ICE-NODE: Integration of Clinical Embeddings with Neural Ordinary Differential Equations

Asem Alaa ASEM.A.ABDELAZIZ@IMPERIAL.AC.UK
Departments of Computing, Mathematics, and Cancer and Surgery
& UKRI Centre for Doctoral Training in AI for Healthcare,
Imperial College London, UK

Erik Mayer E.MAYER@IMPERIAL.AC.UK
Department of Cancer and Surgery,
Imperial College London, UK

Mauricio Barahona M.BARAHONA@IMPERIAL.AC.UK
Department of Mathematics
& Centre for Mathematics of Precision Healthcare,
Imperial College London, UK

Abstract

Early diagnosis of disease can lead to improved health outcomes, including higher survival rates and lower treatment costs. With the massive amount of information available in electronic health records (EHRs), there is great potential to use machine learning (ML) methods to model disease progression aimed at early prediction of disease onset and other outcomes. In this work, we employ recent innovations in neural ODEs combined with rich semantic embeddings of clinical codes to harness the full temporal information of EHRs. We propose ICE-NODE (Integration of Clinical Embeddings with Neural Ordinary Differential Equations), an architecture that temporally integrates embeddings of clinical codes and neural ODEs to learn and predict patient trajectories in EHRs. We apply our method to the publicly available MIMIC-III and MIMIC-IV datasets, and we find improved prediction results compared to state-of-the-art methods, specifically for clinical codes that are not frequently observed in EHRs. We also show that ICE-NODE is more competent at predicting certain medical conditions, like acute renal failure, pulmonary heart disease and birth-related problems, where the full temporal information could provide important information. Furthermore, ICE-NODE is also able to produce patient risk trajectories over time that can be exploited for further detailed predictions of disease evolution.

1. Introduction

With the wider availability of EHRs, and the massive amount of information they contain, there is a rising demand to exploit such data using current advances in ML to improve healthcare outcomes. For instance, many healthcare systems across the world suffer from delayed cancer diagnosis, leading to lowered survival rates in cancer patients (Arnold et al., 2019). In 2020, over 19 million people were diagnosed with cancer and around 10 million people died from cancer (Ferlay et al., 2021). In England and Wales alone, over three hundred and fifty thousand cancer cases are diagnosed yearly (averaged over 2016-2018), and from those diagnosed with cancer in 2010-2011 only 50% survive for ten years or more (Cancer Research UK). Employing new innovations in ML for disease progression modeling (DPM)

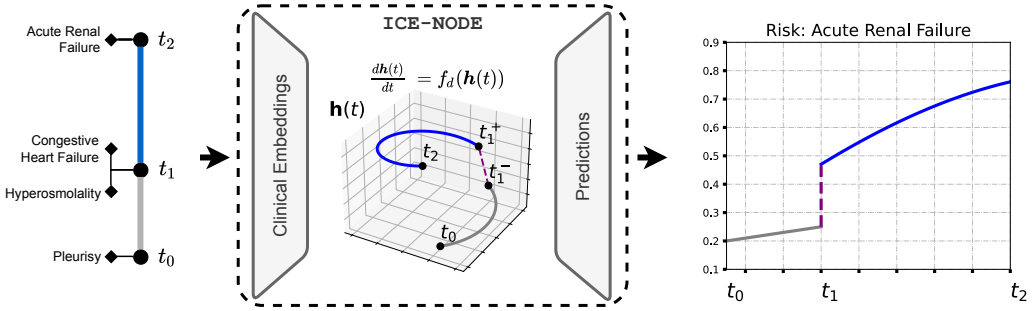


Figure 1: A schematic view of ICE-NODE. The input is an EHR at three consecutive timestamps $\{t_0, t_1, t_2\}$ ICE-NODE learns a neural ODE model from the clinical codes at each timestamp by generating predictions at the observation times of the samples in the training set. Once the model is learnt, it can generate a health state trajectory over time and continuous-time clinical embeddings that can be used to predict clinical codes, enabling risk predictions for medical conditions over time (e.g., for 'Acute renal failure' in this case).

has the potential to take advantage of the available data in patient histories to aid clinicians in their quest for early prediction of disease onset.

Information in EHRs is stored as timestamped *medical concepts* of diverse type (e.g., symptoms, procedures, lab tests). The medical concepts in EHRs are stored using coding schemes, such as *SNOMED CT*[1] or *ICD*[2], which provide a comprehensive, interrelated classification of diseases within a compactly structured data system.

The complexity of the information in EHRs poses a number of challenges for many ML predictive models, which usually make strong assumptions about the data when dealing with feature extraction and representation of clinical codes, and with the treatment of the temporal dimension in the patient history. Many DPM methods (e.g. Cherry et al., 2020; Weegar and Sundström, 2020) transform the EHR into a tabular format discarding the temporal information altogether, whereas other approaches (e.g. Choi et al., 2018, 2016a) only capture the *sequential* information within the EHR. recurrent neural networks (RNNs) and other natural language processing (NLP)-inspired models usually underpin the methods of this latter group. Yet, while these methods can aid in capturing the temporal ordering of medical concepts, they do not capture the irregularity of time intervals between consecutive timestamps. Such assumptions may limit the learning capabilities of models applied to such temporal patterns, which might be important for particular medical conditions.

Recently, a new set of ML methods (Chen et al., 2018; Brouwer et al., 2019; Rubanova et al., 2019; Kidger et al., 2020; Kidger, 2022) have developed a class of neural networks that include ordinary differential equations (ODEs) through implicit layers, as opposed to the common feed-forward static layers. Such neural ODEs models have the potential to learn the dynamics of EHRs as sporadically observed time series, irregularly sampled in time and feature dimensions. Indeed, neural ODEs have shown superiority over other methods (including different variants of GRU and LSTM models) when applied to such time series (Brouwer et al., 2019).

