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
Imagine a patient in critical condition. What and when should be measured to forecast detrimental events, especially under the budget constraints? We answer this question by deep reinforcement learning (RL) that jointly minimizes the measurement cost and maximizes predictive gain, by scheduling strategically-timed measurements. We learn our policy to be dynamically dependent on the patient’s health history. To scale our framework to exponentially large action space, we distribute our reward in a sequential setting that makes the learning easier. In our simulation, our policy outperforms heuristic-based scheduling with higher predictive gain and lower cost. In a real-world ICU mortality prediction task (MIMIC3), our policies reduce the total number of measurements by 31% or improve predictive gain by a factor of 3 as compared to physicians, under the off-policy policy evaluation.

1. Introduction
Redundant and expensive screening procedures and lab measurements have increased the overall health care costs (Feldman, 2009). This phenomenon, either due to commercial interests or over-concern, has been observed in numerous clinical practices (Hoffman & Cooper, 2012; Brodersen et al., 2018). For example, numerous studies (Iosifina et al., 2013; Pageler et al., 2013) found no evidence that regular blood testing improves diagnosis; frequent blood tests may even cause anemia and infection (Salisbury et al., 2011). To combat the situation, 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 re-admission rates in cardiac and surgical ICU. These findings point toward the need for principled data-driven approaches for lab test scheduling to improve the healthcare system.

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 help the most to make the prediction given current patient’s condition.

We propose a scalable and 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 promising 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. Compared to directly using the event as reward, our approach of using event probabilities from a learned classifier gives the RL immediate reward for every action, making reward assignment and training comparably easier. To handle the exponentially large action space in the measurement scheduling problems, we use sequential action setting that successfully handle the number of measurements in an unprecedented scale.

We show that in the simulation setting, when given a near-perfect classifier, our method is able to learn a strategically timed measurement scheduling that outperforms all the heuristic-based scheduling. We then 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 (OPPE) method, showing that our learned policies reduce measurement costs by 31% or increase information gain by a factor of 3 compared to physician’s policy. Our data preprocessing and code are avail-

---

*Equal contribution
1University of Toronto, Toronto, ON, Canada
2Vector Institute, Toronto, ON, Canada
3The Hospital for Sick Children, Toronto, ON, Canada
Correspondence to: Chun-Hao Chang <kingsley@cs.toronto.edu>.

able online at https://github.com/zzzace2000/autodiagnosis.

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 representation and then used a neural network to predict disease. Li & Marlin (2016); Futoma et al. (2017) used multi-output Gaussian process (MGP) to impute the irregularly-sampled time series data on the grid points and used those to make predictions via recurrent neural network (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 using 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 by MGP to impute the missing value and adopts RNN as the Q-learning network. Wang et al. (2018) learns a safe treatment scheduling policy that both matches existing physician policy and maximizes long-term reward using actor-critic framework. 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. Moreover, they either consider the effect of action to be independent or allow only a few actions to be scheduled. Our framework is scalable to large number of actions and consider multiple actions jointly, which is a more realistic setting in the clinical practice.

Beside treatment scheduling, Cheng et al. (2019) also aims to learn a measurement policy by RL. They use fitted Q-iteration to schedule 4 different lab tests relevant to diagnosis of sepsis and acute renal failure in the ICU setting. Our work differs in three main ways. First, they treat the scheduling of each measurement independent, making it unsuitable in ICU since the lab measurement values are highly correlated and sampling policy should be considered jointly across all measurements. We show that this independent design underperforms substantially than our sequential design policy in section 4.2, and could be the reason for their sometimes subpar performance against random policy. Second, their reward is different from ours: they design a multi-objective reward such as SOFA score or missingness, while we represent the informativeness of a new measurement using a trained classifier as our reward and use linear combination to combine multiple objectives. Third, their MDP state formulation doesn’t explicitly capture historical information of the patient. Instead, our work summarizes the historical information trends explicitly using LSTMs, and shares this representation both for risk scoring and action choosing.

