Improving Deep Reinforcement Learning with Transitional Variational Autoencoders: A Healthcare Application

Preprint · May 2020

DOI: 10.13140/RG.2.2.21167.28324

3 authors, including:

Matthew Baucum
University of Tennessee
8 PUBLICATIONS **2** CITATIONS

SEE PROFILE

Anahita Khojandi
University of Tennessee
44 PUBLICATIONS **100** CITATIONS

SEE PROFILE

Improving Deep Reinforcement Learning with Transitional Variational Autoencoders: A Healthcare Application

M. Baucum, A. Khojandi, *Member, IEEE*, and R. Vasudevan

***Abstract*— Reinforcement learning is a powerful tool for developing personalized treatment regiments from healthcare data. Yet training reinforcement learning agents typically requires an environment that the agent can query to learn the effects of various treatment actions. Since the underlying dynamics of many health conditions are unknown, researchers often must estimate models of such 'patient environments' from finite datasets, oftentimes using Hidden Markov Models. We instead propose using variational autoencoders, which use neural networks to learn a direct mapping between distributions over patient measurements at consecutive time points. We test this approach on a dataset of ICU patients and find that variational autoencoders produced more realistic patient trajectories and better-performing treatment policies compared to hidden Markov models. We find that hidden Markov models' reliance on discrete latent states resulted in volatile patient trajectories, diminishing their effectiveness at training reinforcement learning agents.**

***Index Terms*— reinforcement learning, hidden Markov models, variational autoencoders**

I. Introduction

The reinforcement learning (RL) paradigm offers a promising way to optimize treatment regimes for acute and chronic conditions. In recent years, researchers have applied RL and deep RL (DRL) techniques to optimize treatment strategies in a large array of diseases, e.g., for HIV medication regimens ([1]), neurostimulation therapy for epilepsy ([2]), sepsis treatment ([3]), drug therapy for myeloma patients ([4]), and anticoagulant medication administration ([5]).

Reinforcement learning seeks to learn a policy that maps an environmental state (e.g., a patient's health status or symptom severity) to an optimal action, with the goal of maximizing long-term expected reward. Yet reinforcement learning in healthcare is complicated by the fact that treatment policies cannot be learned from actual patients in real time, for obvious ethical reasons. One way to address this limitation is to train reinforcement learning policies directly from existing datasets (e.g., [3]). This approach, known as off-policy reinforcement learning, allows an agent to map patient states to optimal treatment actions using data generated from a policy other than its own (e.g., from the patient's original physician).

Yet many state-of-the-art reinforcement learning algorithms require on-policy learning (where an agent can receive feedback on its own policy), which is only possible when an agent has access to a fully responsive environment that provides feedback on any state/action pair. This poses a 'counterfactual problem' in healthcare datasets, where it is not known how a patient would have responded to a treatment or dosage other than the one they actually received.

In this work, we propose *generative neural networks* as a solution for such patient-modeling tasks. Specifically, we propose a modification of the variational autoencoder (a type of generative neural network) and test its ability to simulate realistic patient trajectories for reinforcement learning. This data-driven modeling approach can improve RL healthcare policies by allowing state-of-the-art, on-policy algorithms to be trained from preexisting datasets, while making no restrictive assumptions about the underlying disease dynamics. We compare its performance to a commonly-used classical method for patient modeling and draw insights on the effectiveness of each modeling approach.

A. Generative Neural Networks

Unlike standard neural networks, which optimize predictive accuracy, generative neural networks attempt to learn underlying representations for their training data that can be used to generate novel, synthetic data points from an underlying 'latent' distribution. Generative neural networks have already shown promise in generating synthetic patient profiles based on real data [6, 7, 8]. They have also been used to augment RL algoritmhs; [9] shows that generative neural networks can be used to approximate POMDPs in game-like tasks and subsequently used to train RL agents. Here, we build upon this line of work and examine the role of generative neural networks in healthcare, where the underlying structure of a disease's progression is often unknown.

In this work, we specifically focus on variational autoencoders (VAEs) [10], which learn a multivariate latent Gaussian distribution that can produce realistic synthetic data points when sampled from. VAEs impose no distributional assumptions, and incorporate stochasticity that distinguishes them

*The research is partially supported by Science Alliance, The University of Tennessee, and the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory, managed by UT-Battelle, LLC, for the U.S. Department of Energy

M. Baucum and A. Khojandi are with the Department of Industrial & Systems Engineering, University of Tennessee, Knoxville, TN 37996 (corresponding author: khojandi@utk.edu).

R. Vasudevan is with the Center for Nanophase Materials Sciences, Oak Ridge National Laboratory, Oak Ridge, TN 37830

from standard, deterministic neural networks. We propose a variant of the VAE, the 'transitional variational autoencoder' (tVAE), that adapts the VAE structure to the type of longitudinal patient data that is frequently encountered in the healthcare domain. The tVAE builds a data-driven patient model that, while able to stand on its own for synthetic patient generation, can also pair with an RL algorithm for learning treatment policy.

B. Other Modeling Approaches

Aside from generative neural networks, there are other approaches for fitting parameterized models to patient datasets in a way that allows for on-policy reinforcement learning. Researchers in [1] relied partly on kernel-based models, which are well-suited to clustered datasets, while [2] used Markov models, which are well-suited to datasets with discrete or low-dimensional state spaces. One of the most common modeling approaches for patient data, and the focus of the present study, is the hidden Markov model (HMM) [11], which has been used for learning optimal treatment policies [4, 5] and modeling disease progression for HIV [12], glaucoma [13], and breast cancer [14]. HMMs generally model patient measurements as manifestations of a discrete set of latent disease states, and estimate the transitions between such states as well as the measurement distributions associated with each state. The HMM's intuitive structure and relatively interpretable parameters make them an appealing option for fitting generative models to patient data.