1. https://termbrowser.nhs.uk/
2. https://www.who.int/classifications/classification-of-diseases

In this work, we aim to bring advanced ML methods apt for the analysis of sporadically observed time series to the disease progression modeling task. Specifically, we propose ICE-NODE, a model that learns from the timestamped sequences of diagnostic codes in patient trajectories as they interact with healthcare providers. ICE-NODE takes into account explicitly the temporal dimension in the EHRs using neural ODEs, with a flexible framework that enables integrating advanced embedding modules for the medical concepts. Below, we present, test, and analyse ICE-NODE with two different integrated modules for the embedding of medical concepts: simple matrix embeddings and GRAM embeddings (Choi et al., 2017).

ICE-NODE achieves high predictive accuracy on unseen patient trajectories, outperforming the state-of-the-art methods in certain DPM tasks. In particular, our analysis with ICE-NODE identifies a set of diagnostic codes, including acute renal failure and pulmonary heart disease, which are predicted with a significantly higher accuracy when the irregularity of time intervals between timestamps is incorporated explicitly. Furthermore, we show that ICE-NODE enables access to an inherent (hidden) temporal variable that describes the patient state at any time between timestamps in the patient history. This feature can be exploited to generate a time-continuous disease risk for the patient that could be used as an aid in clinical decision-making (see Fig. 1 for an illustration). Studying and visualising the time-continuous trajectories of multiple medical conditions in this way can enable better understanding of the patient health state and the temporal interlinkages between conditions.

We apply our methodology to two publicly available, de-identified, EHRs datasets available through *PhysioNet* (Goldberger et al., 2000): MIMIC-III (Johnson et al., 2016a,b), which contains EHRs for more than 46K patients over an 11-year period, and MIMIC-IV (Johnson et al., 2021), which contains EHRs for more than 256K patients over 11 years. The source code of our ICE-NODE implementation, along with the numerical experiments in this paper, is available at https://github.com/barahona-research-group/ICE-NODE.

Generalisable Insights about Machine Learning in the Context of Healthcare

Medical conditions evolve in time and may interact with each other, and such complex dynamical interactions are reflected in EHR patient histories. The main motivation behind developing ICE-NODE is to deploy recent innovations in neural ODEs that allow us to take into account the detailed temporal information and the full complexity of diagnostic codes to learn the underlying dynamical health state of patients from timestamped diagnostic codes collected through EHRs. In summary:

- We develop a framework that represents timestamped medical diagnostic codes from patient histories and feeds them to a specialised neural ODE model to learn the temporal evolution of medical conditions. ICE-NODE incorporates the full temporal information of EHRs, unlike traditional methods that ignore time or use only sequential orderings.

- We show through performance analysis that exploiting the full temporal information results in a significant improvement in the prediction accuracy of disease onset for particular medical conditions like acute renal failure and pulmonary heart disease.

- We learn time-continuous trajectories for disease risk prediction, and discuss opportunities to exploit these trajectories for a better understanding of patient health.

2. Related Work

The high explanatory potential of EHR data and their availability through databases such as *PhysioNet* (Goldberger et al., 2000) has motivated a large body of research on learning from EHRs, while tackling their complex and unstructured format (Xiao et al., 2018; Choi et al., 2016b,a, 2017; Cai et al., 2018; Choi et al., 2018; Zhang et al., 2018; Peng et al., 2019; Wang et al., 2019; Zhu and Razavian, 2019; Choi et al., 2020; Xu et al., 2020).

Many of the traditional DPM methods in the literature transform the unstructured EHRs into a tabular format before applying common machine learning methods (Cherry et al., 2020; Ferroni et al., 2019; Weegar and Sundström, 2020; Seneviratne et al., 2018). This process involves careful *feature engineering*, whereby features from the EHRs are treated and extracted based on prior knowledge by domain experts. Such methods have managed to achieve accurate predictions on very specific DPM tasks, such as pancreatic cancer prediction (Cherry et al., 2020), breast cancer prognosis (Ferroni et al., 2019), cervival cancer prediction (Weegar and Sundström, 2020), and metastatic prostate cancer prediction (Seneviratne et al., 2018). However, such methods suffer from issues of generalisability, including the fact that the selection of features by domain experts may prevent discovering new features that could help explain the disease progression. Additionally, ignoring the time dimension discards critical information on how the features evolve with time, thus preventing opportunities to learn the temporal patterns in EHRs.

A major challenge in learning from EHRs is thus to incorporate the temporal dimension of the data. Until now, several methods have considered only the *temporal ordering* of the clinical codes, discarding the irregular time intervals between timestamps. Those approaches have enabled the application of a wide array of computational methods, mostly inspired by research in NLP, where EHRs are seen as 'text' formed by a succession of 'words' (medical codes). Inspired by the eponymous `Doc2Vec` algorithm (Le and Mikolov, 2014) for NLP, Choi et al. (2016b) proposed `Med2Vec`, which learns latent clinical codes and visit-level representations. In the same year, Choi et al. (2016a) developed the `RETAIN` model that employs an attention mechanism integrated with RNN to learn to attend patient visits in reverse time order mimicking the behaviour of physicians when screening patient history. Later on, Choi et al. (2017) presented `GRAM`, a method that employs an attention mechanism that has access to a medical ontology that organises the clinical codes into a hierarchical structure. `GRAM` is motivated when the dataset contains clinical codes that are sparse and rare, mitigating the risk of model overfitting on those rare codes. In (Choi et al., 2018), `MIME` is presented as a multilevel representation learning method. The clinical codes in each visit are separated into treatment codes and diagnosis codes, each fed into different passes to the model. In a task for heart failure prediction, `MIME` outperformed both `Med2Vec` (Choi et al., 2016b) and `GRAM` (Choi et al., 2017), which were both trained only on diagnostic codes.

