Dynamic Measurement Scheduling for Event Forecasting using Deep RL

Chun-Hao Chang * 1 2 3 Mingjie Mai * 1 2 3 Anna Goldenberg 1 2 3

Abstract

Current clinical practice for monitoring patients’ health follows either regular or heuristic-based lab test (e.g. blood test) scheduling. Such practice not only gives rise to redundant measurements accruing cost, but may even lead to unnecessary patient discomfort. From the computational perspective, heuristic-based test scheduling might lead to reduced accuracy of clinical forecasting models. A data-driven measurement scheduling is likely to lead to both more accurate predictions and less measurement costs. We address the scheduling problem using deep reinforcement learning (RL) and propose a general and scalable framework to achieve high predictive gain and low measurement cost, by scheduling fewer, but strategically timed tests. Using simulations we show that our policy outperforms heuristic-based measurement scheduling with higher predictive gain and lower cost. We then learn a scheduling policy for mortality forecasting in the real-world clinical dataset (MIMIC3). Our policy decreases the total number of measurements by 31% without reducing the predictive performance, or improves 3 times more predictive gain with the same number of measurements using off-policy policy evaluation.

1. Introduction

Redundant and expensive screening procedures and lab measurements have increased the overall health care costs (Feldman, 2009). Over-diagnosis rate has also been observed to increase (Ezzie et al., 2007; Moynihan et al., 2012; Hoffman & Cooper, 2012). For example, numerous studies (Iosifina et al., 2013; Pageler et al., 2013) found no evidence that regular blood testing improves diagnosis in hospitals; frequent blood test may even worsen patient’s health (Eyster & Bernene, 1973). Also, clinical heuristics of when to sample the measurements often relies on healthy outpatient volunteers, leading to a suboptimal guideline that has drastically different distribution than the ICU stay patients (PD et al., 2018). To combat the situation, Lee & Maslove (2015) analyzed the redundancy of intensive care unit (ICU) daily measurements by quantifying conditional entropy over first three days of admission, and found the expected amount of novel information in most lab values decreases over time. Dewan et al. (2017) devised a simple rule to reduce the frequency of blood tests by 87% in pediatric ICU. Similarly, Kotecha et al. (2017) showed that the measurement costs can be significantly reduced without increase in mortality or readmission rates in cardiac and surgical ICU. These findings point toward the need for principled data-driven approaches for lab test scheduling to improve both the healthcare system and the patient experience.

Recently developed time-series forecasting models solve the much needed problem of early detection of adverse events (e.g. sepsis) based on sparse and irregular measurements (Ghassemi et al., 2015; Soleimani et al., 2017a; Futoma et al., 2017). However, the timing of these measurements varies from doctor to doctor and from one hospital to another, leading to a drastically different input distribution that may result in inferior classifier performance. Additionally, these classifiers are not often built to provide insights into which measurements were the most helpful to making the model-based prediction given current patient’s condition.

We propose a flexible framework that learns a data-driven and dynamic sampling policy using deep Q-learning. Deep Q-learning, a type of Reinforcement Learning (RL), is a powerful framework that can learn from large amount of retrospective data even when the data does not represent optimal behaviors. In addition, it has been shown to be effective for solving various clinical problems (Raghu et al., 2017; Futoma et al., 2018).

Our framework is a two-tier system. First, we learn an event forecasting model to represent the patient’s condition. Then we train RL to maximize this model’s performance while minimizing the cost of the needed measurements. Our approach of using a learned classifier gives the RL agent immediate reward for each action, which makes credit assignment and RL training comparably easier. We first show that in the simulation, when given a near-perfect classifier,
our method is able to learn a strategically timed measurement scheduling that outperforms all the heuristic-based scheduling. Then we test it on MIMIC3, a real ICU temporal dataset. We compare our learned policies, physician’s policy, as well as random policies using off-policy policy evaluation method, and show that our learned policies increase 3 times more information gain or reduce 31% of measurement costs than physician’s policy.

2. Related Work

2.1. Clinical Event Forecasting Models

Several models have been proposed for event forecasting on irregularly sampled EHR data. Zhang et al. (2017) first used a deep generative variational recurrent neural network (VRNN) to learn feature representations and then used a neural network to predict disease. Futoma et al. (2017) used Multi-output Gaussian process (MGP) to impute the irregularly-sampled timeseries data on the grid points and used those to make predictions via RNN. Soleimani et al. (2018) also used a MGP to impute the missing data, but instead uses a survival model to predict the disease.

2.2. Deep RL in healthcare

Several recent works use RL to learn a treatment plan in ICU. Weng et al. (2017) uses Q-learning to address glycemic control problem for sepsis patients. Prasad et al. (2017) also uses Q-learning to recommend personalized sedation dosage and ventilator support. Raghu et al. (2017) and Komorowski et al. (2018) focuses on treatments for sepsis by learning continuous-state deep Q-learning. The action space is discretized over doses of two drugs commonly given to septic patients. Futoma et al. (2018) improves the Q-learning model using MGP to impute and adopting RNN as the Q-learning network. Wang et al. (2018) optimizes treatment scheduling agent using actor-critic framework in addition to constraint agent policy to match with physician policy. However, this approach is less meaningful when physician policy is sub-optimal, which may be the case for measurement scheduling. All the RL frameworks in healthcare above focus on the treatment scheduling problem. They also do not consider multiple-action settings in our reinforcement learning problems.