C. Objective and Contributions

The purpose of this work is to examine the contribution of the proposed generative neural network, tVAE, in the context of DRL, and benchmark it against the best-known and most-used generative model for patient data, HMM. We argue that the HMM's assumptions of normally-distributed patient data and a *discrete* latent space may limit its effectiveness for creating a realistic RL environment. Our proposed generative neural network approach thus contributes to the healthcare RL literature by 1) placing no distributional assumptions on the data, and 2) allowing for a more flexible mapping between subsequent patient measurements compared to existing methods, thereby allowing on-policy RL algorithms to be used on a wider array of patient data structures.

We showcase the contributions of our proposed method in the context of optimal medication administration planing (dosage and timing) in the intensive care unit (ICU). Specifically, we use an existing dataset of anticoagulant medication administration records (timing/dosage) and the corresponding outcomes in a cohort of ICU patients to learn and prescribe optimal medication administration policies. We use both the HMM and our proposed tVAE alternative to generate 'environments' for training on-policy RL agents. We generate synthetic patient trajectories with each environment and assess their resemblance to actual patient data, as well as the correspondence between actual patient outcomes and model-forecasted outcomes. We also assess each environment's effectiveness at training optimal patient treatment policies through on-policy RL, by 'cross-testing' RL agents trained in each environment. While neither the HMM or tVAE will perfectly represent the 'true' patient environment from which the data were drawn, we propose that a desirable RL environment should be able to train agents that are robust to alternative environment models/specifications. Thus, we expect the best-performing model (i.e., the model that produces more realistic patient trajectories) to be capable of training an RL agent that, when tested on the *other* model's environment, still performs well compared to that model's agent.

It is worth noting that, while we attempt to demonstrate the tVAE's usefulness in creating DRL environments from patient data, we do not claim this to be the *best* approach for doing so. Other generative methods, such as generative adversarial networks (GANs), may also outperform HMMs in constructing realistic patient environments. We leave it to future work to directly compare the effectiveness of various generative deep learning methods for healthcare DRL.

II. Methods

In this section, we first discuss our dataset, and introduce our proposed tVAE model and the benchmark HMM approach. We then present our method for using DRL to learn optimal dosage policies from each model's environment. Fig. 1 outlines our overall modeling and analytical approach.

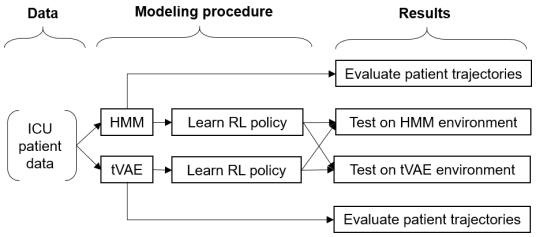

Fig. 1: Methodology overview. HMM and tVAE models were learned from ICU patient data, then evaluated based on their simulated trajectories and ability to train RL agents.

A. Dataset

Data consisted of intensive care unit data from 2,067 patients in the publicly available MIMIC dataset [15, 16]. This dataset is a subset of the dataset used in [5], which also used hidden Markov models to learn optimal medication dosing through reinforcement learning.

Each patient in the dataset received intravenous administration of heparin, an anticoagulant sometimes used to prevent blood clots, treat strokes, or prepare patients for surgery. Heparin's effectiveness is generally measured by activated partial thromboplastin time (aPTT), which is a measure of blood coagulation, defined as the number of seconds required for blood to clot when exposed to specific activator chemicals.

The dataset consists of hourly measurements of each patient's heparin dosage, aPTT values, and 13 other clinical variables - arterial carbon dioxide, heart rate, creatinine, the Glasgow Coma Score, hematocrit, hemoglobin, international normalized ratio of prothrombin, platelet count, prothrombin

time, arterial oxygen saturation, temperature, urea, and white blood cell count.

These measurements are included in our study because of their general relevance to patient health or their specific relevance to blood anticoagulation [5].

Because heparin dosages are weight-based (expressed as mL/kg of patient body weight [17]), each patient's hourly heparin dosage was divided by their weight and discretized into six categories [5]. Heparin dosages of zero (no administration) constituted one dosage category, with the remaining dosages split along the 20th, 40th, 60th, and 80th percentiles into five additional categories, resulting in six heparin dosage groups $d_t \in \{0, 1, 2, 3, 4, 5\}$ for all time t.

As in [5], missing heparin dosages were imputed according to sample-and-hold interpolation. Missing aPTT measurements were imputed according to a neural network, which predicted aPTT values from the remaining 13 clinical values, heparin dosage, as well as the patients' gender, age, and weight. The final dataset consisted of 54,906 measurements, with an average of 26.6 sequential hourly measurements for each of the 2,067 patients. Each of the 14 state variables (i.e., aPTT values and the 13 other clinical variables) was standardized prior to modeling.