2.3. Active feature acquisition

Several works (Contardo et al., 2016; He et al., 2016; Shim et al., 2017) study the problem of selecting a subset of features to achieve the maximum prediction performance for a non-time-series classifier. We tackle time-series feature acquisition problem where historical information matters. This is especially true in a healthcare 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 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. 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 informativeness of a new measurement - 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 a RL model. See Figure 1 for an overview.
We list all the hyperparameters in appendix B. To handle the sparse time-series data in LSTM, we use value stream and advantage stream (Wang et al., 2015), i.e. there is no known true scheduling order. Given

\[ k \]

\[ \eta \]

\[ \xi \]

For the first part, we train a multi-layer LSTM classifier testing (Hochreiter & Schmidhuber, 1997) to forecast events of interest using various features. We then frame measurement scheduling question as a sequential action acquisition problem by RL. We train a dueling deep Q-learning network (DQN) to schedule measurements that maximizes the classifier's predictive probability 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 fill in the missing measurement values. We concatenate the imputed measurement values with missingness indicators and the static demographics for each timepoint \( t \) and individual \( i \). To learn the classifier \( \mathcal{I} \), we minimize cross-entropy loss between RNN’s prediction and true label by backpropagation (Figure 1, Forecasting Model). We list all the hyperparameters in appendix B.

### 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.

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

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

In our clinical data, lots of measurements have the exact same time for convenience, i.e. there is no known true scheduling order. Given \( K \) possible measurements, at any given time 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, Right). Specifically, we include a new action \( \Omega \) to represent stopping making any more action. Then at each time point, the agent chooses the action with maximum \( Q \)-value one at a time until the agent selects action \( \Omega \) (Algorithm 1).

#### Algorithm 1 Running policy

**Input:** LSTM hidden state \( h_t \), policy \( Q \)

**Output:** DQN actions \( A_t \)

1. Initialize actions \( A_t = \emptyset \)
2. while \( \Omega \notin A_t \) do
   1. \( s_t \leftarrow [h_t, A_t] \)
   2. \( a \leftarrow \arg \max_{a' \notin A_t} Q(s_t, a') \)
   3. Add \( a \) into \( A_t \)
3. end while

**Action** We add a new stop-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 taken at this timepoint, otherwise \( a_{v,k} = 0 \).

**Reward** We define the reward function as a linear combination of the information gain \( g_x \) and measurement cost \( c \), i.e. \( r(s_t, a_v) = g_x(s_t, a_v) - \lambda * c(a_v) \), where \( v \) represents the step in the MDP (to differentiate between timepoint \( t \)).

---

**Figure 1.** Our System Pipeline. (Left) Given a medical dataset with time-invariant and variant features, we train a forecasting model LSTM which produces an event (e.g. mortality) probability. Then we train a dueling deep Q network to maximize the event probability and minimize number of measurements. Its input \( s_t \) is the concatenation of LSTM hidden state \( h_t \) (summarizing the past information) and the one-hot encoded measurements \( m_v \) already made at this time. (Right) Policy illustration. The agent sequentially decides whether to take another measurement (M1, M2, M3) or stops making more measurements (\( \Omega \)) at the current timepoint.
To encourage the predictive performance of the classifier $\mathcal{I}$, we define the information gain $g(s_v, a_v)$ as the probability change of the classifier $\mathcal{I}$, conditioned on the label, i.e.

$$g_\mathcal{I}(\Delta p) = \begin{cases} 
\Delta p, & \text{if label} = 1 \\
-\Delta p, & \text{otherwise}
\end{cases} \quad (2)$$

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.

$$c(a_v) = \begin{cases} 
1, & \text{if } a_v \neq \Omega \\
0, & \text{otherwise}
\end{cases} \quad (3)$$

**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 $[h_t, m_t', a, r, \gamma]$ in a sequential manner (Algorithm 2). 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 thus treat every order equally likely. We generate training experience based on several 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 $(M1, M2, M3, \Omega)$, $(M2, M1, M3, \Omega)$, or $(M3, M2, M1, \Omega)$ etc. To avoid the total reward received change across different random orders, we do not decay the reward ($\gamma = 1$) in these experiences. Under our linear additive reward design (eq. 2), the random order produces the same cumulative reward no matter which order is used. Also, we do not update the hidden state $h$ for these measurement experiences within the same time $t$ since we do not know the measurement until $t + 1$.

We optimize the RL agent by minimizing the Bellman-equation square error (Algorithm 3). Note that when calculating the $Q_{\text{target}}$, the best action considered can not be in the action set $m_t'$ already performed in the current time, i.e.

$$Q_{\text{target}}(s, a, s') = r(s, a) + \gamma \max_{a' \in m_t'} Q(s', a')$$

All the training hyperparameters are listed in Table 4.