Beside treatment scheduling, concurrent work by Cheng et al. (2019) uses fitted Q-iteration to schedule 4 blood tests relevant to diagnosis of sepsis and acute renal failure in ICU setting. Our work differs in three main ways. First, their MDP state formulation doesn’t explicitly capture historical information of the patient. Second, they consider the scheduling of each measurement independent, making it unsuitable in ICU since the patients’ measurement values are highly correlated and sampling policy should be considered jointly across all measurements. Third, they design hand-crafted reward formulation. We compress the informativeness of a new measurement using a predictive model, which not only captures dependency between different measurements, but also makes our method easier to adapt to clinical settings other than sepsis and acute renal failure.

2.3. Active feature acquisition

Several works study the problem of selecting a subset of features to achieve the maximum prediction performance in a non-time-series classifier (Contardo et al., 2016; He et al., 2016; Shim et al., 2017). We tackle time-series feature acquisition problem where historical information matters. This is especially true in a health care setting. In addition, being time-series, the choice of a measurement at the current timepoint affects the performance of the prediction model at a future timepoint.

2.4. Active sensing in medical setting

The focus of active sensing is to determine what and when to measure, which is important when acquiring measurements is costly. Ahuja et al. (2017) handles single-measurement scheduling problem for breast cancer screening by adopting a fixed model-based transition model. Unfortunately, it requires strong assumption, knowing the disease model dynamics, and does not handle multiple types of measurements.

Similarly, Yoon et al. (2018) proposes a method of scheduling measurements to trade between uncertainty in prediction and the measurement cost. They first use multi-directional RNN (M-RNN) to impute and infer missingness in existing measurements and measurements in the next time stamp. They then use another M-RNN to learn an error estimate for each measurement estimate, which is used to construct a confidence interval for each inferred measurement at the next time stamp. Each CI of the measurement is translated to an upper bound and lower bound in prediction. They defined uncertainty as the difference between upper bound and lower bound estimation in prediction. Their model performs a measurement for the next time stamp if the decreases in the uncertainty in prediction exceed the measurement cost. Our approach differs in three ways. First, we use Q-learning to learn policy that maximizes cumulative discounted reward of patient trajectories, while they greedily select measurements that would exceed the utility threshold at the next time stamp. Second, we consider a different definition of information gain - gain in predictive probability. Consider a binary case, where the model produces a wrong estimate, a measurement that encourages a lower uncertainty would not be the ideal choice of action. Third, at test time, instead of evaluating reward at run time, our RL agent speeds up the computation by amortized inferring the corresponding
Q-value by the learned Q function.

3. Methods

Our framework is composed of two parts: a forecasting predictive model and an RL model. See Figure 1 for an overview. For the first part, we train a multi-layer LSTM classifier (Hochreiter & Schmidhuber, 1997) to forecast events of interest using various features. We then frame measurement scheduling question as a sequential feature acquisition problem by using RL. The RL agent decides to sample measurements that increase the LSTM predictive performance while lowering measurement cost given patient’s history up to the given timepoint.

3.1. Deep LSTM Classifier

To handle the sparse time-series data in LSTM, we use mean imputation to impute on evenly spaced grid points for multi-variate, irregularly sampled time series of clinical measurements. Imputed data, missingness indicator and time-invariant features are then feed into a multi-layer LSTM RNN, which produces an event probability. Policy learning A dueling DQN agent observes the last LSTM state and history measurements made at the current time point to learn which time-series measurement should be performed at the next time step. Policy illustration To efficiently learn to make multi-action at each timepoint, the agent sequentially decides whether to take another measurement (M1, M2, M3) or stop making more measurements (Ω) at the current timepoint.

Algorithm 1 Running policy

Input: LSTM hidden state $h_t$, policy $Q$
Output: DQN actions $A_t$
Initialize actions $A_t = \emptyset$
while $\Omega \notin A_t$ do
  $s_t \leftarrow [h_t, A_t]$
  $a \leftarrow \arg\max_{a' \notin A_t} Q(s_t, a')$
  Add $a$ into $A_t$
end while

3.2. Dueling Deep Q Network (DQN)

Dueling DQN factorizes the computation of Q-value into value stream and advantage stream (Wang et al., 2015), i.e.

$$Q(s, a) = V_\eta(f_\xi(s)) + A_\psi(f_\xi(s), a) - \sum_{a'} A_\psi(f_\xi(s), a') \frac{1}{N_{action}}$$

where $\xi$, $\eta$, and $\psi$ are respectively, the parameters of the shared encoder $f_\xi$ of the value stream $V_\eta$, and of the advantage stream $A_\psi$ (Figure 1, Policy Learning).

Sequential Actions Design Given $K$ possible measurements, at any given time point the agent has to decide among $2^K$ large combinations of measurements, which is clearly unscalable to large $K$. In addition, naively assigning reward to a set of actions without considering the commonality between sets of actions lead to more difficult learning and gets lower sample efficiency. To overcome these two difficulties, we design the RL to take actions in a sequential manner to overcome the large action space and assign separate reward to each individual action (Figure 1, Policy Illustration). Specifically, we include a new action $\Omega$ to represent moving...
We represent RL agent’s action vector of size $\Omega$ or patient’s discomfort. In this work we simply define it as could represent its monetary cost, operational complexity parameter and should be defined by the domain expert which $\Omega$ into RL actions.