B. Transitional Variational Autoencoders (tVAE)

1) tVAE Overview: The proposed transitional variational autoencoder (tVAE) adapts the structure of standard VAEs to model transitions between consecutive patient measurements. A standard VAE seeks to maximize the probability of a dataset, $P(X)$, by assuming that entries in X are distributed according to some function of a latent normal variable Z, which is assigned a prior distribution $P(Z) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ [10]. That is, $X \sim g(Z)$ such that maximizing $P(X)$ is equivalent to maximizing $\mathbb{E}_{Z\sim P(Z)} P(X|Z)$. Since large regions of $P(Z)$ may yield near-zero values of $P(X|Z)$, VAEs assume an approximator distribution $Q(Z|X)$ that conditions Z on X and thus identifies values of Z that are likely to have produced X. The model learns a function h that maps X to $Q(Z|X)$, i.e., $h : X_t \mapsto Q(Z_t|X_t)$ at time t, typically assuming $Q(Z|X) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu} = h_\mu(X)$ and $\boldsymbol{\Sigma}$ is a diagonal matrix with entries $\boldsymbol{\sigma^2} = h_\sigma(X)$. The model also learns a function g that maps $Q(Z|X)$ to $P(X|Z)$, i.e., $g : Q(Z_t|X_t) \mapsto P(X_t|Z_t)$ for time t. In other words, g learns to 're-map' a distribution of latent values to a distribution over observations.

If a suitable approximator distribution is found, then $\mathbb{E}_{Z\sim Q(Z|X)} P(X|Z)$ can be substituted for $\mathbb{E}_{Z\sim P(Z)} P(X|Z)$. This substitution yields the following relation between $P(X)$ (the quantity to be maximized) and $\mathbb{E}_{Z\sim Q(Z|X)} P(X|Z)$ [18]:

$$\log P(X) = D_{KL}[Q(Z|X)||P(Z|X)] + \mathbb{E}_{Z\sim Q(Z|X)}[\log P(X|Z)] - D_{KL}[Q(Z|X)||P(Z)] \quad (1)$$

where D_{KL} is the Kullback-Leibler divergence. The above equation forms the basis for the VAE objective function, and follows directly from the expression for the Kullback-Leibler divergence between $Q(Z|X)$ and $P(Z|X)$ (the true, unknown distribution for Z given X):

$$\begin{aligned} &D_{KL}[Q(Z|X)||P(Z|X)] \\ &= \mathbb{E}_{Z\sim Q(Z|X)}[\log Q(Z|X) - \log P(Z|X)] \\ &= \mathbb{E}_{Z\sim Q(Z|X)}[\log Q(Z|X) - \log(P(Z) \cdot P(X|Z)/P(X))] \\ &= \mathbb{E}_{Z\sim Q(Z|X)}[\log Q(Z|X) - \log P(Z) - \log P(X|Z)] \\ &+ \log P(X) \\ &= D_{KL}[Q(Z|X)||P(Z)] - \mathbb{E}_{Z\sim Q(Z|X)}[\log P(X|Z)] \\ &+ \log P(X) \end{aligned} \quad (2)$$

Note that the last line in equation (2) can easily be rewritten as the expression in equation (1).

Thus, based on equation (1), the VAE aims to maximize $D_{KL}[Q(Z|X)||P(Z|X)]$ and $\mathbb{E}_{Z\sim Q(Z|X)}[\log P(X|Z)]$, and minimize $D_{KL}[Q(Z|X)||P(Z)]$. The term $\mathbb{E}_{Z\sim Q(Z|X)}[\log P(X|Z)]$ represents the likelihood of the data given Z, *according to the conditional distribution* $Q(Z|X) = \mathcal{N}(h_\mu(X), \text{diag}(h_\sigma(X)))$. The VAE thus learns functions h_μ and h_σ that produce conditional distributions over Z, which, in turn, yield high values of $P(X|Z)$. Minimizing $D_{KL}[Q(Z|X)||P(Z)]$ ensures that the learned distribution for $Q(Z|X)$ does not stray too far from the prior distribution $P(Z) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Note that the term $D_{KL}[Q(Z|X)||P(Z|X)]$ cannot be computed, since $P(Z|X)$ is unknown; however, since this term will always be nonnegative, maximizing the sum of the other two terms maximizes the lower bound for $\log P(X)$. VAEs are thus designed to maximize this "evidence lower bound" (ELBO) as a proxy for maximizing $P(X)$.

More intuitively, the VAE learns a neural network h that maps individual data points to mean and variance vectors for the distribution $Q(Z|X)$, in such a way that samples from $Q(Z|X) \sim \mathcal{N}(h_\mu(X), \text{diag}(h_\sigma(X)))$ can be passed to a decoder network g capable of reconstructing the original input. Maximizing the loss term $\mathbb{E}_{Z\sim Q(Z|X)}[\log P(X|Z)]$ is equivalent to minimizing the reconstruction loss between the data X and the output of the decoder network. Minimizing the term $D_{KL}[Q(Z|X)||P(Z)]$ regularizes the encoder network h to produce latent distributions that are relatively close to $\mathcal{N}(\mathbf{0}, \mathbf{I})$; this ensures that $Q(Z|X)$ is relatively compact and can be directly sampled from without producing erroneous decodings from g.

We modify the standard VAE with the assumption that, for sequential data of length T, data at each time point arise from identically distributed latent variables $\{Z_0, ..., Z_T\}$, and that changes between consecutive time points arise from changes in Z, i.e., for two consecutive time points t and $t+1$, $X_t \sim g(Z_t)$ and $X_{t+1} \sim g(Z_{t+1})$. While the actual function governing $Z_t \mapsto Z_{t+1}$ is unknown, we assume that it is Markovian (dependent on the current state only), and hypothesize that for any t, Z_t can be used to predict Z_{t+1}.

