APPENDIX

This appendix is organized as follows:

- **Sec A**: the detailed algorithm.
- **Sec B**: experimental settings and additional results on the *Bouncing Ball* domain.
- **Sec C**: experimental settings and additional results on the *Sepsis* domain.

A RL WITH ACTIVE FEATURE ACQUISITION ALGORITHM

Algorithm 1 RL with Active Feature Acquisition

Input: learning rate $\alpha > 0$, dataset $\mathcal{D}$
Initialize RL policy π_f, π_c, VAE parameters θ, ϕ.
Train VAE on dataset $\mathcal{D}$ using Eq (**??**).
while Not Converge **do**
  Reset the environment.
  Initialize null observation $\mathbf{x}_1^p = \emptyset$, feature acquisition action $\mathbf{a}_0^f$ and control action $\mathbf{a}_0^c$.
  for $i = 1$ **to** T **do**
    Compute representation with VAE: $\mathbf{b}_t = q_\phi(\mathbf{x}^p_{\leq t}, \mathbf{a}_{<t})$.
    Sample a feature acquisition action $\mathbf{a}_t^f \sim \pi_f(\mathbf{b}_t)$ and a control action $\mathbf{a}_t^c \sim \pi_c(\mathbf{b}_t)$.
    Step the environment and receive partial features, reward and terminal: $\mathbf{x}^p_{t+1}, r_t, \text{term} \sim \text{env}(\mathbf{a}_t^f, \mathbf{a}_t^c)$
    Compute cost $c_t = \sum_i c \cdot \mathbb{I}(\mathbf{a}_t^{f(i)})$.
    Save the transitions $\{\mathbf{b}_t, \mathbf{a}_t^f, \mathbf{a}_t^c, r_t, c_t, \text{term}\}$.
    if term **then**
      break
    end if
  end for
  Update π_f, π_c using the saved transitions with an RL algorithm under learning rate α.
end while

B BOUNCING BALL$^+$

B.1 TASK SPECIFICATION

The task consists of a ball moving in a 2D box of size 32×32 pixels. The radius of the ball equals to 2 pixels. At each step, a binary image is returned as an observation of the MDP state. At the beginning of every episode, the ball starts at a random position in the *upper left* quadrant (sampled uniformly). The initial velocity of the ball is randomly defined as follows: $\vec{v} = [V_x, V_y] = 4 \cdot \tilde{\vec{v}}/\|\tilde{\vec{v}}\|$, where the x- and y-component of $\tilde{\vec{v}}$ are sampled uniformly from the interval $[-0.5, 0.5]$. There is a navigation target set at $(5, 25)$ pixels, which is in the *lower left* quadrant. The navigation is considered to be successful if the ball reaches the specified target location within a threshold of 1 pixel along both x/y-axis.

The action spaces is defined as follows. There are five task actions $\mathcal{A}^c$:

- Increase velocity leftwards, i.e., change V_x by -0.5
- Increase velocity rightwards, i.e., change V_x by $+0.5$
- Increase velocity downwards, i.e., change V_y by $+0.5$
- Increase velocity upwards, i.e., change V_y by -0.5
- Keep velocities unchanged

The maximum velocity along the x/y-axis is 5.0. The velocity will stay unchanged if it exceeds this threshold. The feature acquisition action $\mathbf{a}^f \in \mathcal{A}^f$ is specified as acquiring the observation of a subset

of the quadrants (this also includes acquiring the observation of all 4 quadrants). Thus, the agent can acquire $0-4$ quadrants to observe. Each episode runs up to 50 steps. The episode terminates if agent reaches the target location.

B.2 Implementation Details

For all the compared methods, *Zero-Imputing* Nazabal et al. (2018) is adopted to fill in missing features with a fixed value of 0.5.

End-to-End The end-to-end model first processes the imputed image by 2 *convolutional* layers with filter sizes of 16 and 32, respectively. Each *convolutional* layer is followed by a *ReLU* activation function. Then the output is passed to a *fully connected* layer of size 1024. The weights for the *fully connected* layer are initialized by *orthogonal weights initialization* and the biases are initialized as zeros.