### 3.3. Off-Policy Policy Evaluation (OPPE)

OPPE is currently the only way to evaluate RL performance on retrospective data, and is crucial to report OPPE to make it possible for a future online evaluation in a healthcare setting. 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 method since it would require an exact match with physician actions under our deterministic policy.

---

**Algorithm 2** Generate experience for a patient at time $t$

**Input:** Pretrained LSTM model $\mathcal{I}$, current observation $y_t = \{y_{t,1}, \ldots, y_{t,K}\}$, patient’s history observations $q_{t-1} = \{y_{t-1,1}, \ldots, y_{t-1,1}\}$, decay factor $\gamma$, total number of measurement $K$, action cost scale factor $\lambda$.

$h_{\mathcal{I}}(q, x), p_{\mathcal{I}}(q, x)$: last hidden state and the probability of $\mathcal{I}$ with patient’s observations $q$ and prediction time $x$.

**Output:** All training experiences tuple $E$

$$E = \emptyset$$

Store time-passing experience from from $t - 1$ to $t$

$$[h = h_{\mathcal{I}}(q_{t-1}, t-1), m = y_{t-1}, h' = h_{\mathcal{I}}(q_{t-1}, t), m' = \{y_{t,1}, \ldots, y_{t,v}\}, a = \text{index}(y_{t,v}), r = (p_{\mathcal{I}}(q_{t-1}) - p_{\mathcal{I}}(q_{t-1}, t-1)), \gamma = \gamma] \in E$$

Randomly shuffle $y_t$

for $v = 1$ to $K$ do

Store measurement experience $[h = h_{\mathcal{I}}(q_{t-1}, t), m = \{y_{t,1}, \ldots, y_{t,v}\}, h' = h_{\mathcal{I}}(q_{t-1}, t), m' = \{y_{t,1}, \ldots, y_{t,v}\}, a = \text{index}(y_{t,v}), r = (p_{\mathcal{I}}(q_{t-1}) - p_{\mathcal{I}}(q_{t-1}, t), t) - \lambda \cdot c(a)], \gamma = 1] \in E$

end for

To encourage the predictive performance of the classifier $\mathcal{I}$, we define the information gain $g_\mathcal{I}(s_v, a_v)$ as the probability change of the classifier $\mathcal{I}$, conditioned on the label, i.e.

$$g_\mathcal{I}(\Delta p) = \begin{cases} 
\Delta p, & \text{if label} = 1 \\
-\Delta p, & \text{otherwise}
\end{cases} \quad (2)$$

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.

$$c(a_v) = \begin{cases} 
1, & \text{if } a_v \neq \Omega \\
0, & \text{otherwise}
\end{cases} \quad (3)$$

**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 2). 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 thus treat every order equally likely. We generate training experience based on several 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 $(M1, M2, M3, \Omega)$, $(M2, M1, M3, \Omega)$, or $(M3, M2, M1, \Omega)$ etc. To avoid the total reward received change across different random orders, we do not decay the reward ($\gamma = 1$) in these experiences. Under our linear additive reward design (eq. 2), the random order produces the same cumulative reward no matter which order is used. Also, we do not update the hidden state $h$ for these measurement experiences within the same time $t$ since we do not know the measurement until $t + 1$.

We optimize the RL agent by minimizing the Bellman-equation square error (Algorithm 3). Note that when calculating the $Q_{\text{target}}$, the best action considered can not be in the action set $m_t'$ already performed in the current time, i.e.

$$Q_{\text{target}}(s, a, s') = r(s, a) + \gamma \max_{a' \in m_t'} Q(s', a')$$

All the training hyperparameters are listed in Table 4.

---

**Algorithm 3** Training sequential DQN

**Input:** Pretrained LSTM model $\mathcal{I}$, patient’s database $D = \{q^1, \ldots, q^N\}$, patient’s trajectory length $T^i$.