For any t, a standard VAE would identify regions of Z likely to have produced X_t using $Q(Z_t|X_t)$; under the assumption outlined above, we instead use $Q(Z_t|X_{t-1})$. Intuitively, we assume that we can learn some function h' which will map an observation at $t-1$ to regions of the latent space that were likely to have generated the *next* observation at t, i.e.,

$h' : X_{t-1} \mapsto Q(Z_t|X_{t-1})$. The decoder function g' then maps the latent distribution $Q(Z_t|X_{t-1})$ to a distribution over observations at time t, i.e., $g' : Q(Z_t|X_{t-1}) \mapsto P(X_t|Z_t)$. Note that we can easily condition the latent distribution on an action a_{t-1} taken at time $t-1$, by simply including a_{t-1} in the input to h', resulting in the latent distribution $Q(Z_t|X_{t-1}, a_{t-1})$. Thus, whereas a standard VAE encodes an input X_t into the latent distribution $Q(Z_t|X_t)$ before reconstructing X_t, our tVAE encodes a data point X_{t-1} into a latent distribution $Q(Z_t|X_{t-1})$ (or $Q(Z_t|X_{t-1}, a_{t-1})$), then attempts to construct the observation X_t. Fig. 2 outlines the structure of standard VAEs versus the tVAE.

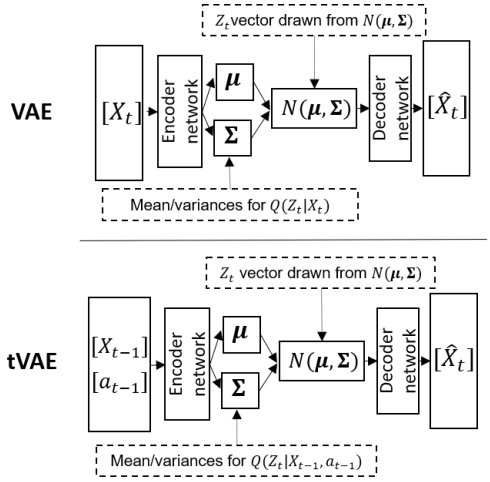

Fig. 2: Structure of the variational autoencoder (VAE, top) and our transitional variational autoencoder (tVAE, bottom).

Note that there are other methods of building generative neural networks that allows for dependence between sequential observations. Conditional Variational Autoencoders (CVAEs, [18]) allow for dependence between generated data and input labels, and have been used in object trajectory prediction tasks [19] and handwritten digit generation [20]. CVAEs are typically used when the data to be reconstructed is of higher dimensionality than the input features, whereas in sequential datasets the input dimensionality (action taken plus current patient state) often exceeds output dimensionality (next patient state). Thus, CVAEs performed relatively poorly in preliminary tests despite having more parameters; thus, we proceed with the tVAE architecture outlined above, which lends itself better to sequential decision-making problems where input dimensionality exceeds output dimensionality.

2) tVAE-Based Environment: For a given time t, the model took as input the 14-length state vector for time t appended to a one-hot encoding of the action take at that time point (e.g., $[1, 0, 0, 0, 0, 0]$ for $d_t = 0$). Each input vector passed through a single hidden layer with u units with sigmoid activations, then passed to a layer defining l means ($\boldsymbol{\mu}$) and l variances ($\boldsymbol{\sigma^2}$) for the latent distribution Z. An l-length vector was drawn from $\mathcal{N}(\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma^2}))$ and passed through a u-unit decoder layer with sigmoid activations, before reaching the output layer predicting the patient state at $t+1$. The final model had $u = 10$ hidden units (roughly 2/3 of the output feature space) and $l = 7$ latent dimensions (roughly 1/2 of output feature space), based on guidelines in [21] and the model architecture used in [8].

The model was trained using gradient descent to minimize a weighted sum of the prediction loss (measured in mean squared error) and the KL divergence between $\mathcal{N}(\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma^2}))$ and $\mathcal{N}(\mathbf{0}, \mathbf{I})$. While the theoretical derivation of the VAE loss formula specifies equal weighting for these terms, in practice, this can cause the model to undertrain toward prediction loss and over-regularize Z, such that $\mathcal{N}(\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma^2}))$ is very close to the prior $\mathcal{N}(\mathbf{0}, \mathbf{I})$ [22, 23]. Preliminary testing suggested that weighting prediction loss at 80% and regularization at 20% improved training stability and model performance.

C. Benchmark Method: Hidden Markov Models (HMM)

1) HMM Overview: An HMM assumes that a data sequence X_t for $t \in \{0, ..., T\}$ is distributed according to an unobserved Markov process between m latent states, defined by the tuple $< \pi, P, \Psi >$. Here, π is a discrete probability distribution over the m hidden states at time $t = 0$, P is an $m \times m$ transition probability matrix (TPM) that specifies the transition probabilities among the m states between consecutive time points, and Ψ is an emission model that describes the distribution of observations from each hidden state. In the case of continuous observations, Ψ is a collection of m unique k-dimensional normal distributions $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma_i})$, where k is the dimensionality of the observation vectors and i denotes the index of the latent state. Thus, when the system occupies hidden state i, the observation vector at that time point is assumed to be distributed according to $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma_i})$. We focus on time-homogeneous HMMs in which P does not vary based on the time t, and we assume all $\boldsymbol{\Sigma_i}$ are diagonal matrices (i.e., all covariance between the k observation variables is accounted for by the latent state). Note that in our models, $k = 14$, corresponding to the total number of state variables.

HMMs are fit with the Baum Welch algorithm [11], an expectation maximization algorithm that identifies parameters for $< \pi, T, \Psi >$ that maximize $P(X)$, the probability of observing the dataset. Multiple values of m can be tested, with the final value being determined by some preferred measure of model fit (e.g., [4]). Note that HMM's can also be stratified to produce different parameter estimates for different subgroups (e.g., [14]). If we assume that the transition probabilities depend on some action taken at time t, $a_t \in A$, we can train the HMM separately for sequences in which different actions were taken, estimating separate transition probability matrices P_a for $a \in A$.