Although several of the aforementioned methods exploit the temporal ordering of visits, none of them use the length of delays between the visits and its potential impact in describing the patient state. Zhang et al. (2018) proposed `Patient2Vec`, which applies time-binning to patient visits and then employs a double attention mechanism: one to produce latent representations from the clinical codes within each time bin, and a second one to integrate the latent representation of the time bins into a sequence using a gated recurrent unit (GRU). In the task of predicting future hospitalisation from an EHR, `Patient2Vec` (Zhang et al.,

2018) outperformed RETAIN (Choi et al., 2016a). Although Patient2Vec uses some of the temporal information, time-binning still imposes strong assumptions on the structure of the data. In addition, Cai et al. (2018) developed a time-aware attention mechanism and Peng et al. (2019) developed the TeSAN model; these two approaches exploit the time gaps between successive patient visits based on the neural attention mechanism.

Our method departs from these methods by employing neural ODEs to model the temporal evolution of the medical conditions without over-engineering the attention mechanism, yet explicitly incorporating the time-intervals between consecutive timestamps. Our proposed method strives to provide a simple framework that (i) flexibly incorporates several information sources, such as timestamped procedures and timestamped numerical lab tests, and (ii) provides a simple, direct reconstruction of time-continuous risk trajectories between the consecutive timestamps.

3. Methods

In this section, we first describe the two main constituents of ICE-NODE: neural ODEs and medical code embeddings. After that, We introduce the ICE-NODE model and architecture.

Notation. Throughout, we use $[\boldsymbol{A}, \boldsymbol{B}]$ to denote the horizontal concatenation of two vectors or matrices, and $[\boldsymbol{A}; \boldsymbol{B}]$ for vertical concatenation. When writing functions, we use a semicolon to separate the function input variables from its parameters, e.g., $f(\boldsymbol{x}; \boldsymbol{\theta})$.

3.1. Neural ODEs

We use neural ODEs to learn from timestamped diagnostic codes from EHRs. In contrast to RNN models, which only use the temporal ordering of medical codes, ODEs capture the irregular intervals between timestamps, which range from days to months or even years.

We begin our design by assuming that each patient is described by a time-continuous hidden state $\boldsymbol{h}(t) \in \mathbb{R}^{d_h}$ with d_h dimensions. A system of ODEs is used to model the temporal evolution of $\boldsymbol{h}(t)$ as:

$$\frac{\mathrm{d}\boldsymbol{h}(t)}{\mathrm{d}t} = f_d\left(\boldsymbol{h}(t); \boldsymbol{\theta}_d\right) \tag{1}$$

where $f_d : \mathbb{R}^{d_h} \mapsto \mathbb{R}^{d_h}$ is the dynamics function parametrised by $\boldsymbol{\theta}_d \in \mathbb{R}^{d_p}$, a vector of d_p parameters that can be learnt from data, as shown below.

To evolve the hidden state $\boldsymbol{h}(t)$ from timestamp t_0 to t_1, we write the following initial value problem (IVP):

$$\boldsymbol{h}(t_1) = \boldsymbol{h}(t_0) + \int_{t_0}^{t_1} f_d\left(\boldsymbol{h}(t); \boldsymbol{\theta}_d\right)\, dt\,, \tag{2}$$

where $\boldsymbol{h}(t_0)$ is the initial hidden state at t_0, to be estimated from the data as shown in Section 3.3. In general, the IVP in Eq. (2) is solved numerically at all timestamps t_k using an IVPSolve routine:

$$\boldsymbol{h}(t_k) = \texttt{IVPSolve}\left(f_d, \boldsymbol{\theta}_d, \boldsymbol{h}(t_{k-1}), [t_{k-1}, t_k]\right) + \varepsilon\,, \tag{3}$$

where ε is the numerical approximation error. In this work, we use adaptive-step Runge-Kutta 4(5) (Dormand and Prince, 1980) to perform the numerical integrations.

In Section 3.3, we develop a model that learns to predict a diagnostic code at time t from the patient state $\boldsymbol{h}(t)$, which is also learnt from the data. The algorithm therefore uses the data (i.e., the observed diagnostic codes collected at timestamps t_k) to learn the parameters $\boldsymbol{\theta}_d^*$ of the dynamics function in Eq. (1) by minimising a loss function over the whole dataset with terms

$$\mathcal{L}(\boldsymbol{h}(t_k)) = \mathcal{L}\left(\texttt{IVPSolve}(f_d, \boldsymbol{\theta}_d, \boldsymbol{h}(t_{k-1}), [t_{k-1}, t_k])\right).$$

The loss function is minimised using stochastic gradient descent (SGD), which requires computing the gradient $\nabla_{\boldsymbol{\theta}_d}\mathcal{L}$ using deep learning libraries (e.g., `PyTorch` or `JAX`) with reverse-mode automatic differentiation (AD) (Baydin et al., 2018). AD efficiently constructs a computational graph linking the inputs of the function with all the intermediate operations leading to the output of the function, on which backpropagation can then be applied to obtain the gradient. However, when applied to a loss function that depends on an ODE solver (3), this requires storing all the intermediate values of the `IVPSolve` iterations, which may cause problems with computer memory. Fortunately, an efficient algorithm (Chen et al., 2018) based on the *adjoint method* computes the gradient using a time-backward IVP from t_k to t_{k-1} and only requires storing the final value $\boldsymbol{h}(t_k)$.

