

Predicting healthcare trajectories from medical records: A deep learning approach

Trang Pham*, Truyen Tran, Dinh Phung, Svetha Venkatesh

Center for Pattern Recognition and Data Analytics, Deakin University Geelong, Australia

**Corresponding author. E-mail address: phtra@deakin.edu.au*

Abstract

Personalized predictive medicine necessitates the modeling of patient illness and care processes, which inherently have long-term temporal dependencies. Healthcare observations, stored in electronic medical records are episodic and irregular in time. We introduce DeepCare, an *end-to-end* deep dynamic neural network that reads medical records, stores previous illness history, infers current illness states and predicts future medical outcomes. At the data level, DeepCare represents care episodes as vectors and models patient health state trajectories by the memory of historical records. Built on Long Short-Term Memory (LSTM), DeepCare introduces methods to handle irregularly timed events by moderating the forgetting and consolidation of memory. DeepCare also explicitly models medical interventions that change the course of illness and shape future medical risk. Moving up to the health state level, historical and present health states are then aggregated through multiscale temporal pooling, before passing through a neural network that estimates future outcomes. We demonstrate the efficacy of DeepCare for disease progression modeling, intervention recommendation, and future risk prediction. On two important cohorts with heavy social and economic burden – diabetes and mental health – the results show improved prediction accuracy.

Keywords: Electronic medical records, predictive medicine, Long-Short Term Memory, irregular timing, healthcare processes.

1. Introduction

When a patient is admitted to a hospital, there are two commonly asked questions: “what is happening?” and “what happens next?” The first question is about illness diagnosis, the second is about predicting future medical risk [43]. Whilst there are a wide
5 array of diagnostic tools to answer the first question, fewer technologies address the second [41]. Traditionally, the prognostic question may be answered by experienced clinicians who have seen many patients or by clinical prediction models with well-defined risk factors.

But both methods are expensive and restricted in availability. Modern electronic medical records (EMRs) promise a fast and cheap alternative. An EMR contains the history of hospital encounters, diagnoses, interventions, lab tests and clinical narratives. The wide
10 adoption of EMRs has led to recent research to build predictive models from this rich data source [26, 45, 48, 49].

Answering prognostic inquiries necessitates modeling patient-level temporal health-care processes. Effective modeling must address four open challenges: (i) *Long-term dependencies in healthcare*: the future illness and care may depend critically on historical
15 illness and interventions. For example, the onset of diabetes in middle age remains a risk factor for a person’s remaining life; cancers may recur after years; and a previous surgery may prevent certain future interventions. (ii) *Representation of admission information*: an admission episode consists of a variable-size discrete set containing diagnoses and interventions. (iii) *Episodic recording and irregular timing*: medical records vary greatly in
20 length, are inherently episodic in nature and irregular in time [47]. The data is episodic because it is only recorded when the patient visits the hospital and undergoes an episode of care. The episode is often tightly packed in a short period, typically ranging from a day to two weeks. The timing of arrivals is largely random. (iv) *Confounding interactions between disease progression and interventions*.
25

We address the four challenges to construct a predictive system that is both *end-to-end* and *generic* so that it can be deployed on different hospital implementations of EMRs. An end-to-end system requires minimal or no feature engineering, reads medical records, infers present illness states and predicts future outcomes.

Existing methods are poor in handling such complexity. They inadequately model
30 variable length [45] and ignore the long-term dependencies [24, 31, 51]. Temporal models based on the Markovian assumption are limited to model temporal irregularity and have no memory, and thus they may completely forget previous major illness given an irrelevant episode [1]. Deep learning, which has recently revolutionized cognitive fields
35 such as speech recognition, vision and computational linguistics, holds a great potential in constructing end-to-end systems [27]. However, little work has been done using deep learning for healthcare [8, 13, 28, 46]. While work in deep learning has been done to address the challenge of long-term dependencies [5, 7, 29], the three other challenges remain unsolved.

To this end, we introduce DeepCare, an *end-to-end* deep dynamic memory neural
40 network that addresses the four aforementioned challenges [38]. DeepCare is built on Long Short-Term Memory (LSTM) [15, 21], a recurrent neural network equipped with *memory cells* to store experiences. At each time-step, the LSTM reads an input, updates the memory cell, and returns an output. Memory is maintained through a *forget gate* that
45 moderates the passing of memory from one time step to another, and is updated by seeing

new input at each time step. The output is determined by the memory and moderated by an *output gate*. In DeepCare, the LSTM models the illness trajectory and healthcare processes of a patient encapsulated in a time-stamped sequence of admissions. The inputs to the LSTM are information extracted from admissions. The outputs represent illness states at the time of admission. *Memory maintenance enables capturing of long-term dependencies*, thus addressing the first challenge. In fact, this capacity has made LSTM an ideal model for a variety of sequential domains [15, 17, 44].

Addressing the other three drawbacks, DeepCare introduces C-LSTM as an extension of the standard LSTM unit (Fig. 1). For representing an admission, which is a set of discrete elements in different types such as diagnoses and interventions, the solution is to embed these elements into continuous vector spaces. Vectors of the same type are then pooled into a single vector. Type-specific pooled vectors are then concatenated to represent an admission. In that way, *variable-size admissions are embedded in to continuous distributed vector space*. The admission vectors then serve as input features for the C-LSTM. As the embedding is learned from data, the model does not rely on manual feature engineering.

For *irregular timing*, the *forget gate* is extended to be a function of irregular time gap between consecutive time steps. We introduce two new forgetting mechanisms: monotonic decay and full time-parameterization. The decay mimics natural forgetting when learning a new concept in human. The parameterization accounts for more complex dynamics of different diseases over time. The resulting model is sparse in time and efficient to compute since only observed records are incorporated, regardless of the irregular time spacing. Finally, in DeepCare *the confounding interaction between disease progression and interventions is modeled* as follows: Interventions influence the output gate of current illness states and the forget gate which moderates memory carried into the future. As a result, the illness states (the output) are moderated by past and current interventions.

Once illness states are outputted by the C-LSTM layer, they are aggregated through a new time-decayed multiscale pooling strategy for future projection. This allows further handling of time-modulated memory. Finally at the top layer, pooled illness states are passed through a neural network for future prognosis (See Fig. 1 for a graphical depict of DeepCare). Overall, DeepCare is an end-to-end prediction model that relies on no manual feature engineering, is capable of reading generic medical records, memorizing a long history, inferring current illness states and predicting the future risk.

We demonstrate our DeepCare on answering a part of the question “what happens next?”. In particular, we validate our model on *disease progression*, *intervention recommendation* and *future risk prediction*. Disease progression refers to the next disease occurrence given the medical history. Intervention recommendation is about predicting a subset of treatment procedures for the current diagnoses. Future risk may involve read-

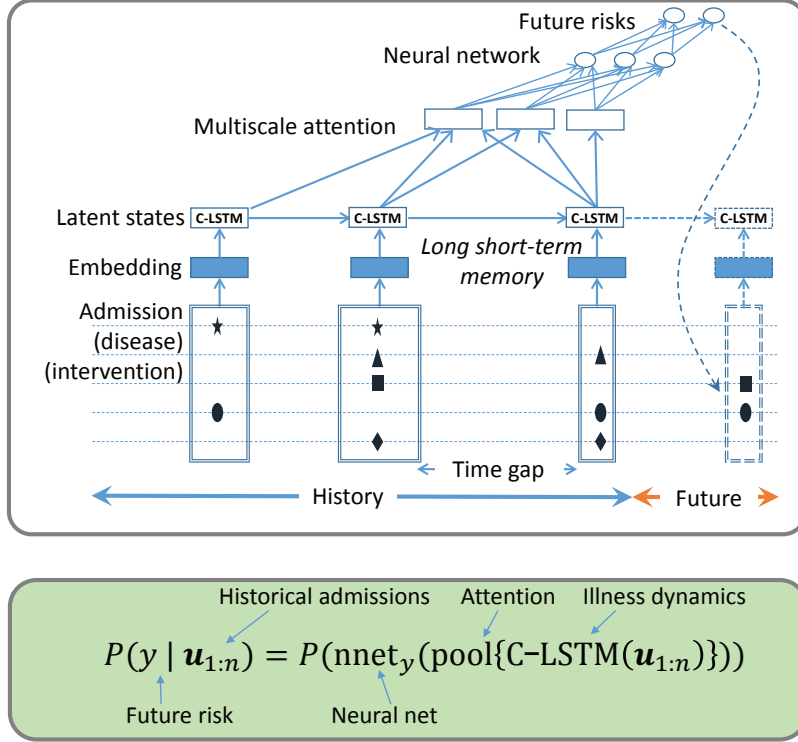


Figure 1: DeepCare architecture. **Top:** The healthcare dynamics are modelled as a sequence of C-LSTM units which model irregular timing and interventions (see Fig. 4 for more detail). The symbols (e.g., stars, circles and triangles) in the bottom rectangles are diagnosis and interventions codes (See Sec. 2.2). **Bottom:** Predictive computation summarized in an equation.

mission or mortality within a predefined period after discharge. Our experiments are demonstrated on two datasets of very different nature – diabetes (a well-defined chronic condition) and mental health (a diverse mixture of many acute and chronic conditions). The cohorts were collected from a large regional hospital in the period of 2002 to 2013. We show that DeepCare outperforms state-of-the-art classification methods.

Initial implementation of our framework has been conducted and preliminarily reported in [38]. Here we provide a complete account of the model and more comprehensive results on two chronic cohorts (diabetes and mental health). To summarize, through introducing DeepCare, we make four modeling contributions: (i) handling long-term dependencies in healthcare; (ii) introducing a novel representation of variable-size admission as fixed-size continuous vectors; (iii) modeling episodic recording and irregular timing; and (iv) capturing confounding interactions between disease and interventions. We also contribute to the healthcare analytic practice by demonstrating the effectiveness of DeepCare on disease progression, intervention recommendation and medical risk prediction.

The rest of this paper is organized as follows. Section 2 presents preliminaries for DeepCare model: LSTM and the coding of EMRs. DeepCare is described in Section 3 while the experiments and results are reported in Section 4. Finally, Section 5 discusses further and concludes the paper.

2. Preliminaries

2.1. Long Short-Term Memory

A Recurrent Neural Network (RNN) is a neural network repeated over time. In particular, an RNN allows self-loop connections and shared parameters across different time steps. While a feedforward neural network maps an input vector into an output vector, an RNN maps a sequence into a sequence (see Fig. 2 for a graphical illustration).