2) HMM-Based Environment: To compare tVAE performance to methods currently used in healthcare RL, we also estimated an RL environment based on an HMM. This was performed in two stages. First, we used the Baum Welch algorithm to learn the $k \times 1$ mean vector and $k \times k$ covariance matrix (where $k = 14$) for each of the model's m hidden states, after choosing m to optimize model fit (as measured by the Bayesian Information Criteria [24]). We then removed two states that were visited less than 0.1% of the time, resulting in $m = 9$, i.e., a 9-state HMM.

Next, we stratified the state sequences per the heparin dosage groups to learn "action-specific" HMMs. Stratifying datasets for separate HMM parameter estimation is sometimes used to accommodate data heterogeneity (e.g., [14]), and is well-suited for allowing the HMM to depend on the administered heparin dosage. Specifically, we split each patient's data sequence into subsequences, such that all heparin dosage groups administered within each subsequence were identical. That is, for any sequence i beginning at time point t_i and ending at time point T_i, there is a single heparin dosage group d_i such that $d_t = d_i$ for $t \in \{t_i, t_i + 1, ..., T_i - 1\}$. As an example, consider a patient with 5 consecutive data points for $t \in \{0, 1, 2, 3, 4\}$, who received dosage $d_t = 1$ for $t \in \{0, 1, 2\}$ (the first three hours), $d_3 = 2$, and $d_4 = 3$. This sequence would be split at $t = 3$, i.e., one subsequence containing time points $\{0, 1, 2, 3\}$ and one subsequence containing $\{3, 4\}$. All transitions from $t \mapsto t + 1$ involve $d_t = 1$ for the first subsequence and $d_t = 2$ for the second subsequence, while the action administered at the final time point is irrelevant, since data for the next time point is unknown.

Consequently, we trained separate HMMs for each heparin dosage group, based on the patients' subsequences in which each heparin dosage group was administered. For each of these "action-specific" HMMs, the state means, covariance matrices, and initial state probabilities were fixed according to the HMM trained on the full dataset, while their transition probabilities varied. As such, six TPMs were learned, corresponding to the six possible heparin dosage groups. The final HMM model thus consisted of an $m \times 1$ initial state probability distribution, m 14×1 state mean vectors, m 14×14 state covariance matrices, and six $m \times m$ state transition matrices. Note that the final HMM parameters generally predicted that patients would remain in their same latent state (average same-state transition probability of 0.924, which only varied by 0.0304 across dosage groups).

D. A3C Reinforcement Learning Algorithm

Using both the HMM-based and tVAE-based environments, we trained RL agents to maximize expected cumulative discounted reward (using discount factor $\gamma = 0.99$) through a heparin dosage policy $\pi(a_t|s_t)$, which specifies the probability of taking action a_t in state s_t. The agent's action space consisted of the six possible heparin dosages $d_t \in \{0, 1, 2, 3, 4, 5\}$ for all t, and the reward value at each state followed the same reward function from [5], which rewards aPTT values based on their proximity to the therapeutic range of 60–100 seconds: $r(aPTT) = \frac{2}{1+e^{-(aPTT-60)}} - \frac{2}{1+e^{-(aPTT-100)}} - 1$. We trained the RL agents with the Asynchronous Advantage Actor-critic algorithm (A3C; [25]). A3C is on-policy and considered state-of-the-art, and therefore an appropriate example of a powerful reinforcement learning algorithm that can only be used when provided a fully-defined model of the patient environment. The algorithm uses neural networks to map each encountered state into an estimated reward-to-go (the "value network") and a stochastic policy over the possible actions (the "policy network", denoted $\pi_\theta(a_t|s_t)$ and parameterized by θ). From any given state s_t, the agent evaluates a possible action a_t based on its advantage $A(s_t, a_t)$ – the difference between the action's expected reward-to-go and the state's expected reward to-go, averaged across all actions, i.e.,

$$A(s_t, a_t) = \sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k V(s_{t+k}) - V(s_t) \qquad (3)$$

where $k \leq T$ and $V(s_t)$ is the expected reward-to-go from state s_t, approximated with a neural network. The agent then updates the policy network weights (denoted with θ) based on the gradient $\nabla_\theta log(\pi_\theta(a_t|s_t)A(s_t, a_t)) + \eta\nabla_\theta H(\pi_\theta(s_t))$, where $H(\pi_\theta(s_t))$ is the entropy of the policy distribution over all a_t from s_t. Intuitively, the policy update encourages actions that yield large advantage values, regularized with an entropy term that encourages action exploration. The algorithm is "asynchronous" because it performs these updates based on the interactions of multiple agents moving through the environment in parallel. On both the HMM-based and tVAE-based environments, agents were trained for 5000 epochs.

III. Results

In this section, we provide an overview of the patient characteristics, and compare our tVAE-based environment's simulated patient trajectories with actual patient trajectories and those of the benchmark HMM-based environment. Finally, we evaluate the performance of RL policies learned from our tVAE model against those obtained under the benchmark.

A. Patient Characteristics

The sample was 42.4% female, with a median age of 70.4 years (IQR 58.3–79.8) and a median weight of 173 lbs (IQR 145–205). The median sequence length available for each patient was 27 hours (IQR 19-35), and on average, patients received non-zero heparin dosages 96.0% of the time (IQR 85.7%–100%).

B. Simulated Trajectory Characteristics

To evaluate the characteristics of each model's simulated patient trajectories, we used the tVAE and HMM to generate 100 "alternative" trajectories for each patient in the dataset. These trajectories started from the patient's first measurement vector (aPTT and additional clinical predictors), lasted for the same number of hours, and used the same heparin dosage groups that the patient actually received, yielding 206,700 tVAE-generated trajectories and 206,700 HMM-generated trajectories ($100 \times 2,067$ patients).

