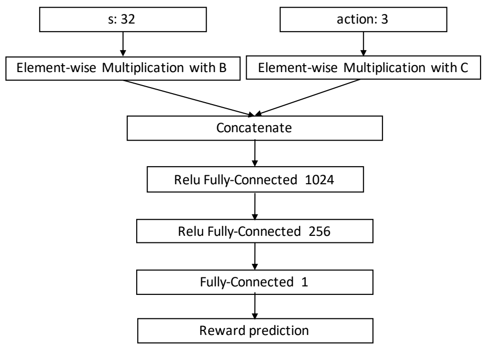
s: 32
action: 3
Element-wise Multiplication with B
Element-wise Multiplication with C
Concatenate
Relu Fully-Connected 1024
Relu Fully-Connected 256
Fully-Connected 1
Reward prediction

Figure 15: Network architecture of reward.

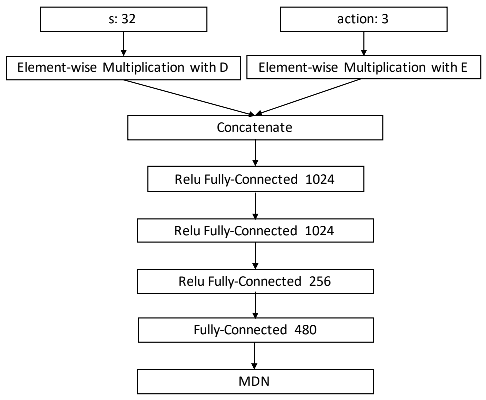
s: 32
action: 3
Element-wise Multiplication with D
Element-wise Multiplication with E
Concatenate
Relu Fully-Connected 1024
Relu Fully-Connected 1024
Relu Fully-Connected 256
Fully-Connected 480
MDN

Figure 16: Network architecture of transition/dynamics.

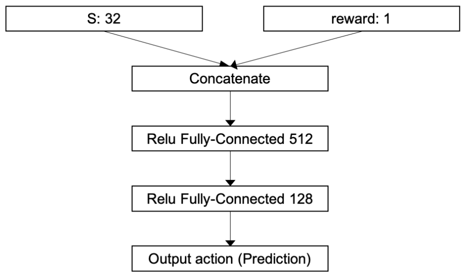

Figure 17: Network architecture of action prediction.

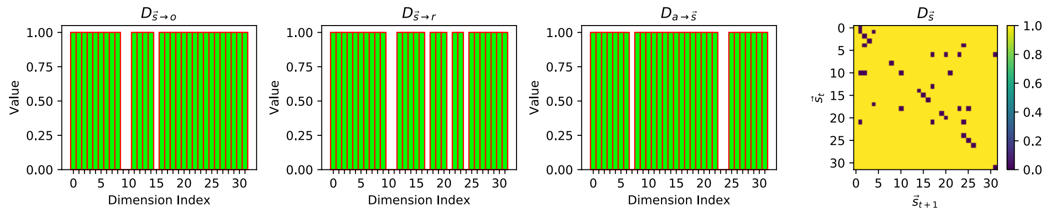

Figure 18: Visualization of estimated structural matrices $D_{\vec{s}\to o}$, $D_{\vec{s}\to r}$, $D_{a\to\vec{s}}$, and $D_{\vec{s}}$ in Car Racing, without the explicit sparsity constraints.

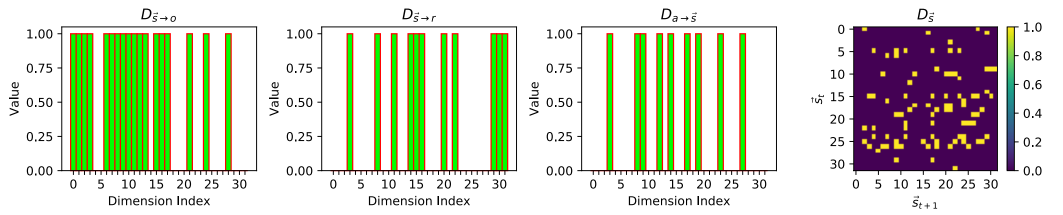

Figure 19: Visualization of estimated structural matrices $D_{\vec{s}\to o}$, $D_{\vec{s}\to r}$, $D_{a\to\vec{s}}$, and $D_{\vec{s}}$ in VizDoom.