**Output:** DQN model $Q_{\theta}$

$R \leftarrow \emptyset$ // Initialize prioritized experience replay buffer $R$

for $q^i$ in $D$ do

while $L$ is not converged do

$$E \sim R$$

$$h, m, h', m', a, r, \gamma \leftarrow E$$

$$s = [h, m], s' = [h', m']$$

$$Q_{\text{target}}(s, a, s') = r(s, a) + \gamma \max_{a' \in m_t'} Q_{\theta}(s', a')$$

minimize $L = (Q_{\theta}(s, a) - Q_{\text{target}}(s, a, s'))^2$

Update priority of $E$ in $R$ using $L$

end while
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 information gain probability changes of model $I$. 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 to 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: $M = \{0,1 : 0 = Healthy, 1 = Critical\}$. 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 $S_t$. 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 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 $\{\{y_{t,k}\}_{k=1}^{10}\}_{t'=4}$, 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 time points with $p(a_{t+5} = 1) = softmax(\sum_{t'=4}^{t'} \sum_{k=1}^{10} y_{t,k} \cdot f_k \cdot \eta^{t'}).$ 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 $c(a_{v}) = 1$. The RL agent takes $\{\{y_{t,k}\}_{k=1}^{10}\}_{t'=4}^{t'+5}$ as input. We set reward discount factor $\gamma = 0.999$ in this task.

We simulated a dataset of 5,000 patient trajectories with $T \in [25,50]$ 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 ($Random$). Another class of baseline policies is randomly selecting $x$ measurements ($Random$).
As we vary $\lambda$, 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, showing the benefits of dynamically measure patients conditioned on the patient state.

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 the past 24 hours of observations. The observations include 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. By combining the classifier and RL, we are able to learn clinically relevant policies from off-line data and show our policies perform better than clinician’s policy using off-policy policy evaluation.

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

<table>
<thead>
<tr>
<th></th>
<th>AUC</th>
<th>AUPR</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR</td>
<td>0.931</td>
<td>0.752</td>
</tr>
<tr>
<td>RF</td>
<td>0.935</td>
<td>0.756</td>
</tr>
<tr>
<td>RNN</td>
<td>0.950</td>
<td>0.803</td>
</tr>
</tbody>
</table>

**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 earlier rather than later. We vary our DQN architecture, random seed and action cost coefficient $\lambda$ (range listed in Table 4) to train 150 different policies by random search, and select the best performing policies based on validation set. We list all the hyperparameters in Appendix Table 4.

We use regression based OPPE 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 (denoted sequential DQN in Figure 5) reduce action costs by 31% under the same information gain, or increase the information gain 3 times relative to the lowest information gain with the same action costs. In addition, we scale up Cheng et al. (2019) to our setting by considering the reward independently for each measurement and model the Q-values using a multi-output neural network (denoted independent DQN in Figure 4). This approach only increases information gain by 30%, decreasing the cost by 12%.
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.

In Figure 6, we compare our agent’s and physician’s 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 sequential DQN. Sequential DQN makes decision based on the history of the physician’s sampled measurements and it is adaptive to patients’ history, recommending probing patients when no recent measurement exists and vice versa.

5. Discussion and Future Work

In this work, we propose a scalable method for measurement scheduling using data-driven approach. We show that our scheduling policy achieves better predictive power with lower measurement costs in both simulations and on MIMIC3. We also examine our policies qualitatively and show that our policies sample clinically-relevant measurements and act based on the patient’s measurement history.

In this study, we assume sampling lab tests at the current time point provide us with information by the next time point. In reality, some lab tests can take hours to get the information 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. We also assume all medical measurements are needed for scheduling and we only consider the simplest set-
Dynamic Measurement Scheduling for Event Forecasting Using Deep RL

Figure 7. Action frequency of physician and RL agent policies. Each panel has 39 cells aligned on three rows with each row of 13 actions (e.g. 1st row for action 0-12 etc.), with color representing the sampling frequency per action. The measurement corresponding to the action index is shown on the right. Top left is the action frequency of physician policy in our 24h mortality forecasting task. The rest are the action frequencies of Sequential DQN policies with increasing action costs, from left to right, from top to bottom. Physician’s most selected measurements in the last 24h are highlighted in bold. The RL agent policy with comparable action cost as physician policy but higher information gains is highlighted in the black box.

We believe that RL has a great role to play in helping diagnose and ultimately prevent critical events in the healthcare.
ACKNOWLEDGMENTS

We thank Amir-massoud Farahmand and Marzyeh Ghassemi for their helpful discussions. AG and CC are supported by the Early Researcher Award from the Ministry of Research and Innovation.

References


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