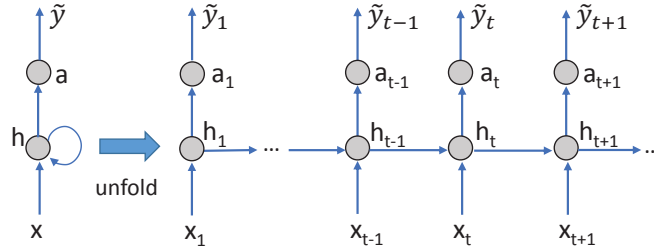


Figure 2: **(Left)** A typical Recurrent Neural Network that recurrently reads new input x , re-computes the hidden state h and returns the output \tilde{y} . **(Right)** an RNN unfolded in time. Each RNN unit at time step t reads input x_t and previous hidden state h_{t-1} , generates output a_t and predicts the label \tilde{y}_t .

An RNN unit has three connections: a recurrent connection from the previous hidden state to the current hidden state ($h_{t-1} \rightarrow h_t$), an input-to-hidden-state connection ($x_t \rightarrow h_t$) and a hidden-state-to-output connection ($h_t \rightarrow a_t$). At time step t , the model reads the input $x_t \in \mathbb{R}^M$ and the previous hidden state $h_{t-1} \in \mathbb{R}^K$ and compute the hidden state h_t , where M and K are vector dimensions of input and hidden state at every step. Thus h_t summarizes information from all previous inputs x_1, x_2, \dots, x_t . The output $a_t \in \mathbb{R}^{n_c}$ is generated by a transformation function of h_t , where n_c is the number of classes in the classification tasks. Many experiments have shown that learning long RNNs is difficult due to vanishing or exploding gradients [4, 36].

Long Short-Term Memory (LSTM)

LSTM is a RNN that effectively solves the vanishing gradient problem [21]. Central to an LSTM is a linear self-loop memory cell that allows gradients to flow through long sequences. The memory cell is gated to moderate the information flow to or from the cell. LSTMs have been successful in many applications, such as machine translation [44], handwriting recognition [16] and speech recognition [18].

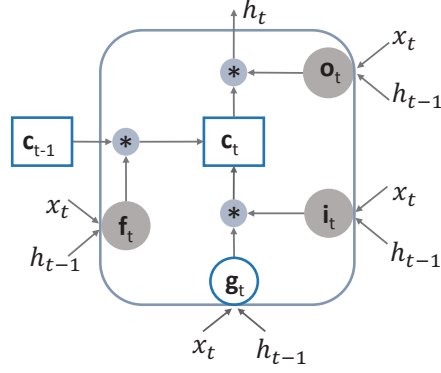


Figure 3: An LSTM unit that reads input \mathbf{x}_t and previous output state \mathbf{h}_{t-1} and produces current output state \mathbf{h}_t . An unit has a memory cell \mathbf{c}_t , an input gate \mathbf{i}_t , an output gate \mathbf{o}_t and a forget gate \mathbf{f}_t .

Fig. 3 describes an LSTM unit. Instead of a simple RNN unit, an LSTM unit has a memory cell that has state $\mathbf{c}_t \in \mathbb{R}^K$ at time t . The information flowing through the memory cell is controlled by three gates: an *input* gate, a *forget* gate and an *output* gate. The input gate $\mathbf{i}_t \in \mathbb{R}^K$ controls the input flowing into the cell, the forget gate $\mathbf{f}_t \in \mathbb{R}^K$ controls the forgetting of the memory cell, and the output gate $\mathbf{o}_t \in \mathbb{R}^K$ moderates the output flowing from the memory cell. Before proceeding with technical details, we denote the element-wise sigmoid function of a vector by σ and the element-wise product of two vectors by $*$.

The three gates are all sigmoid units that set every element of the gates to a value between 0 and 1:

$$\mathbf{i}_t = \sigma(W_i \mathbf{x}_t + U_i \mathbf{h}_{t-1} + \mathbf{b}_i) \quad (1)$$

$$\mathbf{f}_t = \sigma(W_f \mathbf{x}_t + U_f \mathbf{h}_{t-1} + \mathbf{b}_f) \quad (2)$$

$$\mathbf{o}_t = \sigma(W_o \mathbf{x}_t + U_o \mathbf{h}_{t-1} + \mathbf{b}_o) \quad (3)$$

where $W_{\{i,f,o\}}$, $U_{\{i,f,o\}}$, $\mathbf{b}_{\{i,f,o\}}$ are parameters. The gates control the amount of information passing through: full flow when the gate value is 1, to complete blockage when the value is 0.

At each time step t , the input features are first computed by passing input $\mathbf{x}_t \in \mathbb{R}^M$ and the previous hidden state $\mathbf{h}_{t-1} \in \mathbb{R}^K$ through a squashing tanh function:

$$\mathbf{g}_t = \tanh(W_c \mathbf{x}_t + U_c \mathbf{h}_{t-1} + \mathbf{b}_c) \quad (4)$$

The memory cell is updated through partial forgetting of the previous memory cell and

the moderated input features as follows:

$$\mathbf{c}_t = \mathbf{f}_t * \mathbf{c}_{t-1} + \mathbf{i}_t * \mathbf{g}_t \quad (5)$$

140 The memory cell sequence is additive, and thus the gradient is also updated in a linear fashion through the chain rule. This effectively prevents the gradient from vanishing or exploding. The memory cell plays a crucial role in memorizing past experiences through the *learnable* forgetting gates \mathbf{f}_t . On the contrary, $\mathbf{f}_t \rightarrow \mathbf{1}$, all the past memory is preserved, and new memory keeps updated with new inputs. If $\mathbf{f}_t \rightarrow \mathbf{0}$, only the new experience is updated and the system becomes memoryless.

145 Finally, a hidden output state \mathbf{h}_t is computed based on the memory \mathbf{c}_t , gated by the output gate \mathbf{o}_t as follows:

$$\mathbf{h}_t = \mathbf{o}_t * \tanh(\mathbf{c}_t) \quad (6)$$

Note that since the system dynamics are deterministic, \mathbf{h}_t is a function of all previous inputs: $\mathbf{h}_t = \text{LSTM}(\mathbf{x}_{1:t})$. The output states are then used to generate outputs.

150 2.2. EMR coding

An electronic medical record (EMR) is a digital version of a patient's health information. A wide range of information can be stored in EMRs, such as detailed records of symptoms, data from monitoring devices and clinician's observations [37]. A typical EMR contains information about a sequence of admissions for a patient. There are two types of admission methods: planned (routine) and unplanned (emergency). Unplanned admission refers to transfer from the emergency department. EMRs typically store admitted time, discharge time, lab tests, diagnoses, procedures, medications and clinical narratives. Diagnoses, procedures and medications are typically coded in standardized formats. Diagnoses are represented using WHO's ICD-10 (International Classification of Diseases) coding schemes¹. For example, in ICD-10, E10 encodes Type 1 diabetes mellitus, E11 encodes Type 2 diabetes mellitus while F32 indicates depressive episode. The procedures are coded in CPT (Current Procedural Terminology) or ICHI (International Classification of Health Interventions) schemes². Medication names can be mapped into the ATC (Anatomical Therapeutic Chemical) scheme³.

¹<http://apps.who.int/classifications/icd10/browse/2016/en>

²<http://www.who.int/classifications/ichi/en/>

³http://www.whocc.no/atc_ddd_index/

165 3. DeepCare: A *Deep* learning framework for *Care* episodes

In this section, we present our main contribution: DeepCare for modeling illness trajectories and predicting future outcomes. DeepCare is built upon LSTM to exploit its ability to model long-term dependencies in sequences. We extend the LSTM unit to C-LSTM unit to address the three major challenges: (i) *variable-size discrete inputs*, (ii) 170 *confounding interactions between disease progression and intervention*, and (iii) *irregular timing*.

3.1. Model overview

DeepCare (see Fig. 1) is a deep dynamic neural network that has three main layers. The bottom layer is built on C-LSTM whose memory cells are modified to handle irregular 175 timing and interventions, the capacity not seen in standard LSTM units (see Fig. 4). More specifically, the input is a sequence of admissions. Each admission t contains a set of diagnosis codes (which is then formulated as a feature vector $\mathbf{x}_t \in \mathbb{R}^M$), a set of intervention codes (which is further formulated as a feature vector $\mathbf{p}_t \in \mathbb{R}^M$, where M is the vector dimension of \mathbf{x}_t and \mathbf{p}_t), the admission method $m_t \in \mathbb{R}$ and the elapsed time Δt between the current admission and its previous one. Denote by $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_n$ 180 the input sequence, where $\mathbf{u}_t = [\mathbf{x}_t, \mathbf{p}_t, m_t, \Delta t]$, the C-LSTM computes the corresponding sequence of distributed illness states $\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_n$, where $\mathbf{h}_t \in \mathbb{R}^K$ and K is the vector dimension. (see Fig. 4b). The middle layer aggregates illness states through multiscale weighted pooling $\bar{\mathbf{h}} = \text{pool} \{\mathbf{h}_1, \mathbf{h}_1, \dots, \mathbf{h}_n\}$, where $\bar{\mathbf{h}} \in \mathbb{R}^{sK}$ for s scales.

185 The top layer is a neural network (nnet_y) that takes pooled states and other statistics to estimate the final outcome probability, as

$$P(y | \mathbf{u}_{1:n}) = P(\text{nnet}_y(\bar{\mathbf{h}}))$$

The probability $P(y | \mathbf{u}_{1:n})$ depends on the nature of outputs and the choice of statistical structure. For example, for binary outcomes, $P(y = 1 | \mathbf{u}_{1:n})$ is a logistic function; for multiclass outcomes, $P(y | \mathbf{u}_{1:n})$ is a softmax function; and for continuous outcomes, 190 $P(y | \mathbf{u}_{1:n})$ is a Gaussian. In what follows, we describe the first two layers in greater detail.