3.2. Clinical Embeddings module

Central to our tasks here is to learn highly informative embeddings for the clinical codes that conform our data. Let us consider the set of all possible clinical codes, $\mathcal{C}$, with cardinality $C = |\mathcal{C}|$. The aim of the embedding is to find a transformation that represents a set of clinical codes $c \subseteq \mathcal{C}$ through a fixed-size vector $\boldsymbol{g} \in \mathbb{R}^{d_e}$, which provides a compact representation that retains high information content about the clinical codes. We start with a multi-hot encoding of the set of clinical codes c as a binary vector $\boldsymbol{v} \in \{0,1\}^C$ with coordinates equal to 1 for the codes present in c and coordinates equal to 0 elsewhere. This large, sparse, binary vector $\boldsymbol{v}$ is then transformed into a compact representation $\boldsymbol{g}$, i.e., an embedding is obtained. There is a large array of embedding techniques in the literature. We now describe briefly the two methods used in this work.

Simple matrix embedding. A straightforward approach is to transform $\boldsymbol{v} \in \{0,1\}^C$ into a compact representation $\boldsymbol{g} \in \mathbb{R}^{d_e}$ by using an affine map $f_M : \{0,1\}^C \mapsto \mathbb{R}^{d_e}$:

$$\boldsymbol{g} = f_M(\boldsymbol{v}; \boldsymbol{\theta}_M) = \boldsymbol{W}_M\, \boldsymbol{v} + \boldsymbol{b}_M \tag{4}$$

where $\boldsymbol{W}_M \in \mathbb{R}^{d_e \times C}$ is a learnable matrix, $\boldsymbol{b}_M \in \mathbb{R}^{d_e}$ is a learnable vector, and $\boldsymbol{\theta}_M$ denotes the concatenation $[\boldsymbol{W}_M, \boldsymbol{b}_M]$.

GRaph-based Attention Model (`GRAM`) embedding. Several of the most widely used clinical coding schemes, such as *SNOMED-CT* and *ICD*, can be organised into a medical ontology, i.e., a hierarchy of codes where high level (parent) codes represent abstract medical concepts and, as we go down the hierarchy, children codes represent increasingly detailed and precisely described medical concepts. Mathematically, this hierarchy is represented by a directed acyclic graph (DAG), which we denote as $\mathcal{G}$. Choi et. al (Choi et al., 2017) proposed the `GRAM` algorithm, where each medical concept is represented as a convex combination of vectors of 'basic embeddings' of the code itself and all of its ancestors in

$\mathcal{G}$. This approach enriches the embedding by retaining information about the ancestry of the code, and mitigates the risk of overfitting rare medical concepts in the training dataset. In summary, the method obtains a set of basic embeddings $\boldsymbol{\theta}_E = [\boldsymbol{e}_1; \ldots; \boldsymbol{e}_C] \in \mathbb{R}^{C \times d_e}$, one for each medical code $c_i \in \mathcal{C}$. (To initialise the learning process, the basic embeddings $\boldsymbol{\theta}_E$ can be randomly initialised, or obtained with *GloVe* (Pennington et al., 2014).) The final embedding of a medical code c_i is given by the convex combination:

$$\boldsymbol{g}_i = \sum_{j \in \mathcal{A}(i)} \alpha_{ij} \boldsymbol{e}_j, \quad \text{with} \quad \sum_{j \in \mathcal{A}(i)} \alpha_{ij} = 1, \quad \alpha_{ij} \geq 0, \tag{5}$$

where $\mathcal{A}(i)$ is the set of coordinates for the union of the medical concept c_i and all of its ancestors in $\mathcal{G}$. The weights $\alpha_{ij} \in \mathbb{R}$ are computed with the *softmax function*:

$$\alpha_{ij} = \frac{\exp\left(f_R(\boldsymbol{e}_i, \boldsymbol{e}_j)\right)}{\sum_{k \in \mathcal{A}(i)} \exp\left(f_R(\boldsymbol{e}_i, \boldsymbol{e}_k)\right)}, \tag{6}$$

where the *self-attention* $f_R : \mathbb{R}^{d_e} \times \mathbb{R}^{d_e} \mapsto \mathbb{R}$ estimates the relatedness between the embeddings. Originally, the authors implemented the function f_R as a multilayer perceptron (MLP) with a single hidden layer of size ℓ:

$$f_R(\boldsymbol{e}_i, \boldsymbol{e}_j; \boldsymbol{\theta}_R) = \boldsymbol{u}_R^T \tanh\left(\boldsymbol{W}_R \begin{bmatrix} \boldsymbol{e}_i \\ \boldsymbol{e}_j \end{bmatrix} + \boldsymbol{b}_R\right), \tag{7}$$

where $\boldsymbol{W}_R \in \mathbb{R}^{\ell \times 2d_e}$, $\boldsymbol{b}_R \in \mathbb{R}^{\ell}$ and $\boldsymbol{u}_R \in \mathbb{R}^{\ell}$. Here, $\boldsymbol{\theta}_R$ denotes $[\boldsymbol{W}_R, \boldsymbol{b}_R, \boldsymbol{u}_R]$. Note that the vertical concatenation $[\boldsymbol{e}_i; \boldsymbol{e}_j]$ preserves the child-ancestor order. In this work, we also used a less parametrised variant suggested in (Kim et al., 2021, Appendix F.2) with stability guarantees:

$$f_r(\boldsymbol{e}_i, \boldsymbol{e}_j; \boldsymbol{\theta}_r) = \exp\left(-\frac{||(\boldsymbol{e}_i - \boldsymbol{e}_j)^T \boldsymbol{\theta}_r||_2^2}{\sqrt{\ell}}\right), \tag{8}$$

where $\boldsymbol{\theta}_r \in \mathbb{R}^{\ell \times d_e}$.