**Action** We add a new time-passing action $\Omega$ into RL actions. We represent RL agent’s action $a_v$ as a multi-hot encoding vector of size $K + 1$. For $k \in [1, K]$, $a_{v,k} = 1$ denotes the $k^{th}$ measurement is scheduled at this timepoint, otherwise $a_{v,k} = 0$.

**Reward** We define the reward function as a linear combination of the information gain $g_\mathcal{I}$ and measurement cost $c$, i.e. $r(s_v, a_v) = g_\mathcal{I}(s_v, a_v) - \lambda \ast c(a_v)$, where $\lambda$ represents the step in the MDP (to differentiate between timepoint $t$). To encourage the predictive performance of the classifier $\mathcal{I}$, we set the information gain $g(s_v, a_v)$ as the probability change of the classifier $\mathcal{I}$, conditioned on the label, i.e.

$$g_\mathcal{I}(s_v, a_v) = \begin{cases} \Delta p, & \text{if } \text{label} = 1 \\ -\Delta p, & \text{otherwise} \end{cases}$$

The cost of scheduling a measurement $c(a_v)$ is a hyperparameter and should be defined by the domain expert which could represent its monetary cost, operational complexity or patient’s discomfort. In this work we simply define it as the number of measurements except the action $\Omega$ i.e.

$$c(a_v) = \begin{cases} 1, & \text{if } a_v \neq \Omega \\ 0, & \text{otherwise} \end{cases}$$

**State** We use a multi-hot encoding $m_v$ to denote the measurements that have been scheduled by the agent at the current timepoint. We use the concatenation of last LSTM layer representation $h_t$ of patient’s history and history measurement $m_v$ as the input to the agent, denoted $s_v = [h_t, m_v]$.

**Learning** We generate RL experience tuples $[h, m, h', m', a, r, \gamma]$ in a sequential manner (Algorithm 3). We generate two kinds of experience, time-passing experiences and measurement experiences. The time-passing experience assigns the probability change due to time shift from $t - 1$ to $t$ to the action $\Omega$. The measurement experience assigns the reward to a specific measurement action. Since multiple measurements are recorded at the same time and we do not know the underlying chronological order, we randomly generate training experiences based on a random order of the measurements at the same time point $t$ as a way of data augmentation. For example, if $M1, M2, M3$ were recorded at a timepoint, the action order could be $(M1, M2, M3, \Omega), (M2, M1, M3, \Omega), (M3, M2, M1, \Omega)$ etc. Note that we do not decay the reward for these experiences since there is no actual time passing. Also, the sum of probability changes $\Delta p$ due to measurements made at the same time $t$ will be the same no matter which action order is selected. We set reward discounted factor $\gamma_v = 1$ for these measurement experiences. Also, we also do not update the hidden state $h$ for measurement experiences within the same time $t$ since we only observe the measurements until the next time point.

We optimize the RL agent by minimizing the Bellman-equation square error (Algorithm 2). Note that when calculating the $Q_{\text{target}}$, the best action considered can not be the action already performed in $m'$. i.e.

$$Q_{\text{target}}(s, a, s') = r(s, a) + \gamma \max_{a' \notin m'} Q(s', a')$$
We use regression-based estimator (Jiang & Li, 2015) to estimate the values of physician and our learned policies using physician collected data. We do not use importance-sampling based estimator since it would require an exact match with physician actions under our deterministic policy, which is virtually impossible in our high-dimensional action space. Besides, it is also shown to be unstable when using with regression-based estimator in Liu et al. (2018).

We use per-time value estimator to evaluate our learned policies (Algorithm 4). First, we train a regression model \( \phi \) that maps the state-action pair to the probability changes. Specifically, at each time \( t \), the input is the concatenation of the latent state \( h_t \) and multi-hot encoding of actions \( a_t \) performed at time \( t \), and the output is the probability changes \( \Delta P = P_{t+1} - P_t \). We use feed-forward neural network to fit the regression with all hyperparameters listed in Appendix Table 5. Then, for each patient at each time \( t \), we estimate the next time \( t+1 \) what is the corresponding reward if the specified action is performed. And we obtain estimated cumulative information gain \( G \) by summing over all estimated information gain \( g_\phi \) across all patients and all time \( t \) with decay as \( \gamma^t \).

### 3.3. Off-Policy Policy Evaluation

We use regression-based estimator (Jiang & Li, 2015) to estimate the values of physician and our learned policies using physician collected data. We do not use importance-sampling based estimator since it would require an exact match with physician actions under our deterministic policy, which is virtually impossible in our high-dimensional action space. Besides, it is also shown to be unstable when using with regression-based estimator in Liu et al. (2018).

We use per-time value estimator to evaluate our learned policies (Algorithm 4). First, we train a regression model \( \phi \) that maps the state-action pair to the probability changes. Specifically, at each time \( t \), the input is the concatenation of the latent state \( h_t \) and multi-hot encoding of actions \( a_t \) performed at time \( t \), and the output is the probability changes \( \Delta P = P_{t+1} - P_t \). We use feed-forward neural network to fit the regression with all hyperparameters listed in Appendix Table 5. Then, for each patient at each time \( t \), we estimate the next time \( t+1 \) what is the corresponding reward if the specified action is performed. And we obtain estimated cumulative information gain \( G \) by summing over all estimated information gain \( g_\phi \) across all patients and all time \( t \) with decay as \( \gamma^t \).