3.2. Representing variable-size admissions

An admission contains multiple diagnoses and interventions. Interventions include procedures and medications. Diagnoses, procedures and medications are coded using 195 coding schemes which are described in Sec. 2.2. Our approach is to embed admissions into vectors (See Fig. 4a). An admission is a variable-size set of codes (diagnoses and interventions). Let \mathcal{D} be the set of diagnosis codes and \mathcal{I} be the set of intervention

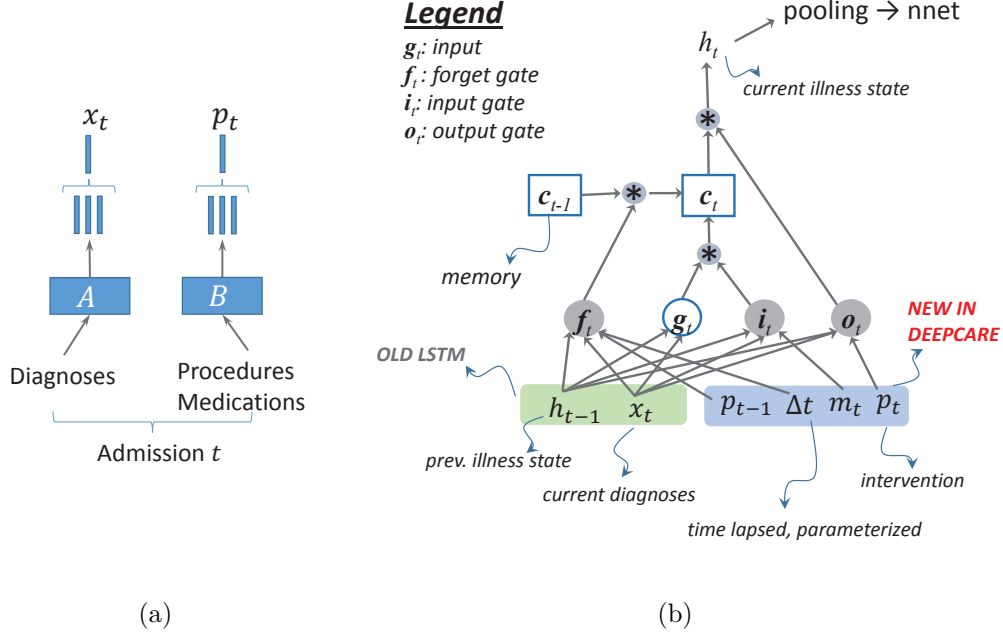


Figure 4: (a) Code embedding. (b) C-LSTM (Care-LSTM) unit as a carrier of illness history. Compared to the original LSTM unit (Fig. 3), the new C-LSTM unit models times, admission methods, diagnoses and intervention.

codes. The two sets are indexed from 1 to $|\mathcal{D}|$ and from 1 to $|\mathcal{I}|$, respectively. Denote the diagnosis embedding matrix by $A \in \mathbb{R}^{M \times |\mathcal{D}|}$ and the intervention embedding matrix by $B \in \mathbb{R}^{M \times |\mathcal{I}|}$. Let A^j be the j^{th} column and A_i^j be the element at the j^{th} column and the i^{th} row of the matrix A . Each admission t contains h diagnoses: $d_1, d_2, \dots, d_h \in \{1, 2, \dots, |\mathcal{D}|\}$ and k interventions: $s_1, s_2, \dots, s_k \in \{1, 2, \dots, |\mathcal{I}|\}$. Codes are first embedded into vectors. The embedded vectors for diagnosis and intervention codes are A^{d_1}, \dots, A^{d_k} and B^{s_1}, \dots, B^{s_k} . We then pool all the present diagnosis vectors to derive $\mathbf{x}_t \in \mathbb{R}^M$. Likewise, we derive a pooled intervention vector $\mathbf{p}_t \in \mathbb{R}^M$. Finally, an admission embedding is a $2M$ -dim vector $[\mathbf{x}_t, \mathbf{p}_t]$. The two embedding matrices are first randomly initialized and then learned through training the prediction tasks.

3.2.1. Pooling

Let x_t^i be the i^{th} element of the vector \mathbf{x}_t and p_t^i be the i^{th} element of the vector \mathbf{p}_t . The admission is pooled by max, sum or mean pooling as:

- *Max pooling admission (max adm.)*. The pooling is element-wise as follows:

$$x_t^i = \max \left(A_i^{d_1}, A_i^{d_2}, \dots, A_i^{d_h} \right)$$

$$\mathbf{p}_t^i = \max(B_i^{s_1}, B_i^{s_2}, \dots, B_i^{s_k})$$

for $i = 1, \dots, M$. This is analogous to paying selective attention to the element of highest impact among the diagnoses and the interventions. It also resembles the usual coding practice that one diagnosis is picked as the primary reason for admission.

- *Normalized sum pooling admission (sum adm.)*. A patient with multiple diseases (multiple comorbidities) is more likely to be at risk than those with a single condition. We propose the following normalized sum pooling:

$$\mathbf{x}_t^i = \frac{A_i^{d_1} + A_i^{d_2} + \dots + A_i^{d_h}}{\sqrt{|A_i^{d_1} + A_i^{d_2} + \dots + A_i^{d_h}|}}$$

$$\mathbf{p}_t^i = \frac{B_i^{s_1} + B_i^{s_2} + \dots + B_i^{s_k}}{\sqrt{|B_i^{s_1} + B_i^{s_2} + \dots + B_i^{s_k}|}}$$

for $i = 1, \dots, M$. The normalization reduces the effect of large diagnosis and intervention sets.

- *Mean pooling admission (mean adm.)*. In absence of primary conditions, a mean pooling could be a sensible choice:

$$\mathbf{x}_t = \frac{A^{d_1} + A^{d_2} + \dots + A^{d_h}}{h}$$

$$\mathbf{p}_t = \frac{B^{s_1} + B^{s_2} + \dots + B^{s_k}}{k}$$

3.2.2. Admission as input

Once admission embedding has been derived, the diagnosis component is used as input for the C-LSTM. As interventions are designed to reduce illness, their effect is modeled separately in Sec. 3.3.1. Recall from Sec. 2.2, there are two main types of admission: planned and unplanned. Unplanned admissions refer to transfer from emergency attendances, which typically indicates higher risk. Recall from Eqs. (1, 4) that the input gate \mathbf{i} controls how much new information is updated into memory \mathbf{c} . The gate is modified to reflect the risk level of admission type as follows:

$$\mathbf{i}_t = \frac{1}{m_t} \sigma(W_i \mathbf{x}_t + U_i \mathbf{h}_{t-1} + \mathbf{b}_i) \quad (7)$$

where $m_t = 1$ if the admission method is unplanned, $m_t > 1$ otherwise, and σ is the element-wise sigmoid function of a vector. (See Supplementary Appendix A.5 for more details about the effect of m_t).

3.3. C-LSTM unit

We now describe C-LSTM, which stands for *Care-LSTM*, units. A C-LSTM unit extends the LSTM unit to reflect the properties of healthcare dynamics. In particular, C-LSTM units model the effect of interventions and capture time irregularities. See Fig. 4b
 240 for a graphical illustration.

3.3.1. Modeling effect of interventions

The intervention vector (\mathbf{p}_t) of an admission is modeled as illustrated in Fig. 4b. Since interventions are designed to cure diseases or reduce patient's illness, the output gate, which controls the illness states, is moderated by the *current* intervention as follows:

$$\mathbf{o}_t = \sigma(W_o \mathbf{x}_t + U_o \mathbf{h}_{t-1} + P_o \mathbf{p}_t + \mathbf{b}_o) \quad (8)$$

where P_o is the intervention weight matrix for the output gate and \mathbf{p}_t is intervention at time step t .

Moreover, interventions may have long-term impacts (e.g., curing disease or introducing toxicity). This suggests the illness forgetting is moderated by *previous* intervention

$$\mathbf{f}_t = \sigma(W_f \mathbf{x}_t + U_f \mathbf{h}_{t-1} + P_f \mathbf{p}_{t-1} + \mathbf{b}_f) \quad (9)$$

where \mathbf{p}_{t-1} is intervention embedded vector at time step $t - 1$ and P_f is the intervention weight matrix for the forget gate.

3.3.2. Capturing time irregularity

When a patient's history is modeled by LSTM (Sec. 2.1), the memory cell carries the
 255 illness history. But this memory needs not be constant as illness states change over time. In C-LSTM, we introduce two mechanisms of forgetting the memory by modifying the forget gate \mathbf{f}_t in Eq. 9:

Time decay. There are acute conditions that naturally reduce their effect through time. This suggests a simple decay modeled in the forget gate \mathbf{f}_t :

$$\mathbf{f}_t \leftarrow d(\Delta_{t-1:t}) \mathbf{f}_t \quad (10)$$

260 where $\Delta_{t-1:t}$ is the time passed between step $t - 1$ and step t , and $d(\Delta_{t-1:t}) \in (0, 1]$ is a decay function, i.e., it monotonically decreases in time. We found that the function $d(\Delta_{t-1:t}) = [\log(e + \Delta_{t-1:t})]^{-1}$ works well, where $\Delta_{t-1:t}$ is measured in days and $e \approx 2.718$ is the base of the natural logarithm.

265 *Parametric time.* Time decay may not capture all conditions as some conditions can get worse, and others can be chronic. This suggests a more flexible parametric forgetting:

$$\mathbf{f}_t = \sigma \left(W_f \mathbf{x}_t + U_f \mathbf{h}_{t-1} + Q_f \mathbf{q}_{\Delta_{t-1:t}} + P_f \mathbf{p}_{t-1} + \mathbf{b}_f \right) \quad (11)$$

where $\mathbf{q}_{\Delta_{t-1:t}}$ is a vector derived from the time difference $\Delta_{t-1:t}$, Q_f is the parametric time weight matrix. For example, we may have: $\mathbf{q}_{\Delta_{t-1:t}} = \left(\frac{\Delta_{t-1:t}}{60}, \left(\frac{\Delta_{t-1:t}}{180} \right)^2, \left(\frac{\Delta_{t-1:t}}{365} \right)^3 \right)$ to model the third-degree forgetting dynamics. $\Delta_{t-1:t}$ is measured in days and is divided by 60, 180 and 365 to prevent the vector $\mathbf{q}_{\Delta_{t-1:t}}$ from having large values.

270 3.4. Trajectory prediction

Once the C-LSTM units have been set up, at each time-step, the hidden illness state \mathbf{h}_t is computed. The states are then used to predict the future trajectory. We consider three tasks: (1) next-step disease progression, (2) intervention recommendation, and (3) future risk prognosis. The first two tasks cover short-range prediction (current and next admissions), but the third task looks far into the future of any horizon.