Given basic embeddings $\boldsymbol{\theta}_E$ and attention parameters $\boldsymbol{\theta}_R$ (or $\boldsymbol{\theta}_r$ if (8) is used), the embedding of the codes $c_i \in \mathcal{C}$ is given by the matrix

$$\boldsymbol{G}_{(\boldsymbol{\theta_E}, \boldsymbol{\theta_R})} = [\boldsymbol{g}_1; \ldots; \boldsymbol{g}_C] \in \mathbb{R}^{C \times d_e}.$$

To transform the multi-hot binary vector $\boldsymbol{v} \in \{0, 1\}^C$, representing the set of multiple medical concepts $c \subseteq \mathcal{C}$, into the embedding space we just apply the transformation

$$\boldsymbol{g} = f_G(\boldsymbol{v}; \boldsymbol{\theta_E}, \boldsymbol{\theta}_R) = \tanh\left(\boldsymbol{v}\, \boldsymbol{G}_{(\boldsymbol{\theta_E}, \boldsymbol{\theta_R})}\right) \tag{9}$$

Remark. ICE-NODE does not assume a particular embedding method; hence $\boldsymbol{g}$ (i.e., the embedding of $\boldsymbol{v}$) can be computed via (4) or (9). The particular choice in the numerical experiments is made explicit in Section 4.

3.3. The ICE-NODE model

ICE-NODE is the architecture that we propose to model the timestamped clinical codes contained in EHRs using neural ODEs coupled with a clinical code embedding module to predict diagnostic codes at a given future timestamp t_f.

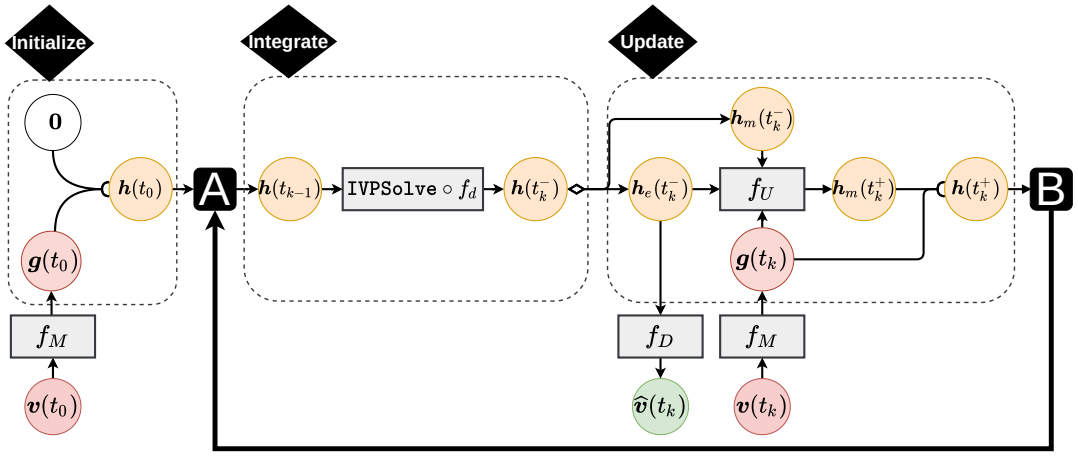


Figure 2: **Schematic representation of the ICE-NODE framework to model patient histories of timestamped clinical concepts, as recorded in their EHR.** The health state $\boldsymbol{h}(t_{k-1})$ is *integrated* with an ODE solver and learnable dynamics function f_d to obtain a future version $\boldsymbol{h}(t_k^-)$, i.e. just before time t_k. The memory state $\boldsymbol{h}_m(t_k^-)$ is then *updated* to accommodate the new information at time t_k by applying an update transformation f_U, which incorporates the memory state itself $\boldsymbol{h}_m(t_k^-)$, the integrated embedding state $\boldsymbol{h}_e(t_k^-)$, and the embeddings of the clinical concepts $\boldsymbol{g}(t_k)$ at the timestamp t_k. This 'Integrate-Update' cycle is repeated for subsequent timestamps.

Notation. Each patient i is represented as a sequence of timestamped clinical codes $\{\big(t_k, c(i,t_k)\big)\}_{t_k \in \mathcal{T}(i)}$, where t_k is the k-th timestamp in the temporally-ordered set of timestamps $\mathcal{T}(i)$ in the EHR of patient i, and $c(i,t_k) \subseteq \mathcal{C}$ is the set of medical concepts present at t_k for this particular patient. To simplify notation, we will drop the patient index i when suitable. The binary (multi-hot) representation of $\{\big(t_k, c(t_k)\big)\}_{t_k \in \mathcal{T}}$ is denoted as $\{\big(t_k, \boldsymbol{v}(t_k)\big)\}_{t_k \in \mathcal{T}}$. The embedding module transforms each set of clinical codes $c(t_k)$ into their embeddings $\boldsymbol{g}(t_k)$, and hence we obtain the set of timestamped embeddings $\{\big(t_k, \boldsymbol{g}(t_k)\big)\}_{t_k \in \mathcal{T}}$. The d_h-dimensional hidden state for the patient $\boldsymbol{h}(t) = [\boldsymbol{h}_m(t); \boldsymbol{h}_e(t)]$ is structured to consist of two components: a memory state $\boldsymbol{h}_m(t) \in \mathbb{R}^{d_m}$ and an embedding state $\boldsymbol{h}_e(t) \in \mathbb{R}^{d_e}$, such that $d_m + d_e = d_h$.