NonSeq-ZI The non-sequential VAE models first process the imputed image by 2 *convolutional* layers with filter sizes of 32 and 64, respectively. Each *convolutional* layer is followed by a *ReLU* activation function. Then the output passes through a *fully connected* layer of size 256, followed by two additional *fully connected* layers of size 32 to generate the mean and variance of a Gaussian distribution. To decode an image, the sampled code first passes through a *fully connected* layer with size 256, followed by 3 *convolutional* layers with filters of 32, 32, and nc and strides of 2, 2 and 1, respectively, where nc is the *channel* size that equals to 2 for the binary image. There are two variants for *NonSeq-ZI*: one employs the *partial* loss that is only for the observed variables; the other employs the *full* loss that is computed on all the variables, i.e., the ground-truth image with full observation is employed as the target to train the model to impute the missing features. The hyperparameters for training *NonSeq-ZI* are summarized in Table 1.

	Hyperparameter				
	β (KL weight)	KL reduction	Loss reduction	learning rate	epochs
NonSeq-ZI (partial)	1.0	sum	sum	1e-4	1k
NonSeq-ZI (full)	1.0	sum	sum	1e-4	1k
Seq-PO-VAE (ours)	1.0	sum	sum	5e-4	2k

Table 1: Hyperparameter settings for training VAE models on the *Bouncing Ball*$^+$ dataset.

Seq-PO-VAE (ours) At each step, the *Seq-PO-VAE* takes an imputed image and an action vector of size 9 as input. The imputed image is processed by 3 *convolutional* layers with filter size 32 and stride 2. Each *convolutional* layer employs *ReLU* as its activation function. Then the output passes through a *fully connected* layer of size 32 to generate a latent representation for the image $\mathbf{f}_x$. The action vector passes through a *fully connected* layer of 32 to generate latent representation for the action $\mathbf{f}_a$. Then the image and action features are concatenated and augmented to form a feature vector $\mathbf{f}_c = [\mathbf{f}_x, \mathbf{f}_a, \mathbf{f}_x * \mathbf{f}_a]$, where $[\cdot]$ denotes *concatenation* of features. Then $\mathbf{f}_c$ is fed to *fully connected* projection layers of size 64 and 32, respectively. The output is then fed to an *LSTM* module, with latent size of 32. The output $\mathbf{h}_t$ of *LSTM* is passed to two independent *fully connected* layers of size 32 for each to generate the mean and variance for the Gaussian distribution filtered from the sequential inputs. To decode an image, the model adopts *deconvolutional* layers that are identical to those for *NonSeq-ZI*. The hyperparameters for training *Seq-PO-VAE* are shown in Table 1.

LSTM-A3C We adopt LSTM-A3C Mnih et al. (2016) to train the RL policy. The policy takes the features derived from the representation learning module as input. For the VAE-based methods, the input features are passed through a *fully connected* layer of size 1024. Then the features are fed to an *LSTM* with 1024 units. The output of the *LSTM* is fed to three independent *fully connected* layers to generate the estimations for value, task policy and feature acquisition policy. We adopt *normalized column* initialization for all the *fully connected* layers and the biases for the *LSTM* module are set to be zero.

B.3 DATA COLLECTION

To train the VAEs, we prepare a training set that consists of 2000 trajectories. Half of the trajectories are derived from a random policy and the other half is derived from a policy learned from end-to-end method. To train the end-to-end method, we employ a cost of 0.01 over first 2m steps and then increase it to 0.02 for the following 0.5m steps. All the VAE models are evaluated on a test dataset that has identical size and data distribution as the training dataset. We present the best achieved task performance of the data collection policy (*End-to-End*) and our representation learning approach in Table 4. We notice that our proposed method, by employing an advanced representation model, leads to significantly better feature acquisition policy than *End-to-End* (smaller number of observations while achieving similar or better reward).

	Model	
	End-to-End	Ours
Average # of observations per episode	17.94	**8.24**
Task reward	1.0	1.0

Table 2: Task performance for the data collection policy and our method on *Bouncing Ball*$^+$.

B.4 IMPUTING MISSING FEATURES VIA LEARNING MODEL DYNAMICS

We present an illustrative example to demonstrate the process of imputing missing features and the role of learning model dynamics. To this end, we collect trajectories under an *End-to-End* policy (the choice of the underlying RL policy is not that important since we just want to derive some trajectory samples for the VAE models to reconstruct) and use different VAE models to impute the observations.