3.4.1. Short-range disease progression

Disease progression refers to occurrence of future diseases in the next time-step. It could be the progression from a stage to another of the same disease, the recurrence of a disease, or the transition to a new disease. The illness state \mathbf{h}_t can be used to predict a diagnosis code d_{t+1} as follows

$$P(d_{t+1} = c \mid \mathbf{u}_{1:t}) = \text{softmax}(\mathbf{w}_c^\top \mathbf{h}_t) \quad (12)$$

where $\text{softmax}(z) = e^z / \sum_{z'} e^{z'}$, \mathbf{w}_c is code-specific parameter.

3.4.2. Short-range intervention recommendation

Intervention recommendation refers to predicting medications and procedures for the current diagnoses. Similar to disease progression, an intervention code s_t at time t can be generated as follows

$$P(s_t = c \mid \mathbf{u}_{1:t}) = \text{softmax}(\mathbf{v}_c^\top \mathbf{h}_t) \quad (13)$$

where \mathbf{v}_c is code-specific parameter.

3.4.3. Long-range prognosis by pooling multiple temporal resolutions

Recall that the C-LSTM has two temporal characteristics:

- Integrating long-range information through gradually forgotten memory. A consequence of forgetting is that recent information affects the current illness state more, and this fits the nature of healthcare processes.
- Representing a complex effect of time lapse between two admissions from parameterization of time in the forget gate.

However, C-LSTM, like its ancestor LSTM, is not explicitly designed to predict the far future. This because the memory is updated at every admission but the global dynamics across multiple admissions are not fully captured. For this reason, we propose to impose a multiscale temporal structure on top of the C-LSTM layer to predict the far future (see Fig. 1). It means to pool historical illness states within multiple time-horizons. This is to reflect the variable rates at which diseases progress.

- For state pooling per time-horizon, the simplest way is to use mean-pooling, where $\bar{\mathbf{h}} = \text{pool}\{\mathbf{h}_{1:n}\} = \frac{1}{n+1} \sum_{t=1}^n \mathbf{h}_t$. However, this does not reflect the recency in history. Here we introduce a simple attention scheme that weighs recent events more: $\bar{\mathbf{h}} = (\sum_{t=t_1}^n r_t \mathbf{h}_t) / \sum_{t=t_1}^n r_t$, where

$$r_t = [m_t + \log(1 + \Delta_{t:n})]^{-1}$$

and $\Delta_{t:n}$ is the elapsed time between the step t and the current step n , measured in months; $m_t = 1$ if emergency admission, $m_t = 2$ if routine admission. The starting time step t_1 is used to control the length of look-back in the pooling, for example, $\Delta_{t_1:n} \leq 12$ for one year look-back.

- For multiple time-horizons, we employ multiple look-backs: 12 months, 24 months, and all available history. Finally, the three pooled illness states are stacked into a vector: $\bar{\mathbf{h}} = [\bar{\mathbf{h}}_{12}, \bar{\mathbf{h}}_{24}, \bar{\mathbf{h}}_{all}]$ which is then fed to a neural network for inferring about the future.

Once all the illness states are pooled and stacked into vector $\bar{\mathbf{h}}$, $\bar{\mathbf{h}}$ is then fed to a neural network to predict the future outcome y . The design of the neural network is flexible with any depth as desirable with or without parameter tying between layers (see for example, recent work in [39]).

In this paper, we use a simple neural network with one hidden layer, as follows:

$$\mathbf{a}_h = \sigma(U_h \bar{\mathbf{h}} + \mathbf{b}_h) \quad (14)$$

$$\mathbf{z}_y = U_y \mathbf{a}_h + \mathbf{b}_y \quad (15)$$

$$P(y | \mathbf{u}_{1:n}) = f_{prob}(\mathbf{z}_y) \quad (16)$$

The function $f_{prob}(\mathbf{z}_y)$ depends on the nature of the future outcome. For example, in the case of binary classification, $f_{prob}(\mathbf{z}_y)$ is a logistic regression. Although not pursued here, this can be easily extended to survival analysis setting, where $f_{prob}(\mathbf{z}_y)$ is partial-likelihood.

In summary, computation steps in DeepCare can be summarized as follows:

$$P(y | \mathbf{u}_{1:n}) = P(\text{nnet}_y(\text{pool}\{\text{C}\sim\text{LSTM}(\mathbf{u}_{1:n})\})) \quad (17)$$

where $\mathbf{u}_{1:n}$ is the input sequence of admission observations, y is the outcome of interest (e.g., readmission), nnet_y denotes estimate by the neural network with respect to outcome y , and P is probabilistic model of outcomes.

3.5. Model training

Recall that there are three prediction tasks – two short-range (intervention recommendation and disease progression) and one long-range prognosis. As the the short-range tasks are indeed special detailed cases of the long-range task, the models learned from the short-range can be reused in the long-range. This is also known as *transfer learning*. In particular, the short-range models will serve as a pre-training step for the long-range task. See Appendix A.2 for more detail on pre-training.

- **For short-range tasks**, models are trained by minimizing the log-loss $L = -\sum_t \log P(y_t | \mathbf{u}_{1:t})$, where y_t is either intervention code s_t in Eq. (13) or disease code d_{t+1} in Eq. (12).
- **For the long-range task**, the loss is $L = -\log P(y | \mathbf{u}_{1:n})$ where $P(y | \mathbf{u}_{1:n})$ is given in Eq. (16)

Despite having a complex structure, DeepCare’s loss functions are fully differentiable, and thus can be minimized using standard back-propagation and supported by current programming frameworks with automatic differentiation facilities. The learning complexity is linear with the number of parameters (see Appendix A.1 for model complexity analysis).

Alg. 1 is an overview of our DeepCare forward pass. In actual implementation, we also make use of recent techniques such as dropouts [42] (see Appendix A.3 for further detail).

4. Case studies on chronic diseases

In this section we report case studies for two chronic cohorts: mental health and diabetes. For each cohort we modeled disease progression, intervention recommendation and future risk prediction. These diseases differ in causes and progression. Further details

Algorithm 1 DeepCare forward pass

- 1: **Input:** EMRs as sequences of sets of diagnosis, intervention codes, admission type and time lapse.
 - 2: **for** each step t
 - * $[\mathbf{x}_t, \mathbf{p}_t] = \text{embedding}(d_1, \dots, d_h, s_1, \dots, s_k)$ (Sec. 3.2.1)
 - * Compute 3 gates: \mathbf{i}_t (Eq. 7), \mathbf{o}_t (Eq. 8), \mathbf{f}_t (Eq. 10 or Eq. 11)
 - * Compute \mathbf{c}_t (Eq. 5) and \mathbf{h}_t (Eq. 6)**endfor**
 - 3: **if** the task is *Disease progression*
 - * Compute the predictive probability using Eq. (12)
 - * Compute the log-loss.**endif**
 - 4: **if** the task is *Intervention recommendation*
 - * Compute the predictive probability using Eq. (13)
 - * Compute the log-loss.**endif**
 - 5: **if** the task is *Future risk prediction*
 - * Compute $\bar{\mathbf{h}}$ (Sec. 3.4.3)
 - * Compute $P(y | \mathbf{u}_{0:n})$ (Eqs. 14, 15, 16)
 - * Compute the log-loss.**endif**
-

of DeepCare implementation are given in Appendix A.4. Code for the experiments can be found in GitHub ⁴

350 4.1. Data

Data for both cohorts were collected for 12 years (2002-2013) from a large regional Australian hospital. Diseases are coded using ICD-10 (See Sec 2.2 for a brief description). We preprocessed the datasets by removing (i) admissions with incomplete patient information; and (ii) patients with less than 2 admissions. The vocabulary is defined
355 as the set of diagnosis, procedure and medication codes. In diabetes cohort, there are 7,153 diagnosis codes and 1,126 intervention codes while in mental health cohort, there are 8,127 diagnosis codes and 1,351 intervention codes. The vocabularies in both datasets are large that may lead to overfitting when training the model. To reduce the vocabulary, we collapsed diagnoses that share the first 2 characters into one diagnosis. For example,
360 E10.1 would be collapsed into E1. Likewise, the first digits in the procedure block were used.

The diabetes cohort contained more than 12,000 patients (55.5% males, median age 73). Data statistics are summarized in Fig. 5. After preprocessing, the dataset contained

⁴<https://github.com/trangptm/DeepCare>

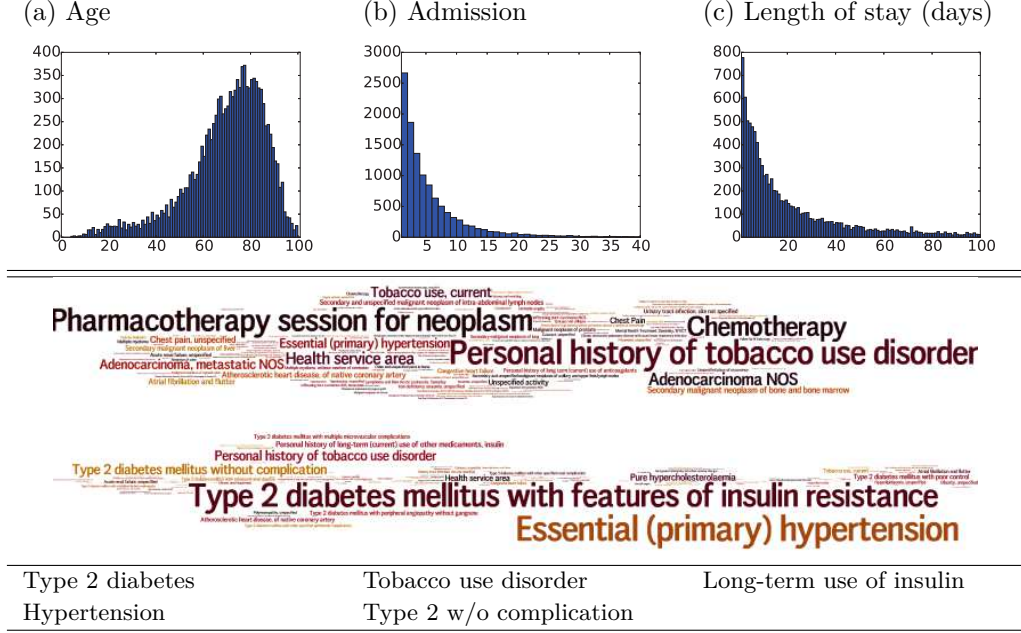


Figure 5: **Top row**: Diabetes cohort statistics (y axis: number of patients; x axis: (a) age, (b) number of admissions, (c) number of days); **Mid row**: Progression from pre-diabetes (upper diag. cloud) to post-diabetes (lower diag. cloud). The size of a diagnosis in a cloud is proportional to its occurrence in the data; **Bottom row**: Top diagnoses.