3.3.1. ICE-NODE Architecture

The initial hidden state $\boldsymbol{h}(t_0) \in \mathbb{R}^{d_h}$ at the first timestamp t_0 is given by $\boldsymbol{h}(t_0) = [\boldsymbol{h}_m(t_0); \boldsymbol{h}_e(t_0)] = [\boldsymbol{0}; \boldsymbol{g}(t_0)]$, where the memory state is initialised with zeros and the embedding state is initialised with the embedding of the set of medical codes observed at t_0.

We next evolve $\boldsymbol{h}(t_0)$ by integration over $[t_0, t_1)$, i.e., to a time t_1^- just before t_1:

$$\boldsymbol{h}(t_1^-) = \begin{bmatrix} \boldsymbol{h}_m(t_1^-) \\ \boldsymbol{h}_e(t_1^-) \end{bmatrix} = \texttt{IVPSolve}(f_d, \boldsymbol{\theta}_d, \boldsymbol{h}(t_0), [t_0, t_1]). \tag{10}$$

To this evolved state, we apply a decoding function $f_D : \mathbb{R}^{d_e} \mapsto [0,1]^C$ to obtain the predicted clinical codes at time t_1:

$$\widehat{\boldsymbol{v}}(t_1) = f_D(\boldsymbol{h}_e(t_1^-); \boldsymbol{\theta}_D). \tag{11}$$

This prediction can then be scored against the observed codes $\boldsymbol{v}(t_1)$ in the dataset.

To accommodate the new information gathered at each timestamp into $\boldsymbol{h}(t)$, we update the memory state:

$$\boldsymbol{h}_m(t_1^+) = f_U(\boldsymbol{h}_m(t_1^-), \boldsymbol{h}_e(t_1^-), \boldsymbol{g}(t_1); \boldsymbol{\theta}_U), \tag{12}$$

where t_1^+ is a time just after the observation timestamp t_1, and $f_U : \mathbb{R}^{d_m} \times \mathbb{R}^{d_e} \times \mathbb{R}^{d_e} \mapsto \mathbb{R}^{d_m}$ adjusts the memory state $\boldsymbol{h}_m(t_1^-)$ to incorporate the new information at t_1. The steps (10)-(11)-(12) are then repeated for all remaining timestamps in $\mathcal{T}$.

After fitting all the timestamps in the history of a patient, we apply a final integration (10) from the last timestamp in $\mathcal{T}$ until the future timestamp t_f, to obtain the hidden health state $\boldsymbol{h}(t_f)$. This is followed by the decoding step (11) to produce the diagnostic predictions $\widehat{\boldsymbol{v}}(t_f)$.

3.3.2. Training ICE-NODE

The training is carried out by minimising the following loss function, which includes the cross-entropy comparing the predictions $\widehat{\boldsymbol{v}}(t_k)$ and the observations $\boldsymbol{v}(t_k)$ of the codes at timestamps $t_k \in \{t_1, \ldots, t_{|\mathcal{T}|-1}\}$ and a regularisation term that enforces the smoothness of the dynamics:

$$\mathcal{L} = \frac{1}{|\mathcal{T}| - 1} \sum_{k=1}^{|\mathcal{T}-1|} \mathcal{L}_{\boldsymbol{v}}(t_k) + \alpha_K \mathcal{R}_K(t_k; \boldsymbol{\theta}_d), \tag{13}$$

where the cross-entropies are:

$$\mathcal{L}_{\boldsymbol{v}}(t_k) = \boldsymbol{v}(t_k)^T \log\left(\widehat{\boldsymbol{v}}(t_k)\right) + \left(\mathbf{1} - \boldsymbol{v}(t_k)\right)^T \log\left(\mathbf{1} - \widehat{\boldsymbol{v}}(t_k)\right), \tag{14}$$

and the regularisation term is given by:

$$\mathcal{R}_K(t_k; \boldsymbol{\theta}_d) = \int_{t_{k-1}}^{t_k} \left\| \frac{\mathrm{d}^K \boldsymbol{h}(t)}{\mathrm{d}t^K} \right\|_2^2 dt\,. \tag{15}$$

The regularisation term $\mathcal{R}_K$ (Kelly et al., 2020), which is inspired by insights from adaptive-step solvers, diminishes the magnitude of the higher order derivatives and increases both trajectory smoothness and step size. This allows us to efficiently concentrate on a smaller subspace of dynamic trajectories. In general, the hyperparameter K must be no larger than m, the order of the ODE solver (Runge-Kutta with $m = 4$ here), but since we further assume that the health state $\boldsymbol{h}(t)$ has vanishing derivatives above second-order, we set $K = 3$. Finally, α_K is the hyperparameter that sets the relative weight of the regularisation term (15). It is set here to a large value of $\alpha_K = 1000$ to enforce vanishing derivatives of order $K \leq 3$.

For each training iteration, we randomly sample (with replacement) a fixed number of patients B from the total of N patients in the training set. The loss (13) is then averaged over the B sampled patients and the gradients are computed, which are then passed to the optimiser to update the parameters. Here, we use the *Adam* optimiser (Kingma and Ba, 2014), which achieves remarkably better convergence than both SGD and *Adamax* (Kingma and Ba, 2014). We find that convergence is improved by using two independent learning rates for the *Adam* optimiser: one for the dynamics parameters, and a separate one for the

other parameters of ICE-NODE. Appendix B describes in detail the training settings. We also describe in Appendix B the strategy for hyperparameter optimisation, including the use of the optuna framework (Akiba et al., 2019), to settle on an optimal choice of functions and their configuration for the dynamics function f_d (10), the decoding function f_D (11), and the update function f_U (12).