### 4. Results

#### 4.1. Simulation

The goal of this simulation is to study the performance of the RL agent given a near-perfect classifier. Here, we simulate a terminal event forecasting task, use a softmax classifier to produce rewards and then train a Dueling DQN agent for measurement scheduling using the rewards generated by the classifier.

**Simulation data** Patient clinical status is simulated to be a binary time series generated under a two-state Markov model: \( M = \{0, 1 : 0 = \text{Healthy}, 1 = \text{Critical}\} \). A consecutive sequence of five 1s in the status series indicates the onset of a terminal event. We simulate patients to have different trajectory lengths \( T \) indexed by \( t \) and 10 types of input signals indexed by \( k \), as follows. Let \( \epsilon_{t,k} \sim N(0, 0.1) \). The first five types of measurements \( k \in [1, 5] \) \( y_t, k = 1 + \epsilon_{t,k} \) when \( S_t = 1 \) and \( -1 + \epsilon_{t,k} \) otherwise. The last five types of measurements \( k \in [6, 10] \) are \( \epsilon_{t,k} \) independent of \( S_t \). We randomly remove 50% of the values from the generated matrix to introduce missingness creating a more realistic scenario. In the case of missingness, the measurement value is set to 0. The measurements are designed such that first five types of measurements have increasing importance while the last five measurements are noise.

**Designed classifier** We design a classifier considering the feature importance vector \( \{f_k\}_{k=1}^{10} = (1 \ 2 \ 3 \ 4 \ 5 \ 0 \ 0 \ 0 \ 0 \ 0) \). The classifier takes in measurements of the 5 most recent timepoints \( \{y_{t,k}\}_{k=1}^{10} \) \( t' \), where \( t' \) is the current time. Let \( \eta \) denote a time decay factor, where past measurements are less important. The classifier then forecasts whether the patient experiences a terminal event within 5 future timepoints with \( p(a_{t+5} = 1) = \text{softmax}(\sum_{t=t'}^{t'+5} \sum_{k=1}^{10} y_{t,k} \cdot f_k \cdot \eta^t) \). The classifier increases the certainty of a terminal event when it discovers more critical signals in the measurement values. To see whether the agent can distinguish features with different importance, we employ a uniform action cost \( c(a_{t}) = 1 \). The RL agent takes \( \{y_{t,k}\}_{k=1}^{10} \) \( t'=t'-4 \) as input. We set reward discount factor \( \gamma = 0.990 \) in this task.

We simulated a dataset of 5,000 patient trajectories with
Dynamic Measurement Scheduling for Event Forecasting using Deep RL

Figure 3. An example trajectory of our dueling DQN policy in the simulation. Blue color denotes all the measurements for healthy state and red for critical state. Darker color represents the measurements taken by the agent. For example, $M_4$ and $M_5$ are taken all the time to probe the state of the patient.

$T \in [25, 50]$ according to the scheme above. 5% of the patients end up with a terminal event. We learn several dueling DQN agents by varying trade-off factor $\lambda$. We include several baselines that resemble the heuristic-based test scheduling. One of the baseline policies is randomly selecting $x$ informative measurements (Random_informative), where $x \in [1, 5]$. Another class of baseline policies is randomly selecting $x$ measurements (Random), where $x \in [1, 10]$.

As we are varying $\lambda$, we learn a range of policies that trade off between action frequency and predictive probability of detecting the terminal event (Figure 2). Under the same action frequency, our learned dueling DQN agent consistently outperform baseline policies in terms of predictive probability of detecting disease.

We show an example patient trajectory of our dueling DQN policy in Figure 3. It always selects the most and the second most informative features ($M_4, M_5$) to probe which state the patient is in. It sequentially selects the other informative features ($M_3, M_2, M_1$) whenever it finds the patient is in a critical state. It doesn’t select any noisy features to avoid accruing total measurement cost.

4.2. Results on MIMIC3

Here we test our policy on a real-world ICU dataset MIMIC3 to gain better clinical sampling policy. The details of our preprocessing of MIMIC3 are in the Appendix A. First, we train a mortality forecasting model. Our task is to predict if patient dies within 24 hours given 39 time-series measurements and 38 static covariates. We show that we train a well performing RNN classifier: with sufficient information

RNN vastly outperforms baselines such as random forest that do not consider long-term dependency (Table 1). The details of the classifier training are in Appendix B. We then show that combining the classifier and an RL model, we are able to learn clinically relevant policies from off-line data and we show our policies perform better than clinician’s policy using off-policy policy evaluation.

Training policies and Off-policy evaluation We take each patient’s last 24 hours and discretize the experience into 30-minutes intervals, leading to 48 time points. We remove the patients with fewer than 12 hours of recording or less than 5 measurements available. We set $\gamma = 0.95$ to encourage the agent to increase predictive performance
Dynamic Measurement Scheduling for Event Forecasting using Deep RL

Figure 5. Focus view of Figure 4. Compared to physician policy, our policies reduce around 31\% of the action costs under the same information gain, or 3 times increase of the information gain under the same action costs relative to the lowest information gain.