7,191 patients with 53,208 admissions. The vocabulary consisted of 243 diagnoses, 773 procedures and 353 medication codes. The mental health cohort contained more than 11,000 patients (49.4% males, median age 37). Data statistics are summarized in Fig. 6. After preprocessing, the mental health dataset contained 6,109 patients and 52,049 admissions with the vocabulary of 247 diagnoses, 752 procedures and 319 medication codes. The average age of diabetic patients is much higher than the average age of mental patients (See Fig 5a and Fig 6a).

For each dataset, 2/3 is used for parameter estimation, 1/6 is for tuning, and 1/6 is for testing.

4.2. Disease progression

For disease progression, the model predicts the next n_p diagnoses at each discharge (see Sec.3.4.1). For comparison, we implemented two baselines: Markov models and plain RNNs. A Markov model is a stochastic model used to model changing systems. A Markov model consists of a list of possible states, the possible transitions between those states and the probability of those transitions. The future states depend only on the present state (Markov assumption). The Markov model has memoryless disease transition probabilities $P(d_t^i | d_{t-1}^j)$ from disease d^j to d^i at time t . Given an admission with disease subset D_t ,

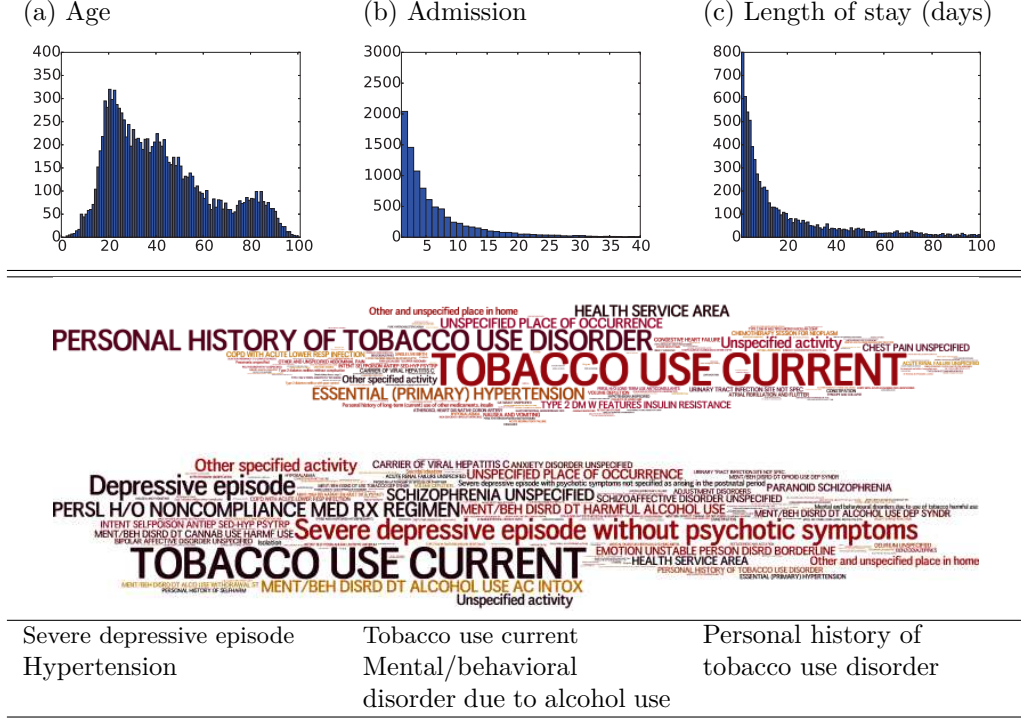


Figure 6: **Top row**: Mental health cohort statistics (y axis: number of patients; x axis: (a) age, (b) number of admissions, (c) number of days); **Mid row**: Progression from pre-mental diseases (upper diag. cloud) to post-mental diseases (lower diag. cloud). The size of a diagnosis in a cloud is proportional to its occurrence in the data; **Bottom row**: Top diagnoses.

the next disease probability is estimated as $Q(d^i; t) = \frac{1}{|D_t|} \sum_{j \in D_t} P(d_t^i | d_{t-1}^j)$. Plain RNNs are described in Sec. 2.1.

We use Precision at K (Precision@K) to measure the performance of the models. Precision@K corresponds to the percentage of relevant results in retrieved results. That means if the model predicts n_p diagnoses of the next readmission and n_r diagnoses among of them are relevant the model's performance is

$$\text{Precision@}n_p = \frac{n_r}{n_p}$$

Dynamics of forgetting

Fig. 7(left) plots the contribution of time into the forget gate. The contributions for all 40 states are computed using $Q_f q_{\Delta_t}$ as in Eq. (11). There are two distinct patterns: decay and growing. This suggests that the time-based forgetting has a very small dimensionality, and we will under-parameterize time using decay only as in Eq. (10), and over-parameterize time using full parameterization as in Eq. (11). A right balance is

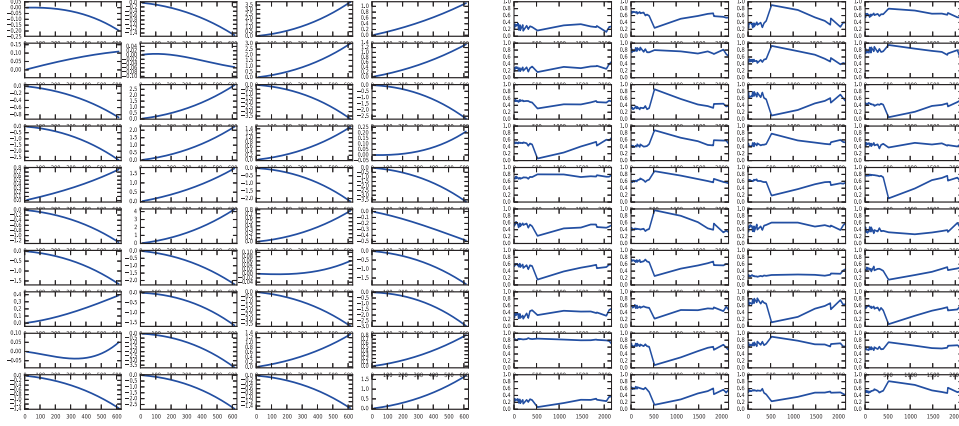


Figure 7: (Left) 40 channels of forgetting due to time elapsed. x axis is Δ_t from 0 to 365 days and y axis is the values of parameterized time in forget gate $Q_f q_{\Delta_t}$. (Right) 40 channels of the forget gates of a patient in the course of their illness.

interesting to warrant a further investigation. Fig. 7(right) shows the evolution of the forget gates through the course of illness (2000 days) for a patient.

395 *Diagnoses prediction result*

Table 1 reports the Precision@ n_p for different values of n_p . For diabetes cohort, using plain RNN improves over memoryless Markov model by 8.8% with $n_p = 1$ and by 27.7% with $n_{pred} = 3$. This significant improvement demonstrates the role of modeling the dynamics in sequential data. Modeling irregular timing and interventions in DeepCare
400 gains a further 2% improvement. For mental health cohort, Markov model fails to predict the next diagnoses (9.5% for $n_p = 1$). Plain RNN gains 50% improvement in Precision@1, while and DeepCare demonstrates a 2% improvement in Precision@1 over RNN.

4.3. Intervention recommendation

We first conducted experiments with DeepCare for intervention recommendation task.
405 The model predicts the current n_p interventions at each admission (see Sec. 4.3). As the current interventions are now the output of the prediction, DeepCare only read the current diagnoses and the previous interventions as input. The Eq. 8 now becomes

$$\mathbf{o}_t = \sigma(W_o \mathbf{x}_t + U_o \mathbf{h}_{t-1} + \mathbf{b}_o)$$

Table 2 reports the results of current intervention prediction. For all values of n_p , RNN consistently outperforms Markov model by a huge margin for both diabetes and

Table 1: Precision@ n_p diagnoses prediction with the confidence interval (CIs), estimated using bootstrap.

Model	Diabetes		
	$n_p = 1$ (95% CIs)	$n_p = 2$ (95% CIs)	$n_p = 3$ (95% CIs)
Markov	55.1 (53.0-57.2)	34.1 (32.5-35.7)	24.3 (23.2-25.5)
Plain RNN (Sec. 2.1)	63.9 (62.3-65.4)	58.0 (56.5-59.5)	52.0 (50.5-53.4)
LSTM (Sec. 2.1)	65.7 (64.2-67.4)	59.6 (58.1-61.1)	53.3 (51.8-54.8)
DeepCare (time decay)	64.9 (63.4-66.4)	58.9 (57.5-60.3)	53.2 (51.8-54.6)
DeepCare (mean adm.)	66.2 (64.6-67.7)	59.6 (58.1-61.1)	53.7 (52.3-55.2)
DeepCare (sum adm.)	65.5 (64.0-67.2)	59.3 (57.8-60.9)	53.5 (52.1-55.0)
DeepCare (max adm.)	66.1 (64.6-67.6)	59.2 (57.7-60.7)	53.2 (51.7-54.7)

Model	Mental		
	$n_p = 1$ (95% CIs)	$n_p = 2$ (95% CIs)	$n_p = 3$ (95% CIs)
Markov	9.5 (7.9-11.1)	6.4 (5.4-7.4)	4.4 (3.4-5.1)
Plain RNN (Sec. 2.1)	50.7 (48.9-52.4)	45.7 (44.1-47.3)	39.5 (38.2-40.8)
LSTM (Sec. 2.1)	51.0 (49.1-52.9)	46.4 (44.7-48.1)	40.0 (38.7-41.3)
DeepCare (time decay)	51.3 (49.6-53.0)	46.4 (44.7-48.0)	39.8 (38.5-41.0)
DeepCare (mean adm.)	52.7 (50.8-54.4)	46.9 (45.3-48.5)	40.2 (39.0-41.4)
DeepCare (sum adm.)	51.7 (49.9-53.5)	46.2 (44.6-47.9)	39.8 (38.5-41.1)
DeepCare (max adm.)	51.5 (49.6-53.3)	46.7 (45.0-48.3)	40.2 (38.9-41.5)

410 mental health cohort. DeepCare with sum-pooling outperforms other models in both diabetes and mental health datasets.