4. Experiments

4.1. Description of the datasets

MIMIC-III (Johnson et al., 2016a,b): Medical Information Mart for Intensive Care III (MIMIC-III) is a publicly available dataset that contains EHRs for over 46.5K patients who had at least one admission at Beth Israel Deaconess Medical Center (BIDMC) between 2001 and 2012. For each admission, the patient can be discharged on the same day or stay for a longer time, but in all cases MIMIC-III stores all the diagnosis codes related to the entire stay and links them to the discharge timestamp. This manner of information collection adds a degree of uncertainty to the timestamps associated with long stays; if a patient stays for two months, they will be discharged with a set of diagnosis codes, but this does not indicate precisely when the diagnoses were made during those two months. We therefore restrict our study to patients (i) who had at least two admissions, and (ii) whose admissions were all at most 2 weeks long. With these restrictions, our dataset includes 4.4K patients. MIMIC-III uses the *ICD-9* coding scheme to store the diagnosis codes, which we convert into the Clinical Classifications Software (CCS)[3] multi-level (i.e. hierarchical) scheme. This conversion reduces substantially the complexity of the data—whereas *ICD-9* provides over 15K diagnosis codes, CCS contains 589 diagnosis codes. When we use GRAM embeddings, we thus employ the corresponding DAG $\mathcal{G}$ of the CCS hierarchical scheme for the analysis of the set of clinical codes $\mathcal{C}$.

MIMIC-IV (Johnson et al., 2021): Medical Information Mart for Intensive Care IV (MIMIC-IV) is another publicly available dataset that contains EHRs for over 256K patients who had a critical care or emergency admission in BIDMC between 2008 and 2019. After applying the same rules for patient selection as for MIMIC-III, we end up having EHRs for over 70K patients. MIMIC-IV uses the *ICD-10* coding scheme, which we map to *ICD-9*[4] and eventually to the CCS coding scheme, so that models trained on MIMIC-III can be tested on MIMIC-IV and vice versa.

Summary: Table 1 presents descriptive statistics of the two datasets constructed from MIMIC-III and MIMIC-IV. For both datasets, we randomly split patients into training:validation:testing with ratios $0.70 : 0.15 : 0.15$. Appendix A includes consort diagrams for the extraction of the training-validation-test splits from both datasets.

3. https://www.hcup-us.ahrq.gov/toolssoftware/ccs/ccs.jsp
4. https://www.nber.org/research/data/icd-9-cm-and-icd-10-cm-and-icd-10-pcs-crosswalk-or-general-equivalence-mappings

Table 1: Summary statistics for the two datasets used in our experiments.

Statistics	MIMIC-III	MIMIC-IV
No. of patients (N_p)	4,385	70,027
No. of admissions (N_a)	10,954	265,637
Avg. admissions per patient (N_a/N_p)	2.47	3.66
Avg. ($\pm$std.) weeks between admissions	66.1 ($\pm$97.3)	53.8 ($\pm$85.8)
Avg. ($\pm$std.) days of stay	5.9 ($\pm$3.5)	3.2 ($\pm$2.8)
Avg. no. of ICD-9 diagnostic discharge codes	11.65	11.46
Avg. no. of CCS diagnostic discharge codes	10.84	9.75

4.2. Methods for benchmarking

Our numerical experiments have been compared and benchmarked against state-of-the-art baseline methods. In most cases, we have implemented a version that uses matrix embeddings (4) and a version that uses `GRAM` embeddings (9). The latter versions are denoted by adding the letter `/G` to the corresponding acronyms.

The baseline methods considered are:

`LogReg`: We implement a standard logistic regression with elastic net regularisation. This method takes as an input a binary vector encoding the occurrence of diagnostic codes in the past. We consider this model as a representative of models that do not exploit the sequential nor the full temporal information of the EHRs.

`RETAIN`: This method (Choi et al., 2016a) learns patient visits in reverse time order using an attention mechanism using the temporal ordering of the codes, but ignoring the irregular time intervals between them.

`GRU` & `GRU/G`: Based on the architecture developed by Choi et al. (2017), we have implemented two versions: the original one with `GRAM` embeddings, denoted `GRU/G`, and another one with matrix embeddings, denoted `GRU`. Again, this method only uses the time ordered information but not the full temporal information in the data.

These baseline methods are compared against our model in different versions:

`ICE-NODE` & `ICE-NODE/G`: We use our model, as developed above, with both matrix embeddings and `GRAM` embeddings, using the full temporal information available in the EHRs.

`ICE-NODE_UNIFORM` & `ICE-NODE_UNIFORM/G`: We have also considered a variant of `ICE-NODE` where we fix the intervals between consecutive timestamps to be one week in two versions: one with matrix embeddings, denoted `ICE-NODE_UNIFORM`, and one with `GRAM` embeddings, denoted `ICE-NODE_UNIFORM/G`. These versions ignore the irregularity of the temporal sampling, and just preserve the ordering. We use this variant particularly to assess whether a clinical code prediction is improved by incorporating the full information of the timestamps, or if using only the temporal ordering is sufficient.

4.3. Analysis of prediction performance

We analysed the different versions of our proposed model along with the baseline models through three experiments:

- **Experiment A**: Training and testing on MIMIC-III
- **Experiment B**: Training and testing on MIMIC-IV
- **Experiment C**: Training on MIMIC-IV followed by testing on the entire MIMIC-III

Evaluating the prediction performance of disease progression models with respect to a large number of clinical codes is a nontrivial problem. While one model can be the most competent in predicting a particular set of clinical codes, the same model can be outperformed by another model in predicting a different set of clinical codes. Typically, researchers often focus on a specific category of diseases, and compare the predictability between multiple models.