From the results presented in Figure 2, we observe that under the partially observable setting with missing features, the latent representation derived from our proposed method provides abundant information as compared to only using information from a single time step and thereby offers significant benefit for the policy model to learn to acquire meaningful features/gain task reward.

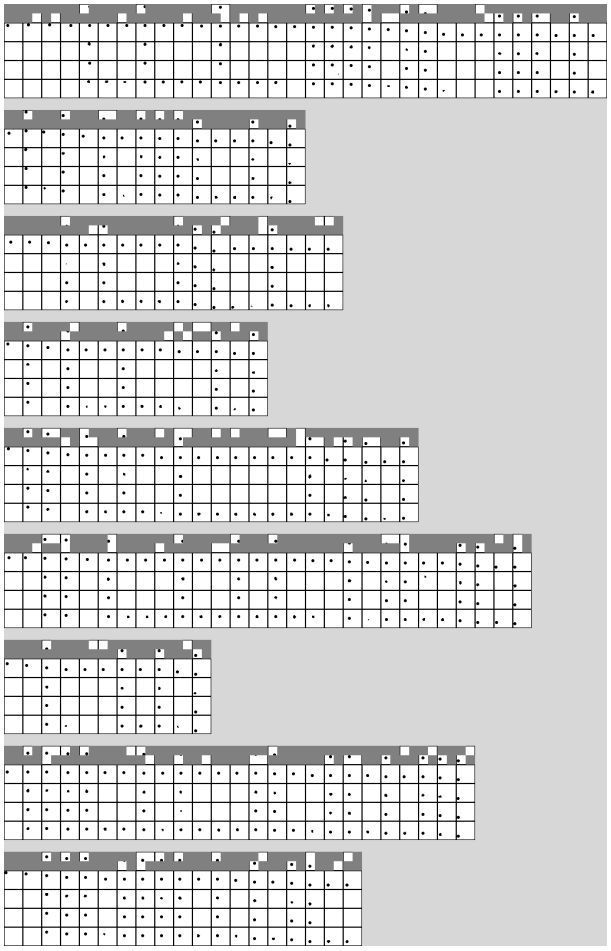
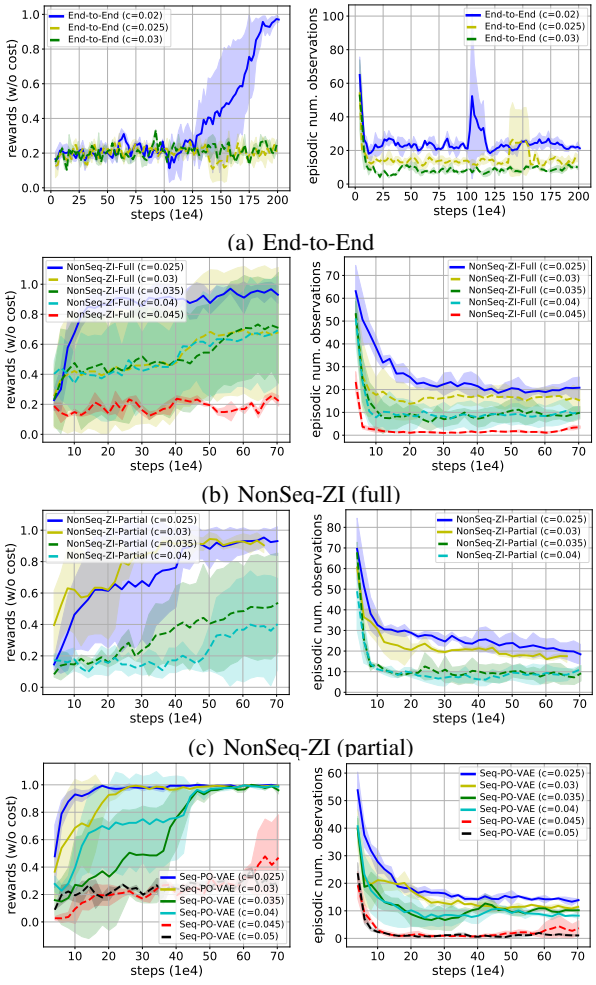