4.4. Predicting future risk

Next we demonstrate DeepCare on long-range risk prediction (see Sec. 3.4.3). For each patient, a discharge is randomly chosen as a prediction point, from which *unplanned*
415 *readmission* and *high risk patients* within X months will be predicted. A patient is at high risk at a particular time T if he or she have at least three unplanned readmissions within X months after time T . We choose $X = 12$ months for diabetes and $X = 3$ months for mental health. Results are measured in F1-score.

For comparison, **baselines** are SVM and Random Forests running on standard non-
420 temporal features engineering using one-hot representation of diagnoses and intervention codes, and plain RNN and LSTM running on sequences of admissions. One-hot representation of a code is a vector with the dimension equal to the vocabulary size, the value at the code index is 1 and all other indices are 0. Then pooling is applied to aggregate over all existing admissions for each patient. Two pooling strategies are tested: *max* and *sum*.
425 Max-pooling is equivalent to the presence-only strategy in [1], and sum-pooling is akin to an uniform convolutional kernel in [45]. This feature engineering strategy is equivalent to zeros-forgetting – any risk factor occurring in the past is memorized.

Table 2: Precision@ n_p intervention prediction with the confidence interval (CIs), estimated using bootstrap.

Model	Diabetes		
	$n_p = 1$ (95% CIs)	$n_p = 2$ (95% CIs)	$n_p = 3$ (95% CIs)
Markov	35.0 (32.7-37.4)	17.6 (16.4-18.7)	11.7 (10.9-12.5)
Plain RNN (Sec. 2.1)	77.7 (75.6-79.6)	54.8 (53.7-55.9)	43.1 (42.1-44.2)
LSTM (Sec. 2.1)	78.2 (76.3-80.0)	54.7 (53.8-55.7)	42.9 (42.0-43.9)
DeepCare (time decay)	77.0 (74.9-78.9)	54.2 (53.1-55.3)	42.8 (41.7-43.8)
DeepCare (mean adm.)	77.8 (76.3-79.5)	54.9 (53.9-55.9)	43.3 (42.3-44.3)
DeepCare (sum adm.)	78.7 (77.1-80.4)	55.5 (54.5-56.5)	43.5 (42.4-44.6)
DeepCare (max adm.)	78.4 (76.7-80.1)	55.1 (54.1-56.1)	43.4 (42.3-44.5)

Model	Mental		
	$n_p = 1$ (95% CIs)	$n_p = 2$ (95% CIs)	$n_p = 3$ (95% CIs)
Markov	20.7 (18.2-23.4)	12.2 (10.5-13.4)	8.1 (7.0-9.3)
Plain RNN (Sec. 2.1)	70.4 (67.6-73.2)	55.4 (53.0-58.0)	43.7 (41.9-45.6)
LSTM (Sec. 2.1)	70.9 (68.3-73.3)	55.6 (53.2-58.1)	44.2 (42.5-45.9)
DeepCare (time decay)	70.5 (67.6-73.0)	55.5 (53.0-58.1)	43.9 (42.0-45.7)
DeepCare (mean adm.)	70.3 (67.4-73.0)	55.7 (53.0-58.5)	44.1 (42.3-46.0)
DeepCare (sum adm.)	71.0 (68.2-73.9)	55.8 (53.4-58.3)	44.7 (43.0-46.4)
DeepCare (max adm.)	70.0 (67.3-72.6)	55.2 (52.8-57.9)	43.9 (42.2-45.9)

Pretraining and Regularization

Table 3 reports the effects of pretraining and regularization on unplanned readmission prediction in diabetes dataset using DeepCare model. Pretraining and regularization improve the results of all three admission pooling methods. While mean pooling admission is found to perform well with regularization, max pooling produces best results with pretraining and sum pooling produces best results with both approaches. Further details of pretraining and regularization are given in Appendix A.2 and Appendix A.3.

Table 3: Effect of pretraining and regularization for unplanned readmission prediction using DeepCare for diabetes dataset. The results are reported in F-score (%)

Approach	Mean adm.	Sum adm.	Max adm.
None	77.8	77.9	78.3
Pretrain	78.3	78.6	78.9
Regularization	79.0	78.7	78.6
Both	78.4	78.9	78.8

Unplanned readmission prediction results

Table 4 reports the F-scores of predicting unplanned readmission. For the diabetes cohort, the best baseline (non-temporal) is Random Forests with *sum pooling* has a F-score of 71.4% [Row 4]. Using plain RNN with simple logistic regression improves over

best non-temporal methods by a 3.7% difference in 12-months prediction [Row 5, ref: Sec. (2.1,3.2)]. Replacing RNN units by LSTM units gains 4.5% improvement [Row 6, ref: Sec. 2.1]. Moving to deep models by using a neural network as classifier helps with a gain of 5.1% improvement [Row 7, ref: Eq. (17)]. By carefully modeling the irregular timing, interventions and recency+multiscale pooling, we gain 5.7% improvement [Row 8, ref: Secs. (3.3.2–3.4.3)]. Finally, with parametric time we arrive at 79.0% F-score, a 7.6% improvement over the best baselines [Row 9, ref: Secs. (3.3.2)].

For the mental health dataset, the best non-temporal baseline is *sum-pooling* Random Forest with result of 67.9%. Plain RNN and LSTM with logistic regression layer gain 2.6% and 3.8% improvements, respectively. The best model is DeepCare with parametric time with a gap of 6.8% improvement compared to *sum-pooling* Random Forest.

Table 4: Results of unplanned readmission prediction in F-score (%) with confidence interval (CIs) within 12 months for diabetes and 3 months for mental health patients. DeepCare 1 is *nnets + mean adm.*; DeepCare 2 is [*interven. + time decay*] + *recent.multi.pool. + nnets + mean adm.*; DeepCare 3 is [*interven. + param. time*] + *recent.multi.pool. + nnets + mean adm.* (*) statistical significance over non-temporal models, and (**) statistical significance over temporal models.

Model	Diabetes (95% CIs)	Mental (95% CIs)
1. SVM (<i>max-pooling</i>)	64.0 (62.2-65.8)	64.7 (62.0-67.4)
2. SVM (<i>sum-pooling</i>)	66.7 (64.9-68.4)	65.9 (63.2-68.8)
3. Random Forests (<i>max-pooling</i>)	68.3 (66.2-70.5)	63.7 (61.1-66.6)
4. Random Forests (<i>sum-pooling</i>)	71.4 (69.4-73.4)	67.9 (65.2-70.6)
5. Plain RNN (Sec. 2.1) (<i>logist. regress.</i>)	75.1 (73.4-76.9)	70.5 (68.0-73.0)
6. LSTM (Sec. 2.1) (<i>logit. regress.</i>)	75.9 (74.1-77.7)	71.7 (67.8-73.0)
7. DeepCare 1	76.5* (74.7-78.2)	72.8* (70.3-75.2)
8. DeepCare 2	77.1* (75.4-78.9)	74.5** (72.2-76.6)
9. DeepCare 3	79.0** (77.2-80.9)	75.4** (73.1-77.5)

High risk prediction results

In this part, we report the performance of DeepCare on high risk patient prediction task. Figure 8 reports the F-score of high risk prediction. RNN improves the best non-temporal model (*sum-pooling* SVM) over 10% F-score for both two cohorts. Max-pooling DeepCare performs best in the diabetes dataset with nearly 60% F-score, while sum-pooling DeepCare wins in the mental health cohort with 50.0% F-score.

5. Discussion

5.1. DeepCare as a model of healthcare memory

DeepCare makes use of embedding to represent the semantics of diagnoses, interventions and admissions. In theory, this embedding is agnostic to of the task at hand. Our

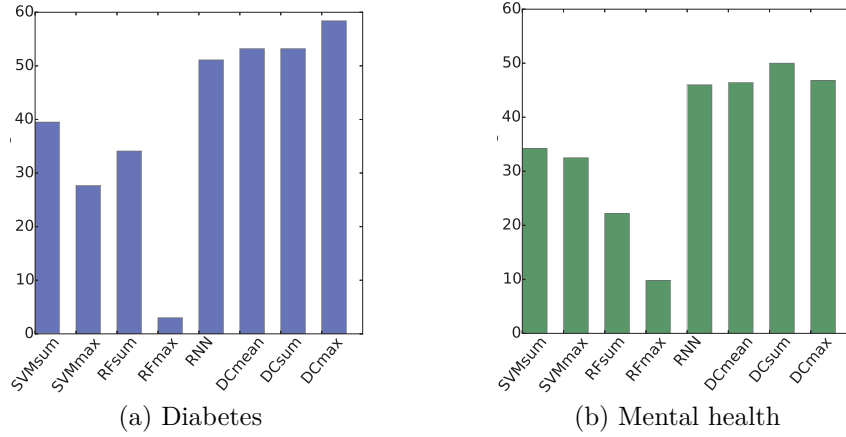


Figure 8: Result of high risk prediction in F-score (%) within 12 months for diabetes (a) and 3 months for mental health (b). DC is DeepCare. Mean, sum, max are 3 admission pooling methods

previous work learns diagnosis and patient embedding [46] using nonnegative restricted Boltzmann machines [33] and known semantic relations and temporal relations [34]. This method uses global contexts, unlike DeepCare, where only local contexts (e.g., next admission) are considered.

It is interesting to see the performance of the model with different pooling methods on embedding vectors. While mean pooling performs best on diagnoses prediction, sum pooling performs best on intervention prediction in both datasets. More evaluations and analyses will be investigated to understand the results. However, the recording practice may hinder a full explanation. For example, codes are recorded for billing purposes, hence there are biases and missing codes. There are also variations between coders. Mean-pooling may be more robust against these potential noises (due to law of large number), and this may explain the results in next-disease prediction (Table 1). However, for treatment recommendation (Table 2), as the treatments are disease-specific, the sum of diseases (sum-pooling) explains the treatments better.