Fig. 3 shows examples of these simulated aPTT trajectories from both HMM-based and tVAE-based environments for five randomly selected patients. In general, trajectories generated from the HMM-based environment exhibit greater variance than the other trajectories. Consecutive aPTT measurements in HMM-generated trajectories differed by a median of 17.2s, compared to 4.9s for tVAE-based trajectories and 7.0s for actual patient trajectories. Thus, tVAE-based trajectories slightly underestimated the variability in patients' aPTT measurements over time, while HMM-based trajectories overestimated such variability by a factor of about 2.5.

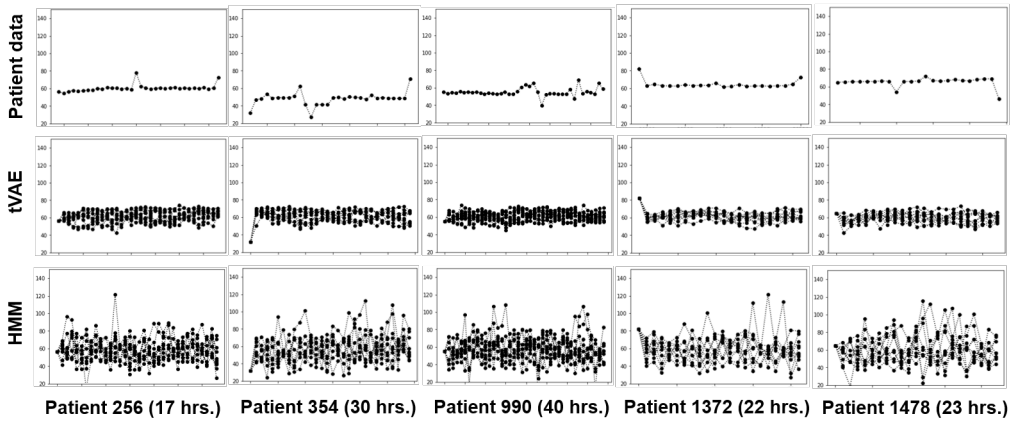

Fig. 3: Comparison of actual and simulated aPTT trajectories for five randomly sampled patients. Plots show ten randomly selected alternate trajectories from the HMM and tVAE models (out of 100 total), compared with each patient's actual aPTT sequence. tVAE-generated trajectories generally appear smoother and exhibit less variance than HMM-generated trajectories.

As suggested by Fig. 3, the HMM-based trajectories also exhibited less dependence between consecutive observations, compared to actual and tVAE-based trajectories. Fig. 4 shows bivariate plots of consecutive aPTT measurements for actual, tVAE-based, and HMM-based trajectories, along with autocorrelation coefficients for each. Consecutive aPTT measurements in actual patient data were correlated at 0.522, though the pattern of dependence was inconsistent; lower aPTT values (40-80) were sometimes followed by much higher measurements (100-150), while higher aPTT measurements (100-150) were sometimes followed by much lower values (40-80). The tVAE and HMM models adapted to this 'noisy' dependence structure differently. The tVAE produced a lower range of aPTT values close to the sample average, while maintaining a similar degree of temporal dependence as actual patient data ($r = 0.432$). The HMM produced a wider range of aPTT values that were much less dependent on their preceding aPTT measurement ($r = 0.191$), as suggested by the sampled trajectories in Fig. 3. Overall, the proportion of variance in aPTT values predicted by the preceding aPTT measurement was 27.2% for actual patient data, 18.7% for tVAE-based trajectories, and 3.6% for HMM-based trajectories.

We also tested tVAE's and HMM's ability to forecast patients' final aPTT values; for each patient, we calculated the mean absolute error (MAE) between their final aPTT measurement and the final estimated aPTT in each of their simulated 'alternative' trajectories. Median MAE was 19.3s for tVAE-generated trajectories and 22.9s for HMM-generated trajectories (equivalent to 0.38 standard deviations difference, after converted to log-scale due to positive skew in MAE values). All results are summarized in Table I.

TABLE I: Descriptive Statistics for tVAE- and HMM-based trajectories. MAE: Mean Absolute Error

	Median Abs. aPTT Change	aPTT Forecasting Median MAE	Average aPTT Autocorrelation
Patients	7.0s	–	0.522
tVAE	4.9s	19.3s	0.432
HMM	17.2s	22.9s	0.191

C. Reinforcement Learning Performance

Using A3C, we trained an RL agent on each environment for 5000 epochs. Fig. 5 presents the physician-prescribed heparin dosages as well as the A3C-prescribed optimal dosages from the HMM- and tVAE-based policies, for each state in the patient's sequence. Note that, since A3C learns stochastic rather than deterministic policies, Fig. 5 presents 50 heparin sequences sampled from each policy.

As seen in the figure, the tVAE-trained policy was more likely to recommend higher heparin dosages than the HMM-trained policy, which almost exclusively recommended a single dosage group. Note that both models' tendency to recommend a single dosage group is similar to the policy behavior in [5], in which patients were prescribed consistent heparin dosages over time. The HMM-trained policy typically recommended the first dosage group (i.e., 1–650 units/mL), while the tVAE-trained policy typically recommended the second one (i.e., 650–800 units/mL). The tVAE-trained policy was overall closer to the average of the dosage groups that patients actually received (2.64), suggesting the HMM-trained policy may under-prescribe heparin. Furthermore, patients who received an average dosage group close to 2 (1.5 to 2.5) saw higher observed cumulative rewards than patients who received an average dosage group close to 1 (0.5 to 1.5; average wards of -0.28 vs. -0.63). Thus, the tVAE-trained policy's higher dosage recommendation was associated with better patient outcomes in our dataset.