Table 1. The test set performances of the trained classifiers in 24 hour mortality prediction.

|      | AUC   | AUPR  |
|------|-------|-------|
| LR   | 0.931 | 0.752 |
| RF   | 0.935 | 0.756 |
| RNN  | 0.950 | 0.803 |

earlier rather than later. We vary our DQN architecture and action cost coefficient \( \lambda \) in a wide range of \( 1e^{-4} \) to \( 1e^{-2} \) to train various policies, and select the best performing policies based on validation set. We list all the hyperparameters in Appendix Table 4. We use regression based off-policy evaluation to evaluate our agent policies, physician policy and random policies shown in Figure 4. Ideally, a great policy should have low action frequency and high information gain. By interpolating across various DQN performing points, we can get a frontier of performance curve for our DQN agent. Using this frontier, compared to physician policy, our policies reduce action costs by 31\% under the same information gain, or increase the information gain 3 times keeping the same action costs relative to the lowest information gain (Figure 5). The lowest information gain is the information gain when no measurements are taken. Maybe surprisingly, sampling all the measurements (relative action cost as 1) do not produce the policy with the highest information gain. We think it is because the classifier tends to make mistakes on some measurements so measuring everything decreases the classifier’s performance. Or it could be the measurement itself is rarely measured or noisy and that confuses the classifier.

We compare our policies’ action frequency with physician’s action frequency to gain insights from the learned policies (Figure 7). We show our policies with increasing action frequency, from left to right, top to bottom. The most frequent measurements performed by physicians within the last 24 hours (red box) are Hemoglobins, Phosphate, Systolic blood pressure and Fraction inspired oxygen (FiO2) (see Appendix Table 3 for a full list), indicating the clinical importance of these 4 measurements. It is reassuring that the closest policy with the same action costs (black box) also focus on these 4 most frequent measurements with focus on Phosphate and FiO2. We find these 2 are strongly correlated with the death in ICU due to Hypophosphatemia (Geerse et al., 2010; Miller et al.) and serving as important functional indicators or hyperoxia (Damiani et al., 2014; Ramanan & Fisher, 2018). As we increase measurement costs, our policies select other features like Calcium Ionized, Mean blood pressure and Oxygen Saturation, indicating the importance of these features for the task of mortality prediction.

Figure 6 shows examples of our policy sampling strategy for the last 24 hours of 3 dying patients. (Left) Physician policy. (Middle) Probability change due to physician’ measurement. (Right) Actions performed by RL. Our agent policy makes decision based on the history of the physician’s sampled measurements. The MIMIC3 icustay_id of these three encounters are 256778, 218183, 227666, respectively.
Dynamic Measurement Scheduling for Event Forecasting using Deep RL

recently. This shows that our policies dynamically sample adaptive to the patient’s condition, rather than a simple rule-of-thumb measurement strategy.

5. Discussion and Future Work

In this work, we propose a flexible method for measurement scheduling. We show that our scheduling policy achieves better long-term predictive power with comparable or lower measurement costs in both simulations and on MIMIC3.

In this study, we assume sampling lab tests at the current time point gives us the measurement values at the immediate next timepoint. Indeed, some lab tests can take hours in hospital to get the analyzed result back. In our framework, we can incorporate the time constraint and relax this assumption by delaying the reward that RL agent receives to a later timepoint to adjust for this bias.

While in this work we only learn from one fixed classifier, dataset shift or covariate shift problem can arise due to change in scheduling policy. One way to solve this is to sequentially train the classifier on newly collected data and retrain RL agent on the classifier. Another concern is that our RL agent might learn to sample features that are over-fitting to the classifier. To avoid this phenomenon, we plan to scale our work by using an ensemble of classifiers to reduce over-fitting and produce a robust reward.

Using regression-based value estimator is known to have provably low variance when the MDP is well estimated. However, bias can be introduced since real-world problems usually have a large state space, and many state-action pairs will not be observed in the data. We plan to use more complex model for the regression based value estimator to capture the underlying dynamics. Further, incorporating causal thinking in RL framework might help learn safer policies, for example, recent work by Kallus & Zhou (2018) presents a model for personalized decision policy learning in the presence of unobserved confounding and its application to acute ischaemic stroke treatment.

We didn’t incorporate treatment information which can be valuable for improving classifier performance. Recent literature on incorporating treatment information to model physiologic signals using causal inference can help address this issue (Schulam & Saria, 2017; Soleimani et al., 2017b).

References

Ahuja, K., Zame, W., and van der Schaar, M. DPSCREEN: Dynamic Personalized Screening. In Advances in Neural
Dynamic Measurement Scheduling for Event Forecasting using Deep RL

Cheng, L.-F., Prasad, N., and Engelhardt, B. E. An optimal policy for patient laboratory tests in intensive care units. *Proceedings of the Pacific Symposium on Biocomputing (PSB)*, 2019. URL https://psb.stanford.edu/psb-online/proceedings/psb19/cheng_l.pdf.

Contardo, G., Denoyer, L., and Artières, T. Sequential cost-sensitive feature acquisition. In *International Symposium on Intelligent Data Analysis*, pp. 284–294. Springer, 2016.

Dewan, M., Galvez, J., Polsky, T., Kreher, G., Kraus, B., Ahumada, L., McCloskey, J., and Wolfe, H. Reducing unnecessary postoperative complete blood count testing in the pediatric intensive care unit. *The Permanente Journal*, 21, 2017.