The memory cells in DeepCare are used to store, update, forget and manipulate illness experiences over time-stamped episodes. The inferred experiences are then pooled to reason about the current illness states and the future prognosis. Like human memory, healthcare risk also has a recency effect, that is, more recent events contribute more towards future risk. In DeepCare, two recency mechanisms are used. First, through forgetting, recent events in DeepCare tend to contribute more to the current illness states. The forgetting gate is influenced by the interventions. While it may appear that the influence is only in short-term, but it is actually not because the multiplicative nature of the forget gate, and the long-range dependency of the memory. For example, if the forget

gate is turned off, then the entire illness history will be forgotten. Second, multiscale pooling (Sec. 3.4.3) has weights that decay over time.

485 DeepCare can be implemented for existing EMR systems. More extensive evaluations on a variety of cohorts, sites and outcomes will be necessary. This offers opportunities for domain adaptations through parameter sharing among multiple cohorts and hospitals.

5.2. DeepCare in relations with existing models

Although healthcare is inherently episodic in nature, it has been well-recognized that 490 the entire illness trajectory is important [14, 23]. Nursing illness trajectory model was popularized by Strauss and Corbin [10], but the model is qualitative and imprecise in time [19]. Thus its predictive power is limited.

Electronic medical records (EMRs) offer a quantitative alternative with precise timing of events. However, EMRs are complex – they reflect the interleaving between the illness 495 processes and care processes. The timing is irregular – patients only visit hospital when the illness is beyond a certain threshold, even though the illness may have been present long before the visit. Existing work that handles such irregularities includes interval-based extraction [45], but this method is coarse and does not explicitly model the illness dynamics.

500 Capturing disease progression has been of great interest [25, 30], and much effort has been spent on Markov models [24, 50] and dynamic Bayesian networks [35]. However, healthcare is inherently non-Markovian due to the long-term dependencies. For example, a routine admission with irrelevant medical information would destroy the effect of severe illness [1], especially for chronic conditions. Irregular timing and interventions have not 505 been adequately modeled to reflect their roles in disease progression [22]. Irregular-time Bayesian networks [40] offer a promise, but its power has yet to be demonstrated. Further, assuming discrete states are inefficient since the information pathway has only $\log(K)$ bits for K states. Our work assumes distributed and continuous states, thus offering a much larger state space.

510 *Deep learning* is currently at the center of a new revolution in making sense of a large volume of data. It has achieved great successes in cognitive domains such as speech, vision and NLP [27]. To date, deep learning approach to healthcare has largely been an unrealized promise, except for several very recent works [6, 9, 28, 29, 46], where irregular timing is not properly modeled. In [32], time gaps are coded as a discrete word and 515 temporal motifs are detected using convolutional nets.

5.3. Limitations

We recognize several limitations. First DeepCare has been designed primarily for coded data (diagnosis, procedure and medication) at the admission level. Numerical data

such as blood sugar levels could be naturally incorporated, however. For time-series data,
520 we can extract a feature vector per series. Second DeepCare is more powerful with long
trajectories of many episodes, whereas young patients typically have only one or two
admissions. With short trajectories, other architectures may be more appropriate [6, 32].

The choice of SVM and Random Forest for baselines of readmission task using one-hot
representation of medical codes is naive. Comparing SVM and Random Forests with non-
525 temporal features against temporal model (e.g., plain RNN and LSTM) is to emphasize
the effectiveness of modeling the temporal property. There is other advanced work that
can account for the temporality in healthcare, such as [7, 51]. Our DeepCare contributions
against these temporal models are modeling the irregular timing and the interventions.

5.4. Conclusion

530 In this paper we have introduced DeepCare, an *end-to-end* deep dynamic memory
neural network for personalized healthcare. It frees model designers from manual fea-
ture extraction. DeepCare reads medical records, memorizes illness trajectories and care
processes, estimates the present illness states, and predicts the future risk. Our frame-
work models disease progression, supports intervention recommendation, and provides
535 prognosis from electronic medical records. To achieve precision and predictive power,
DeepCare extends the classic Long Short-Term Memory by (i) embedding variable-size
discrete admissions into vector space, (ii) parameterizing time to enable irregular tim-
ing, (iii) incorporating interventions to reflect their targeted influence in the course of
illness and disease progression; (iv) using multiscale pooling over time; and finally (v)
540 augmenting a neural network to infer about future outcomes. We have demonstrated
DeepCare on predicting next disease stages, recommending interventions, and estimating
unplanned readmission among diabetic and mental health patients. The results are com-
petitive against current state-of-the-arts. DeepCare opens up a new principled approach
to predictive medicine.

545 6. Acknowledgement

This work is partially supported by the Telstra-Deakin Centre of Excellence in Big
Data and Machine Learning

References

- 550 [1] Ognjen Arandjelović. Discovering hospital admission patterns using models learnt
from electronic hospital records. *Bioinformatics*, 2015.

- [2] Pierre Baldi and Peter J Sadowski. Understanding dropout. In *Advances in Neural Information Processing Systems*, pages 2814–2822, 2013.
- [3] Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007.
- [4] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.
- [5] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. *arXiv preprint arXiv:1606.01865*, 2016.
- [6] Yu Cheng, Fei Wang, Ping Zhang, and Jianying Hu. Risk prediction with electronic health records: A deep learning approach. In *SIAM International Conference on Data Mining*. SIAM, 2016.
- [7] Edward Choi, Mohammad Taha Bahadori, and Jimeng Sun. Doctor AI: Predicting Clinical Events via Recurrent Neural Networks. *arXiv preprint arXiv:1511.05942*, 2015.
- [8] Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter Stewart. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In *Advances in Neural Information Processing Systems*, pages 3504–3512, 2016.
- [9] Youngduck Choi. Learning low-dimensional representations of medical concepts. *Proceedings of the AMIA Summit on Clinical Research Informatics (CRI)*, 2016.
- [10] Juliet M Corbin and Anselm Strauss. A nursing model for chronic illness management based upon the trajectory framework. *Research and Theory for Nursing Practice*, 5(3):155–174, 1991.
- [11] Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. *arXiv preprint arXiv:1511.01432*, 2015.
- [12] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, 11:625–660, 2010.

- [13] Joseph Futoma, Jonathan Morris, and Joseph Lucas. A comparison of models for predicting early hospital readmissions. *Journal of biomedical informatics*, 56:229–238, 2015.
- 585 [14] Bradi B Granger, Debra Moser, Barbara Germino, Joanne Harrell, and Inger Ekman. Caring for patients with chronic heart failure: The trajectory model. *European Journal of Cardiovascular Nursing*, 5(3):222–227, 2006.
- [15] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.
- 590 [16] Alex Graves, Marcus Liwicki, Horst Bunke, Jürgen Schmidhuber, and Santiago Fernández. Unconstrained on-line handwriting recognition with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 577–584, 2008.
- [17] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(5):855–868, 2009.
- 595 [18] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE, 2013.
- 600 [19] Susan J Henly, Jean F Wyman, and Mary J Findorff. Health and illness over time: The trajectory perspective in nursing science. *Nursing research*, 60(3 Suppl):S5, 2011.
- [20] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.
- 605 [21] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [22] George Hripcsak, David J Albers, and Adler Perotte. Parameterizing time in electronic health record studies. *Journal of the American Medical Informatics Association*, page ocu051, 2015.
- 610 [23] Zhengxing Huang, Wei Dong, Huilong Duan, and Haomin Li. Similarity measure between patient traces for clinical pathway analysis: problem, method, and applications. *IEEE Journal of Biomedical and Health Informatics*, 18(1):4–14, 2014.

- [24] Christopher H Jackson, Linda D Sharples, Simon G Thompson, Stephen W Duffy, and Elisabeth Couto. Multistate Markov models for disease progression with classification error. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 52(2):193–209, 2003.
- [25] Anders Boeck Jensen, Pope L Moseley, Tudor I Oprea, Sabrina Gade Ellesøe, Robert Eriksson, Henriette Schmock, Peter Bjødstrup Jensen, Lars Juhl Jensen, and Søren Brunak. Temporal disease trajectories condensed from population-wide registry data covering 6.2 million patients. *Nature communications*, 5, 2014.
- [26] Peter B Jensen, Lars J Jensen, and Søren Brunak. Mining electronic health records: towards better research applications and clinical care. *Nature Reviews Genetics*, 13(6):395–405, 2012.
- [27] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
- [28] Zhaohui Liang, Gang Zhang, Jimmy Xiangji Huang, and Qmming Vivian Hu. Deep learning for healthcare decision making with EMRs. In *Bioinformatics and Biomedicine (BIBM), 2014 IEEE International Conference on*, pages 556–559. IEEE, 2014.
- [29] Z. Lipton, D. Kale, C. Elkan, and R. Wetzel. Learning to Diagnose with LSTM Recurrent Neural Networks. In *International Conference on Learning Representations (ICLR 2016)*, 2016.
- [30] Chuanren Liu, Fei Wang, Jianying Hu, and Hui Xiong. Temporal phenotyping from longitudinal electronic health records: A graph based framework. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 705–714. ACM, 2015.
- [31] H.M. Lu, D. Zeng, and H.C. Chen. Prospective Infectious Disease Outbreak Detection Using Markov Switching Models. *IEEE Transactions on Knowledge and Data Engineering*, 2009.
- [32] Phuoc Nguyen, Truyen Tran, Nilmini Wickramasinghe, and Svetha Venkatesh. Deepr: A convolutional net for medical records. *IEEE Journal of Biomedical and Health Informatics*, 2016.
- [33] T.D. Nguyen, T. Tran, D. Phung, and S. Venkatesh. Learning Parts-based Representations with Nonnegative Restricted Boltzmann Machine . In *Proc. of 5th Asian Conference on Machine Learning (ACML)*, Canberra, Australia, Nov 2013.