We also compared the HMM- and tVAE-trained policies on *both* the HMM- and tVAE-based environments; this allows us to to assess whether one environment was more capable of training policies that were robust to other environment models. We evaluated each policy on both environments over 2000 test runs (with randomly initialized aPTT) that simulated one week (168 hours) of treatment, which is typical of ICU heparin patients [26]. Results are presented in Fig. 6. Both agents

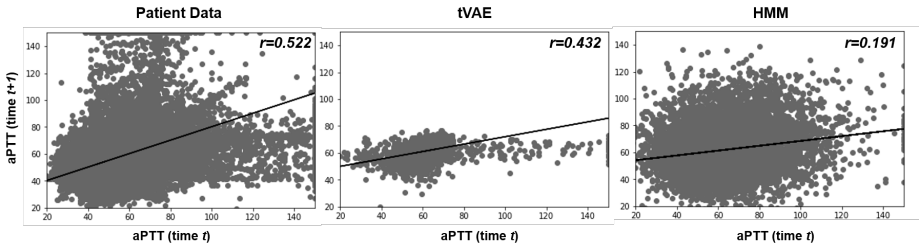

Fig. 4: Plots of consecutive aPTT measurements for actual (left), tVAE-based (middle), and HMM-based (right) trajectories, with lines of best fit and autocorrelation coefficients. tVAE trajectories better match actual patients' aPTT autocorrelation.

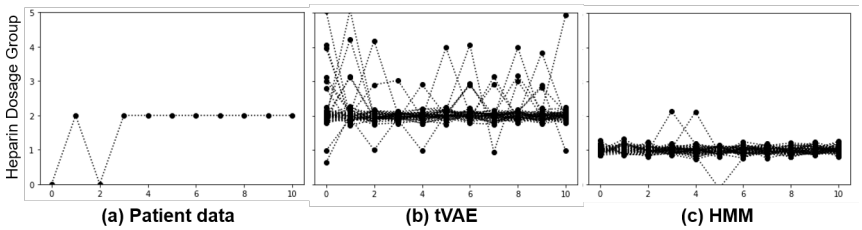

Fig. 5: Simulated heparin dosage sequences for randomly selected patient under the (*a*) physician-prescribed, (*b*) A3C/tVAE-prescribed, and (*c*) A3C/HMM-prescribed policies. Dosage sequences are jittered with random noise for visualization purposes.

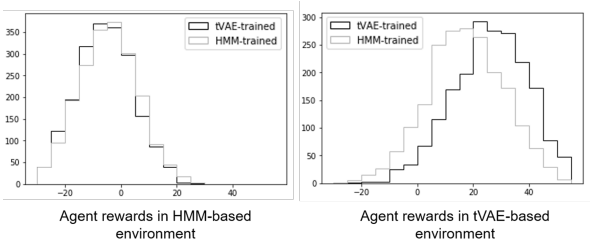

Fig. 6: Cumulative discounted rewards for tVAE- and HMM-trained agents, tested on both environments.

yielded similar cumulative rewards when tested on the HMM-based environment, regardless of the training environment; the HMM-trained agent scored 0.104 standard deviations higher than the tVAE-trained agent (95% margin of error of 0.062). When tested on the tVAE environment, the tVAE-trained agent's returns stochastically dominated those from the HMM-trained agent, and were 0.619 standard deviations higher on average (95% margin of error of 0.059).

IV. Discussion

Patient trajectories generated from a tVAE better matched observed patient trajectories in terms of smoothness, autocorrelation, and forecasted aPTT values, compared with the more commonly used HMM approach. Of these, perhaps the clearest difference between the two RL environments was the volatility of their simulated trajectories. The HMM's oscillatory trajectories may have resulted directly from its mathematical structure. An HMM's emission model must sufficiently cover the observation space, but the number of hidden states is constrained by the need to ensure sufficient visitation to each state. This can lead to high-variance emission distributions; the average aPTT standard deviation across the HMM's nine states was 15.1 seconds, only slightly smaller than the standard deviation of the entire aPTT distribution (16.6 sec).

Thus, even when simulated patients remain in the same latent state, successive clinical measurements may vary widely. This elucidates a broader property of HMMs: they often treat small within-patient changes as "nuisance" variance arising from within the same latent state. This may have been especially true for this dataset, where the average absolute change in patients' consecutive aPTT values (7.0 sec) was smaller than every HMM state's standard deviation. This likely explains the low temporal autocorrelation in HMM-generated trajectories, since successive measurements from our HMM are often independent samples from the same Gaussian distribution (given the HMM's high same-state transition probabilities). It may also explain the HMM's poorer forecasting accuracy.

We also compared the models' abilities to train RL agents to identify optimal dosage policies. Compared to HMM-trained policies, tVAE-trained policies recommended higher heparin dosages that were associated with better patient outcomes and better matched patients' observed dosages. The tVAE-trained agents also performed relatively well in the HMM-based environment, while HMM-trained agents underperformed tVAE-trained agents by about 0.6 standard deviations when tested in the tVAE-environment. Another way of interpreting this finding is that the tVAE environment communicated *unique* information about reward maximization that the HMM-trained agent did not receive from its environment. The HMM's relatively rare state transitions and high within-state variance suggests that it generated high-variance trajectories from a small number of latent states, *regardless* of the policy applied, likely limiting its ability to train effective RL agents. This also may explain the HMM- and tVAE-trained agents' similar returns in the HMM-based environment.