Eyster, E. and Bernene, J. Nosocomial anemia. *JAMA*, 223(1):73–74, 1973.

Ezzie, M. E., Aberegg, S. K., and O’Brien, J. M. Laboratory testing in the intensive care unit. *Critical Care Clinics*, 23(3):435–465, 2007.

Futoma, J., Hariharan, S., Sendak, M., Brajer, N., Clement, M., Bedoya, A., O’Brien, C., and Heller, K. An Improved Multi-Output Gaussian Process RNN with Real-Time Validation for Early Sepsis Detection. *arXiv:1708.05894 [stat]*, August 2017. arXiv: 1708.05894.

Geerse, D. A., Bindels, A. J., Kuiper, M. A., Roos, A. N., Spronk, P. E., and Schultz, M. J. Treatment of hypophosphatemia in the intensive care unit: a review. *Critical Care*, 14(4):R147, 2010.

Ghassemi, M., Pimentel, M. A., Naumann, T., Brennan, T., Clifton, D. A., Szolovits, P., and Feng, M. A multivariate timeseries modeling approach to severity of illness assessment and forecasting in icu with sparse, heterogeneous clinical data. 2015.

Harutyunyan, H., Khachatrian, H., Kale, D. C., and Galstyan, A. Multitask learning and benchmarking with clinical time series data. *arXiv preprint arXiv:1703.07771*, 2017.

He, H., Mineiro, P., and Karampatziakis, N. Active information acquisition. *arXiv preprint arXiv:1602.02181*, 2016.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Hoffman, J. R. and Cooper, R. J. Overdiagnosis of disease: a modern epidemic. *Archives of internal medicine*, 172 (15):1123–1124, 2012.

Iosifina, I., Merkeley, H., Cessford, T., Geller, G., Amiri, N., Baradaran, N., Norena, M., Ayas, N., and Dodek, P. M. Implementation of an on-demand strategy for routine blood testing in icu patients. In *D23. QUALITY IMPROVEMENT IN CRITICAL CARE*, pp. A5322–A5322. Am Thoracic Soc, 2013.

Jiang, N. and Li, L. Doubly robust off-policy value evaluation for reinforcement learning. *arXiv preprint arXiv:1511.03722*, 2015.

Johnson, A. E., Pollard, T. J., Shen, L., Li-wei, H. L., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Celi, L. A., and Mark, R. G. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035, 2016.

Kallus, N. and Zhou, A. Confounding-robust policy improvement. *arXiv preprint arXiv:1805.08593*, 2018.

Komorowski, M., Celi, L. A., Badawi, O., Gordon, A. C., and Faisal, A. A. The artificial intelligence clinician learns optimal treatment strategies for sepsis in intensive care. *Nature Medicine*, 24(11):1716, 2018.

Kotecha, N., Shapiro, J. M., Cardasis, J., and Narayanswami, G. Reducing unnecessary laboratory testing in the medical icu. *The American journal of medicine*, 130(6):648–651, 2017.

Lee, J. and Maslove, D. M. Using information theory to identify redundancy in common laboratory tests in the intensive care unit. *BMC medical informatics and decision making*, 15(1):59, 2015.

Lipton, Z. C., Kale, D. C., and Wetzel, R. Modeling missing data in clinical time series with rnns. 2017.

Liu, Y., Gottesman, O., Raghu, A., Komorowski, M., Faisal, A., Doshi-Velez, F., and Brunskill, E. Representation balancing mdps for off-policy policy evaluation. *CoRR*, abs/1805.09044, 2018.
Miller, C. J., Doepker, B. A., Springer, A. N., Exline, M. C., Phillips, G., and Murphy, C. V. Impact of serum phosphate in mechanically ventilated patients with severe sepsis and septic shock. *Journal of intensive care medicine*, pp. 0885066618762753.

Moynihan, R., Doust, J., and Henry, D. Preventing overdiagnosis: how to stop harming the healthy. *BMJ: British Medical Journal (Online)*, 344, 2012.

Pageler, N. M., Franzon, D., Longhurst, C. A., Wood, M., Shin, A. Y., Adams, E. S., Widen, E., and Cornfield, D. N. Embedding time-limited laboratory orders within computerized provider order entry reduces laboratory utilization. *Pediatric Critical Care Medicine*, 14(4):413–419, 2013.

PD, T., H, D., M, F., and et al. Assessment of intensive care unit laboratory values that differ from reference ranges and association with patient mortality and length of stay. *JAMA Network Open*, 1(7): e184521–, 2018. doi: 10.1001/jamanetworkopen.2018.4521. URL +http://dx.doi.org/10.1001/jamanetworkopen.2018.4521.

Prasad, N., Cheng, L.-F., Chivers, C., Draugelis, M., and Engelhardt, B. E. A reinforcement learning approach to weaning of mechanical ventilation in intensive care units. *arXiv preprint arXiv:1704.06300*, 2017.

Raghu, A., Komorowski, M., Celi, L. A., Szolovits, P., and Ghassemi, M. Continuous state-space models for optimal sepsis treatment—a deep reinforcement learning approach. *arXiv preprint arXiv:1705.08422*, 2017.