Figure 1: Imputation results for different VAE models. We select 9 trajectories obtained from the trained *End-to-End* policy. Each block corresponds to the results for one trajectory (better to view enlarged). The five rows in one block are (top-down): (1) partial observations acquired by the agent; (2) ground-truth image with full observation; (3) Imputation by *NoSeq-ZI (partial)*; (4) Imputation by *NoSeq-ZI (full)*; (5) Imputation by *Seq-PO-VAE (ours)*. Our model can often successfully predict the balls location even if it is not present in the acquired observation. Hence it successfully employs its learned knowledge of the dynamics. In contrast, the non-sequential model (obviously) fails to predict the balls location when the ball is not present in the observation.

B.5 INVESTIGATION ON COST-PERFORMANCE TRADE-OFF

We perform a case study on investigating the cost-performance trade-off for each representation learning method, presented in Figure 2. Apparently, as we increase the cost, the exploration-exploitation task becomes more challenging and each compared method has its own upper bound on the cost above which it fails to learn an effective task policy while acquiring minimum observation. First, we notice that the *End-to-End* model takes a long time to progress in learning task skills, while the VAE-based models can progress much faster. Among the VAE-based methods, we notice that our proposed method (Figure 2(d)) can achieve as low as 8 observations whereas the baselines *NonSeq-ZI (Full)* (Figure 2(b)) and *NonSeq-ZI (partial)* (Figure 2(c)) achieve a standard of $\sim$20 (lowest point among the *solid* lines). Thus, we could conclude that our proposed approach can significantly benefit the cost-sensitive policy training and lead to a policy which acquires much fewer observations while still succeeding in terms of task performance.



(a) End-to-End

(b) NonSeq-ZI (full)

(c) NonSeq-ZI (partial)

(d) Seq-PO-VAE (ours)

Figure 2: Cost-performance trade-off investigation. Each row corresponds to the performance in terms of task reward and episodic number of acquisitions obtained for a specific method (see legends). Each curve is derived from 3 independent runs. We use dotted lines to indicate those instances when the task learning does not always succeed. Thus, the best achievable number of observations should be referred to as the lowest curve among the *solid* lines.

C SEPSIS MEDICAL SIMULATOR

C.1 TASK SPECIFICATIONS

For this task we employ a Sepsis simulator proposed in previous work Oberst & Sontag (2019). The task is to learn to apply three *treatment* actions for Sepsis patients in intensive care units, i.e., $\mathcal{A}^c = \{$*antibiotic*, *ventilation*, *vasopressors*$\}$. At each time step, the agent selects a subset of the *treatment* actions to apply. The state space consists of 8 features: 3 of them specify the current *treatment* status; 4 of them specify the *measurement* status in terms of *heart rate*, *sysBP rate*, *percoxyg stage* and *glucose level*; the remaining one is a categorical feature indicating the patent's antibiotic status. The feature acquisition actively selects a subset among the *measurement* features for observation, i.e., $\mathcal{A}^f = \{$*heart rate*, *sysBP rate*, *percoxyg state*, *glucose level*$\}$. The objective for learning a active feature acquisition strategy is to help the decision making system to reduce *measurement* cost at a significant scale.

C.2 IMPLEMENTATION DETAILS

For all the compared methods, we adopt *Zero-Imputing* Nazabal et al. (2018) to fill in missing features. In particular, a fixed value of -10 which is outside the range of feature values is used to impute missing values.

End-to-End The end-to-end model first processes the imputed state by 3 *fully connected* layers of size 32, 64 and 32, respectively. Each *fully connected* layer is followed by a *ReLU* activation.

NonSeq-ZI The VAE model first processes the imputed state by 2 *fully connected* layers with size 32 and 64, with the first *fully connected* layer being followed by *ReLU* activation functions. Then the output is fed into two independent *fully connected* layers of size 10 for each, to generate the mean and variance for the Gaussian distribution. To decode the state, the latent code is first processed by a *fully connected* layer of size 64, then fed into three *fully connected* layers of size 64, 32, and 8. The intermediate *fully connected* layers employ *ReLU* activation functions. Also, we adopt two variants for *NonSeq-ZI*, trained under either *full* loss or *partial* loss. The details of the hyperparameter settings used for training are presented in Table 3.