- [34] Tu Dinh Nguyen, Truyen Tran, Dinh Phung, and Svetha Venkatesh. Graph-induced restricted boltzmann machines for document modeling. *Information Sciences*, 328:60–75, 2016.
- [35] Kalia Orphanou, Athena Stassopoulou, and Elpida Keravnou. Temporal abstraction and temporal Bayesian networks in clinical domains: A survey. *Artificial intelligence in medicine*, 60(3):133–149, 2014.
- [36] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318, 2013.
- [37] Chris Paxton, Suchi Saria, and Alexandru Niculescu-Mizil. Developing predictive models using electronic medical records: challenges and pitfalls. In *AMIA*, 2013.
- [38] Trang Pham, Truyen Tran, Dinh Phung, and Svetha Venkatesh. Deepcare: A deep dynamic memory model for predictive medicine. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 30–41. Springer, 2016.
- [39] Trang Pham, Truyen Tran, Dinh Phung, and Svetha Venkatesh. Faster training of very deep networks via p-norm gates. *ICPR’16*, 2016.
- [40] Michael Ramati and Yuval Shahar. Irregular-time Bayesian networks. *UAI*, pages 484–491, 2010.
- [41] Ralph Snyderman and R Sanders Williams. Prospective medicine: the next health care transformation. *Academic Medicine*, 78(11):1079–1084, 2003.
- [42] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.
- [43] Ewout W Steyerberg. *Clinical prediction models: a practical approach to development, validation, and updating*. Springer, 2009.
- [44] Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.
- [45] Truyen Tran, Wei Luo, Dinh Phung, Sunil Gupta, Santu Rana, Richard L Kennedy, Ann Larkins, and Svetha Venkatesh. A framework for feature extraction from hospital medical data with applications in risk prediction. *BMC bioinformatics*, 15(1):6596, 2014.

- [46] Truyen Tran, Tu Dinh Nguyen, Dinh Phung, and Svetha Venkatesh. Learning vector representation of medical objects via EMR-driven nonnegative restricted Boltzmann machines (eNRBM). *Journal of biomedical informatics*, 54:96–105, 2015.
- 680 [47] Truyen Tran, Dinh Phung, Wei Luo, Richard Harvey, Michael Berk, and Svetha Venkatesh. An integrated framework for suicide risk prediction. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1410–1418. ACM, 2013.
- [48] Truyen Tran, Dinh Phung, Wei Luo, and Svetha Venkatesh. Stabilized sparse ordinal regression for medical risk stratification. *Knowledge and Information Systems*, 2014. DOI: 10.1007/s10115-014-0740-4.
- 685 [49] Fei Wang, Noah Lee, Jianying Hu, Jimeng Sun, Shahram Ebadollahi, and Andrew F Laine. A framework for mining signatures from event sequences and its applications in healthcare data. *IEEE transactions on pattern analysis and machine intelligence*, 35(2):272–285, 2013.
- 690 [50] Xiang Wang, David Sontag, and Fei Wang. Unsupervised learning of disease progression models. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 85–94. ACM, 2014.
- [51] Jing Zhao, Panagiotis Papapetrou, Lars Asker, and Henrik Boström. Learning from heterogeneous temporal data in electronic health records. *Journal of Biomedical Informatics*, 65:105–119, 2017.
- 695

Appendix A. Supplementary materials

Appendix A.1. Model complexity

The number of model parameters are $M \times |V| + M \times K + K \times K + K \times D$, which consists of the following components:

700

Parameters in the C-LSTM layer

- For admission embedding, we use two embedding matrices A and B . We have $A + B \in \mathbb{R}^{M \times |V|}$
 - The input gate: $W_i \in \mathbb{R}^{M \times K}$, $U_i \in \mathbb{R}^{K \times K}$ and $b_i \in \mathbb{R}^{K \times 1}$
 - The output gate: $W_o \in \mathbb{R}^{M \times K}$, $U_o \in \mathbb{R}^{K \times K}$, $P_o \in \mathbb{R}^{K \times K}$ and $b_o \in \mathbb{R}^{K \times 1}$
- 705

- The forget gate: $W_f \in \mathbb{R}^{M \times K}$, $U_f \in \mathbb{R}^{K \times K}$, $P_f \in \mathbb{R}^{K \times K}$ and $b_f \in \mathbb{R}^{K \times 1}$. In the case of time decay there are no other parameters and in the case of parametric time, the forget gate has a time weight matrix $Q_f \in \mathbb{R}^{N_{time} \times K}$ ($N_{time} = 3$ in our implementation)
- The memory cell: $W_i \in \mathbb{R}^{M \times K}$, $U_i \in \mathbb{R}^{K \times K}$ and $b_i \in \mathbb{R}^{K \times 1}$

Parameters in the Neural network layer

- The neural network layer consists of an input-hidden weight matrix $U_{h1} \in \mathbb{R}^{3K \times D}$, hidden-output weight matrix $U_{h2} \in \mathbb{R}^{D \times 2}$ and two bias vectors $c_1 \in \mathbb{R}^{D \times 1}$ and $c_2 \in \mathbb{R}^{2 \times 1}$

715 *Appendix A.2. Pretraining*

Pretraining can be done by unsupervised learning on unlabeled data [20, 11]. Pretraining has been proven to be effective because it helps the optimization by initializing weights in a region near a good local minimum [3, 12]. In our work we use auxiliary tasks to pretrain the model for future risk prediction tasks. In our case, auxiliary tasks are predicting diagnoses of the next readmission and predicting interventions of current admission. These tasks play a role in disease progression tracking and intervention recommendation.

We use the bottom layer of DeepCare for training auxiliary tasks. As described in Sec. 3.1, the LSTM layer reads a sequence of admissions $\mathbf{u}_0, \mathbf{u}_1, \dots, \mathbf{u}_n$ and computes the corresponding sequence of distributed illness states $\mathbf{h}_0, \mathbf{h}_1, \dots, \mathbf{h}_n$. At each step t , \mathbf{h}_t is used to generate labels y_t by the formula given in Eqs. (12 and 13) where y_t can be a set of diagnoses or interventions. After training, the code embedding matrix is then used to initialize the embedding matrix for training the risk prediction tasks. The results of next readmission diagnosis prediction and current admission intervention prediction are reported in Sec. 4.2 and Sec. 4.3.

730 *Appendix A.3. Regularization with dropouts*

DeepCare may lead to overfitting because it introduces three more parameter matrices to the sigmoid gates to handle interventions and time. Therefore, we use L2-norm and Dropout to prevent overfitting. L2-norm regularization, also called “weight decay”, is used to prevent weight parameters from extreme values. A constant λ is introduced to control the magnitude of the regularization. Dropout is a regularization method for DNNs. During training, units are deleted with a pre-defined probability $1 - p$ (dropout ratio) and the remaining parts are trained through back-propagation as usual [42, 2]. This prevents the co-adaptation between units, and therefore prevents overfitting. At the test time, a single neural net is used without dropout and the outgoing weights of a unit that

is retained with probability p during training are multiplied by p . This combines 2^k (k is the number of units) shared weight networks into a single neural network at test time. Therefore, dropout is also considered as an ensemble method.

In our implementation, dropout in DeepCare is introduced at input layer and neural
745 network layer:

- *Dropout codes:* Before pooling the embedding vectors of diagnoses and interventions in each admission, each of these embedding vectors is deleted with probability $1 - p_{code}$
- *Dropout input features:* After deriving $[\mathbf{x}_t, \mathbf{p}_t]$ as described in Sec. 3.2, each value
750 in these two vector is dropped with probability $1 - p_{feat}$
- *Dropout units in neural network layer:* The pooled state z as described in Sec. 3.2 is feed as the input of the neural network. Dropout is used at input units with probability $1 - p_{in}$ and at hidden units with probability $1 - p_{hidd}$.

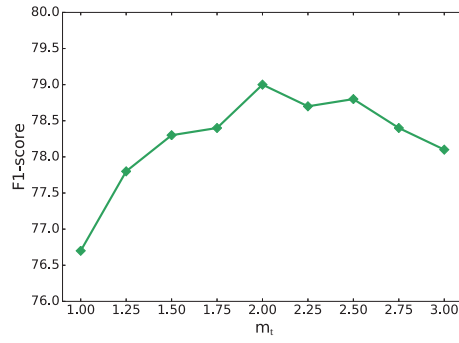
Appendix A.4. DeepCare implementation

DeepCare was implemented in Python using the Theano framework⁵. We vary the
755 embedding and hidden dimensions from 5 to 50 but the results are rather robust. We report best results for disease progression and intervention recommendation tasks with $M = 30$ and $K = 40$ and for prediction tasks with $M = 10$ embedding dimensions and $K = 20$ hidden units (M and K are the number of embedding dimensions and hidden
760 units respectively). The L2-regularizer is set as $\lambda = 0.00025$ and the dropout rates are set as $p_{code} = 0.8$, $p_{feat} = 0.8$ and $p_{hidd} = 0.5$. Learning is by Stochastic Gradient Descent with the mini-batch of 16 sequences. The learning rate λ is modified as follows. We start with $\lambda = 0.01$. When the model cannot find a smaller training cost, we wait n_{wait} epochs before updating λ as $\lambda = \lambda/2$. Initially, $n_{wait} = 5$, and is subsequently modified as
765 $n_{wait} = \min \{15, n_{wait} + 2\}$ for each λ update. Learning is terminated after $n_{epoch} = 200$ or after learning rate smaller than $\epsilon = 0.0001$.

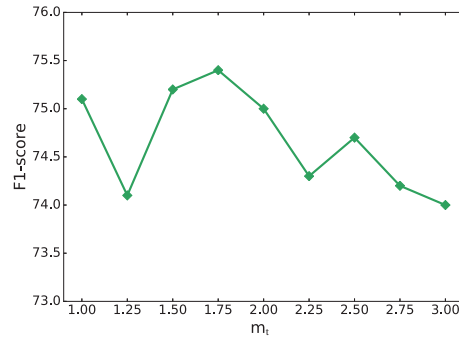
Appendix A.5. Effect of admission method as input

As discussed in Sec. 3.2.2, unplanned admissions usually indicates higher risk. This is reflected in DeepCare model by modeling the admission method m_t in the input gate
770 \mathbf{i}_t (See Eq. 7). We keep $m_t = 1$ for unplanned admissions and choose m_t for planned admissions empirically. m_t is varied in (1, 1.25, 1.5, ..., 2.75, 3). Fig A.9 illustrates the performance of DeepCare on different value of m_t . For diabetes cohort, the best value of m_t is 2 while for mental health cohort, the best value of m_t is 1.75.

⁵<http://deeplearning.net/software/theano>



(a) Diabetes



(b) Mental health

Figure A.9: Performance of DeepCare on unplanned readmission prediction task with different value of m_t for planned admissions. (a) Diabetes cohort and (b) Mental health cohort.