Ramanan, M. and Fisher, N. The association between arterial oxygen tension, hemoglobin concentration, and mortality in mechanically ventilated critically ill patients. *Indian journal of critical care medicine: peer-reviewed, official publication of Indian Society of Critical Care Medicine*, 22(7):477, 2018.

Schulam, P. and Saria, S. What-if reasoning with counterfactual gaussian processes. *History*, 100:120, 2017.

Shim, H., Hwang, S. J., and Yang, E. Why pay more when you can pay less: A joint learning framework for active feature acquisition and classification. *arXiv preprint arXiv:1709.05964*, 2017.

Soleimani, H., Subbaswamy, A., and Saria, S. Treatment-response models for counterfactual reasoning with continuous-time, continuous-valued interventions. *arXiv preprint arXiv:1704.02038*, 2017b.

Soleimani, H., Hensman, J., and Saria, S. Scalable joint models for reliable uncertainty-aware event prediction. *IEEE transactions on pattern analysis and machine intelligence*, 40(8):1948–1963, 2018.

Wang, L., Zhang, W., He, X., and Zha, H. Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2447–2456. ACM, 2018.

Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., and De Freitas, N. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2015.

Weng, W.-H., Gao, M., He, Z., Yan, S., and Szolovits, P. Representation and reinforcement learning for personalized glycemic control in septic patients. *arXiv preprint arXiv:1712.00654*, 2017.

Yoon, J., Zame, W. R., and van der Schaar, M. Deep sensing: Active sensing using multi-directional recurrent neural networks. 2018.

Zhang, S., Xie, P., Wang, D., and Xing, E. P. Medical diagnosis from laboratory tests by combining generative and discriminative learning. *arXiv preprint arXiv:1711.04329*, 2017.
A. MIMIC3 Preprocessing for Survival Forecasting

We use the publicly available dataset MIMIC3 (Johnson et al., 2016) and then follow the preprocessing of Harutyunyan et al. (2017) for the in-hospital mortality prediction task. It excludes the neonatal and pediatric patients and patients with multiple ICU stays. The training set consists of 35,725 patients with 10.81% mortality rate, and test set has 6,294 patients with 9.94% mortality rate. We then split 15% of our training set as our validation set.

For classifier training, we uniformly take 6 timepoints within the last 24 hours of each patient trajectory. For each prediction point, we set the label as 1 if the patient dies in the encounter and 0 otherwise. We only include the prediction points with at least 3 hours of history and 5 measurement values. For RL, we take the last 24 hours of each dying patient and discretize it into 30 minutes interval. We only include the patients with at least 12 hours to remove unstable trajectories.

We select 38 static demographic features and clinical. Age, Gender, Ethnicity, congestive heart failure, cardiac arrhythmias, valvular disease, pulmonary circulation, peripheral vascular, hypertension, paralysis, other neurological, chronic pulmonary, diabetes uncomplicated, diabetes complicated, hypothyroidism, renal failure, liver disease, peptic ulcer, aids, lymphoma, metastatic cancer, solid tumor, rheumatoid arthritis, coagulopathy, obesity, weight loss, fluid electrolyte, blood loss anemia, deficiency anemias, alcohol abuse, drug abuse, psychoses, depression.

The measurement with the largest count is heart rate. We select 39 time-series measurements with counts at least 1% of the count of heart rate. We show the feature choices and their count in Appendix Table 2. We further log-transform and standardize time series measurement values per feature.