We chose to focus on VAEs because of their success in

the machine learning literature, relatively simple structure, and the opportunity to easily modify them for sequential decision making. We do not claim that VAE's are the only suitable alternative to HMMs in this domain - future research will explore how different generative approaches can better suit healthcare reinforcement learning. We also limited our results to the A3C algorithm, which combines the strengths of multiple classes of algorithms. Still, future work will explore how our results might vary based on the training algorithm. Lastly, we recognize that ICU patients represent a unique problem class, and additional work is needed to establish the advantages of our proposed approach in other domains.

V. Conclusion

Though HMMs have been used extensively for disease modeling, their structural characteristics may lead to insensitivity to trajectory changes and oscillatory predictions with certain datasets. tVAEs and similar generative neural networks offer a data-driven alternative that impose few distributional assumptions and may produce more realistic simulated trajectories.

References

[1] S. Parbhoo et al. "Combining kernel and model based learning for HIV therapy selection". In: *AMIA Summits on Translational Science Proceedings* (2017), pp. 239–248.

[2] J. Pineau et al. "Treating epilepsy via adaptive neurostimulation: A reinforcement learning approach". In: *International Journal of Neural System* 19.4 (2009), pp. 227–240.

[3] A. Raghu et al. "Continuous state-space models for optimal sepsis treatment-A deep reinforcement learning approach". In: *arXiv preprint* (2017). arXiv: 1705.08422.

[4] Z. Zhou et al. "How do tumor cytogenetics inform cancer treatments? Dynamic risk stratification and precision medicine using multi-armed bandits." In: *Preprint* (2019). DOI: http://dx.doi.org/10.2139/ssrn.3405082.

[5] S. Nemati, M.M. Ghassemi, and G.D. Clifford. "Optimal medication dosing from suboptimal clinical examples: A deep reinforcement learning approach". In: *38th Annual International Conference of IEEE Engineering in Medicine and Biology Society* (2016), pp. 2978–2981.

[6] C. Fisher, A. Smith, and J. Walsh. "Deep learning for comprehensive forecasting of Alzheimer's Disease progression". In: *arXiv preprint* (2018). arXiv: 1807.03876.

[7] E. Choi et al. "Generating multi-label discrete patient records using generative adversarial network". In: *arXiv preprint* (2017). arXiv: 1703.06490.

[8] M. Baucum and A. Khojandi. "Improving chronic disease forecasting with synthetically augmented datasets". In: *Preprint* (2019). DOI: 10.13140/RG.2.2.34703.33445.

[9] M. Igl et al. "Deep variational reinforcement learning for POMDPs". In: *arXiv preprint* (2018). arXiv: 1806.02426.

[10] D.P. Kingma and M. Welling. "Auto-encoding variational bayes". In: *ICLR* (2014). arXiv: 1312.6114.

[11] L.E. Baum and T. Petrie. "Statistical inference for probabilistic functions of finite state Markov chains". In: *The Annals of Mathematical Statistics* 37.6 (1966), pp. 1554–1563.

[12] C. Guihenneuc-Jouyaux, S. Richardson, and I.M. Longini Jr. "Modeling markers of disease progression by a hidden Markov process: Application to characterizing CD4 cell decline". In: *Biometrics* 56.3 (2000), pp. 733–741.

[13] Y.Y. Liu et al. "Efficient learning of continuous-time hidden Markov models for disease progression". In: *Advances in Neural Information Processing Systems* (2015), pp. 3600–3608.

[14] T. Ayer, O. Alagoz, and N.K. Stout. "A POMDP approach to personalize mammography screening decisions". In: *Operations Research* 60.5 (), pp. 1019–1034.

[15] Alistair EW Johnson et al. "MIMIC-III, a freely accessible critical care database". In: *Scientific data* 3 (2016), p. 160035.

[16] Alistair EW Pollard Tom J abd Johnson. *The MIMIC-III Clinical Database*. http://dx.doi.org/10.13026/C2XW26. 2016. DOI: 10.13026/C2XW26.

[17] M.A. Smythe et al. "Guidance for the practical management of the heparin anticoagulants in the treatment of venous thromboembolism". In: *Journal of Thrombosis and Thrombolysis* 41.1 (2016), pp. 165–186.

[18] C. Doersch. "Tutorial on variational autoencoders". In: *arXiv preprint* (2016). arXiv: 1606.05908.

[19] J. Walker et al. "An uncertain future: Forecasting from static images using variational autoencoders". In: *European Conference on Computer Vision* (2016), pp. 835–851.

[20] K. Sohn, H. Lee, and X. Yan. "Learning structured output representation using deep conditional generative models". In: *Advances in Neural Information Processing Systems* (2015), pp. 3483–3491.

[21] S. Karsoliya. "Approximating number of hidden layer neurons in multiple hidden layer BPNN architecture". In: *International Journal of Engineering Trends and Technology* 3.6 (2012), pp. 714–717.

[22] S.R. Bowman et al. "Generating sentences from a continuous space". In: *arXiv preprint* (2015). arXiv: 1511.06349.

[23] C. Yan et al. "Re-balancing variational autoencoder loss for molecule sequence generation". In: *arXiv preprint* (2019). arXiv: 1910.00698.

[24] Gideon Schwarz. "Estimating the dimension of a model". In: *Annals of Statistics* 6.2 (1978), pp. 461–464. DOI: 10.1214/aos/1176344136.MR468014.

[25] V. Mnih et al. "Asynchronous methods for deep reinforcement learning". In: *International Conference on Machine Learning* (2016), pp. 1928–1937.

[26] E.M. Gettings et al. "Outcome of postoperative critically ill patients with heparin-induced thrombocytopenia". In: *Critical Care* 10.6 (2006).