	Hyperparameter				
	β (KL weight)	KL reduction	Loss reduction	learning rate	epochs
NonSeq-ZI (partial)	0.01	sum	sum	1e-4	1k
NonSeq-ZI (full)	0.01	sum	sum	1e-4	1k
Seq-PO-VAE (ours)	0.01	sum	sum	1e-3	1k

Table 3: Hyperparameter settings for training VAE models on the *Sepsis* dataset.

Seq-PO-VAE (ours) At each time step, the inputs for state and action are first processed by their corresponding projection layers. The projection layers for the state consists of 3 *fully connected* layers of size 32, 16 and 10, where the intermediate *fully connected* layers are followed by a *ReLU* activation function. The projection layer for the action input is a *fully connected* layer of size 10. Then the projected state feature $\mathbf{f}_c$ and action feature $\mathbf{f}_a$ are combined in the following manner: $\mathbf{f}_c = [\mathbf{f}_x, \mathbf{f}_a, \mathbf{f}_x * \mathbf{f}_a]$. $\mathbf{f}_c$ is passed to 2 *fully connected* layers of size 64 and 32 to form the input to the *LSTM* module. The output $\mathbf{h}_t$ of the *LSTM* is fed to two independent *fully connected* layers of size 10 to generate the mean and variance for the Gaussian distribution. The decoder for *Seq-PO-VAE* has identical architecture as *NonSeq-ZI*. The details for training *Seq-PO-VAE* are presented in Table 3.

LSTM-A3C The LSTM-A3C Mnih et al. (2016) takes encoded state features derived from the corresponding representation model as its input. The encoded featuresare fed into an *LSTM* with size 256. Then the $\mathbf{h}_t$ for the *LSTM* is fed to three independent *fully connected* layers, to predict the state value, feature acquisition policy and task policy. *Normalized column* initialization is applied to all *fully connected* layers. The biases for the *LSTM* and *fully connected* layers are initialized as zero.

C.3 Data Collection

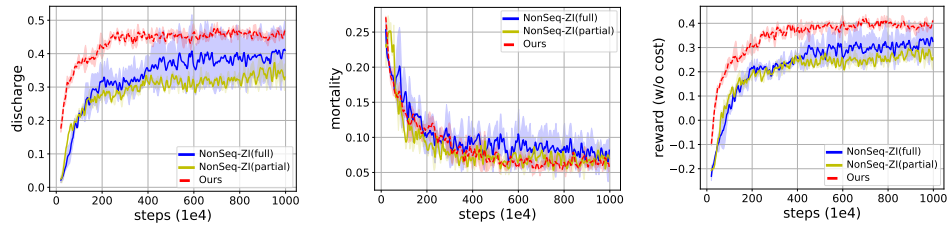
To train the VAEs, we prepare a training set that consists of 2000 trajectories. Half of the trajectories are derived from a random policy and the other half is derived from a policy learned from the *End-to-End* method with cost 0.0. All the VAE models are evaluated on a test dataset that consists of identical size and data distribution as the the training dataset. We present the task treatment reward obtained by our data collection policy derived from the *End-to-End* method and that obtained by our proposed method in Table 4. Noticeably, by performing representation learning, we obtained much better treatment reward as compared to the data collection policy, which demonstrates the necessity of performing representation learning.

	Model	
	End-to-End	Ours
Treatment Reward	0.35	**0.45**

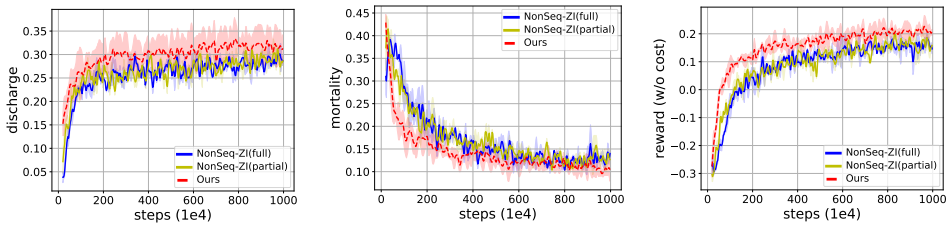
Table 4: Task performance for the data collection policy and our proposed method on *Sepsis*.