B. Classifier Training Details and Performances

We train the RNN as follows. We use LSTM with 1 hidden layer of 32 nodes. We regularize the neural network with $\lambda = 1e^{-5}$ as $\ell_2$ regularization and dropout rate of 0.3 for input and output layer, and 0.5 for the hidden layer. We use mean imputation (set the missing value as the feature mean) for the time-series features, and add missingness indicators for each feature (Lipton et al.). We discretize the time series into 30 minutes interval and take mean value if there are multiple measurements per interval. The RNN takes these 30-minutes discretized grid point for up to 24 hours time point to classify. We train two other baselines: Logistic Regression (LR) with $\lambda = 1e^{-5}$ as $\ell_2$ regularization and Random Forest (RF) with 500 trees. We concatenate all the features across all the time points.
| Feature                              | Count  | Relative Count % |
|--------------------------------------|--------|------------------|
| Anion gap                            | 213442 | 0.051            |
| Bicarbonate                          | 219802 | 0.052            |
| Blood urea nitrogen                  | 220854 | 0.053            |
| Calcium (total)                      | 185718 | 0.044            |
| Chloride (blood)                     | 225476 | 0.054            |
| Creatine kinase                      | 44459  | 0.011            |
| Creatinine (blood)                   | 221715 | 0.053            |
| Diastolic blood pressure             | 3929745| 0.935            |
| Glasgow coma scale total             | 627577 | 0.149            |
| Glucose (blood)                      | 313798 | 0.075            |
| Heart Rate                           | 4204926| 1.0              |
| Hematocrit                           | 253045 | 0.06             |
| Hemoglobin                           | 196859 | 0.047            |
| Magnesium                            | 218030 | 0.052            |
| Mean blood pressure                  | 3904218| 0.928            |
| Mean corpuscular hemoglobin          | 194995 | 0.046            |
| Phosphate                            | 189261 | 0.045            |
| Platelets                            | 205492 | 0.049            |
| Potassium                            | 241110 | 0.057            |
| Prothrombin time                     | 139231 | 0.033            |
| Red blood cell count (blood)         | 194997 | 0.046            |
| Sodium                               | 229893 | 0.055            |
| Systolic blood pressure              | 3930865| 0.935            |
| Temperature (C)                      | 797435 | 0.19             |
| White blood cell count (blood)       | 196268 | 0.047            |
| CO2 (ETCO2, PCO2, etc.)              | 263161 | 0.063            |
| Oxygen saturation                    | 101518 | 0.024            |
| Partial pressure of carbon dioxide   | 263153 | 0.063            |
| Partial thromboplastin time          | 149675 | 0.036            |
| pH (blood)                           | 285076 | 0.068            |
| Bilirubin (total)                    | 47707  | 0.011            |
| Lactate                              | 84510  | 0.02             |
| Lactic acid                          | 89347  | 0.021            |
| Positive end-expiratory pressure     | 53689  | 0.013            |
| Fraction inspired oxygen             | 375335 | 0.089            |
| Calcium ionized                      | 140283 | 0.033            |
| Alanine aminotransferase             | 46850  | 0.011            |
| Alkaline phosphate                   | 45809  | 0.011            |
| Asparate aminotransferase            | 46808  | 0.011            |
| Feature                                | Relative action frequency |
|----------------------------------------|---------------------------|
| Anion gap                              | 0.0021                    |
| Bicarbonate                            | 0.0118                    |
| Blood urea nitrogen                    | 0.0022                    |
| Calcium (total)                        | 0.0120                    |
| Chloride (blood)                       | 0.0022                    |
| Creatine kinase                        | 0.0122                    |
| Creatinine (blood)                     | 0.0059                    |
| Diastolic blood pressure               | 0.0101                    |
| Glasgow coma scale total               | 0.0046                    |
| Glucose (blood)                        | 0.0125                    |
| Heart Rate                             | 0.0022                    |
| Hematocrit                             | 0.0123                    |
| Hemoglobin                             | 0.0642                    |
| Magnesium                              | 0.0029                    |
| Mean blood pressure                    | 0.0085                    |
| Mean corpuscular hemoglobin            | 0.0148                    |
| Phosphate                              | 0.0662                    |
| Platelets                              | 0.0143                    |
| Potassium                              | 0.0112                    |
| Prothrombin time                       | 0.0023                    |
| Red blood cell count (blood)           | 0.0024                    |
| Sodium                                 | 0.0122                    |
| Systolic blood pressure                | 0.0635                    |
| Temperature (C)                        | 0.0111                    |
| White blood cell count (blood)         | 0.0029                    |
| CO2 (ETCO2, PCO2, etc.)               | 0.0059                    |
| Oxygen saturation                      | 0.0073                    |
| Partial pressure of carbon dioxide     | 0.0103                    |
| Partial thromboplastin time            | 0.0116                    |
| pH (blood)                             | 0.0009                    |
| Bilirubin (total)                      | 0.0134                    |
| Lactate                                | 0.0068                    |
| Lactic acid                            | 0.0112                    |
| Positive end-expiratory pressure       | 0.0126                    |
| Fraction inspired oxygen               | 0.0643                    |
| Calcium ionized                        | 0.0133                    |
| Alanine aminotransferase               | 0.0193                    |
| Alkaline phosphate                     | 0.0112                    |
| Asparate aminotransferase             | 0.0075                    |
Table 4. Hyperameters and ranges for Dueling DQN

| Parameter                  | Range                                      |
|----------------------------|--------------------------------------------|
| Num. of representation layers | \{1, 2, 3, 4\}                           |
| Num. of dueling layers      | \{1, 2, 3, 4\}                            |
| Dim. of NN layers           | \{16, 32, 64, 128\}                       |
| Learning rate               | \{5e − 2, 1e − 3, 5e − 3, 1e − 4, 5e − 4, 1e − 5, 5e − 5, 1e − 6\} |
| L2 reg. constant            | \{5e − 1, 1e − 1, 5e − 2, 1e − 2, 5e − 3, 1e − 3, 5e − 4, 1e − 4\} |
| Dropout keep prob.          | \{1.0, 0.9, 0.8, 0.7, 0.6, 0.5\}           |
| Training batch size         | \{32, 64, 128, 256, 512\}                 |
| Action cost coefficient \(\lambda\) | \{1e − 4, 5e − 4, 1e − 3, 5e − 3, 1e − 2\} |

Table 5. Hyperameters and ranges for information gain estimator

| Parameter                  | Range                                      | The best model |
|----------------------------|--------------------------------------------|----------------|
| Num. of representation layers | \{1, 2, 3, 4\}                           | 1              |
| Dim. of NN layers           | \{16, 32, 64, 128, 256, 512\}             | 64             |
| Learning rate               | \{1e − 2, 1e − 3, 1e − 4, 1e − 5, 1e − 6, 1e − 7\} | 1e-3           |
| L2 reg. constant            | \{1e − 2, 1e − 3, 1e − 4, 1e − 5, 1e − 6, 1e − 7\} | 1e-4           |
| Dropout keep prob.          | \{1.0, 0.9, 0.8, 0.7, 0.6, 0.5\}           | 0.7            |
| Training batch size         | \{64, 128, 256, 512, 1024\}               | 64             |