C.4 More Comparison Result under Different Values for Cost

We present additional experiment results that compare our proposed method and the non-sequential baselines under the cost values $\{0, 0.025\}$. The results for cost value of 0.01 are shown in the main paper. Overall, under all the cost settings, our method leads to significantly better discharge ratio and task reward compared to the baselines.



(a) Cost=0



(b) Cost=0.025

Figure 3: Comparison result between our proposed method and the non-sequential VAE baseline models under different values for cost. Each curve is derived from 3 independent runs.

Also, we demonstrate the cost-performance trade-off on *Sepsis* domain. By increasing the value of cost, we could obtain feature acquisition policy that acquires substantially decreased amount of features within each episode.

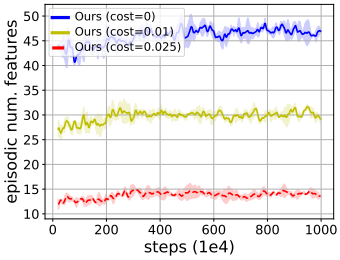


Figure 4: Average number of observations acquired in each episode when training our proposed model under different cost values.

C.5 Illustrative Examples for Missing Feature Imputation in *Sepsis*

We present two illustrative examples in Figure 5 to demonstrate how imputing missing features via learning model dynamics would help the decision making with partial observability in *Sepsis* domain. The policy training process with partial observability could only access very limited information, due to the employment of active feature acquisition. Under such circumstances, imputing the missing features would offer much more abundant information to the decision making process. From the results shown in Figure 5, our model demonstrates considerable accuracy in imputing the missing features, even though it is extremely challenging to perform the missing feature imputation task given the distribution shift from the data collection policy and the online policy. The imputed missing information would be greatly beneficial for training the task policy and feature acquisition policy.

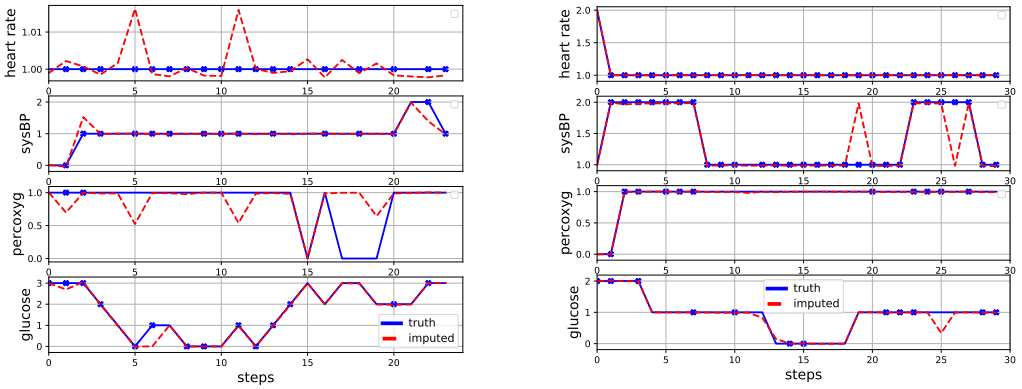


Figure 5: Two example trajectories for illustrating how our method works on the *Sepsis* medical domain. The acquisition policy is trained with a cost of 0. Each block corresponds to one trajectory and the four rows correspond to the four *measurement* features being considered for active feature acquisition. Each dot indicates the employment of feature acquisition on the corresponding *measurement* feature at the presented time point. In each trajectory, we demonstrate the ground-truth signal over time as well as the imputed signal over time predicted by our proposed *Seq-PO-VAE* model. By imputing the missing features via learning model dynamics, our proposed method could offer much more informative representation for the policy training compared to the non-sequential VAE baselines, and thus significantly benefit the policy training with partial observability.

REFERENCES

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937, 2016.

Alfredo Nazabal, Pablo M Olmos, Zoubin Ghahramani, and Isabel Valera. Handling incomplete heterogeneous data using vaes. *arXiv preprint arXiv:1807.03653*, 2018.

Michael Oberst and David Sontag. Counterfactual off-policy evaluation with gumbel-max structural causal models. In *ICML*, 2019.