Alternatively, other evaluations are geared towards systematically evaluating all medical codes, yet controlling for their different frequency, since predicting very common medical codes can be less informative. Clinical codes are then partitioned into quantiles according to their frequency in the dataset, such that the model predictability can be estimated for different percentile ranges separately, from the infrequent to the most frequent codes. With this evaluation method, Choi et al. (2017) showed that their algorithm (denoted here as `GRU/G`) is specifically competent in predicting clinical codes that are observed rarely in the training data. We have applied this approach to evaluate the performance of the methods for Experiments A and B. We report the results in Table 2, where we show the averaged top-15 prediction accuracy (i.e., for each clinical code, we score 1 if it is correctly detected in the top-15 predictions by the model at each visit, and 0 otherwise). We find that `ICE-NODE` is particularly competent in predicting codes that are infrequent in the training dataset (i.e., in the 0-20 and 20-40 quantiles) while still performing well in more frequent codes.

As an additional measure of performance, we have also carried out a quantification of the *relative competency* at the code level using the area under the receiver operating characteristic curve (AUC) values achieved by the different methods and the DeLong test (DeLong et al., 1988). This test allows us to establish the statistical significance of the difference between the AUC values obtained for any given pair of models. Figure 3 summarises the results of this analysis for Experiments A-C. We find that most well-predicted codes are predicted well by all methods, yet different methods are more competent at predicting particular codes.

4.4. Time-continuous risk trajectories

Through its use of neural ODEs, `ICE-NODE` stands out from the baseline methods by a qualitative advantage; namely, the possibility of using the learnt hidden patient state $\boldsymbol{h}(t)$ to carry out time-continuous predictions of, e.g., disease risk. By sampling $\boldsymbol{h}(t)$ (10) at arbitrary times, we can apply the prediction function (11) to generate a risk trajectory for a selected patient history. Two examples of such trajectories for two patients in the datasets with conditions where `ICE-NODE` outperforms the other methods are shown in Figure 4(a), and further examples can be seen in Appendix D.

Table 2: **Top-15 prediction accuracy for clinical codes according to their frequency in the training set.** The clinical codes are partitioned into five quantiles, according to their frequency in the training set. For each admission, we score the top-15 codes. The top performing method for each quantile group is highlighted in dark green and the second best method in light green.

(a) Accuracy of Experiment A (training and testing on MIMIC-III).

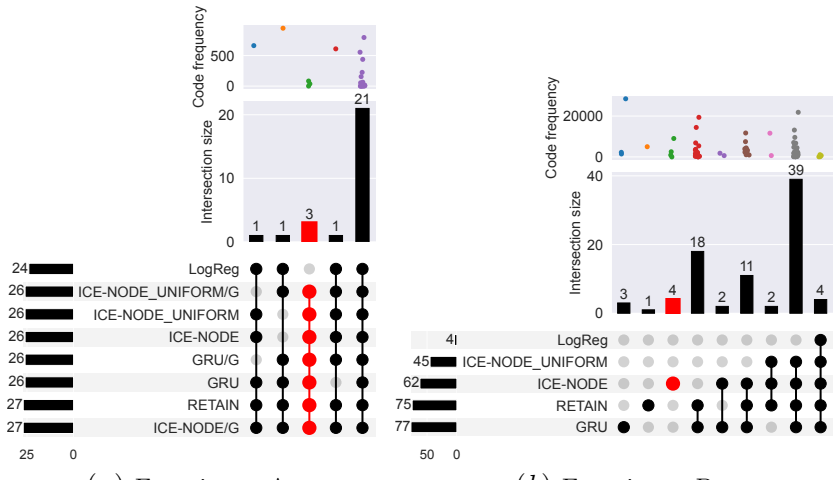
Model	Clinical codes frequency (quantile ranges)				
	0-20	20-40	40-60	60-80	80-100
LogReg	0.164	0.454	0.544	0.683	0.777
RETAIN	0.208	0.415	0.579	0.778	0.926
GRU	0.214	0.400	0.556	0.768	0.900
GRU/G	0.204	0.403	0.546	0.778	0.906
ICE-NODE_UNIFORM	0.211	0.422	0.566	0.786	0.909
ICE-NODE_UNIFORM/G	0.206	0.419	0.565	0.782	0.907
ICE-NODE	0.219	0.425	0.578	0.782	0.902
ICE-NODE/G	0.205	0.421	0.569	0.782	0.907

(b) Accuracy of Experiment B (training and testing on MIMIC-IV).

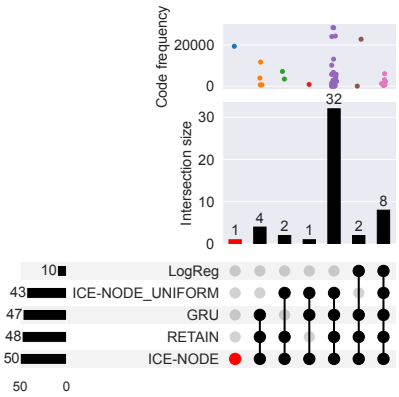
Model	Clinical codes frequency (quantile ranges)				
	0-20	20-40	40-60	60-80	80-100
LogReg	0.019	0.347	0.606	0.869	0.943
RETAIN	0.362	0.483	0.621	0.806	0.928
GRU	0.355	0.491	0.633	0.808	0.917
GRU/G	0.357	0.491	0.628	0.809	0.921
ICE-NODE_UNIFORM	0.376	0.486	0.598	0.773	0.904
ICE-NODE_UNIFORM/G	0.370	0.485	0.593	0.779	0.903
ICE-NODE	0.375	0.493	0.605	0.774	0.904
ICE-NODE/G	0.374	0.491	0.601	0.776	0.906

5. Discussion

The results in Table 2 suggest significant competency of the different variants of ICE-NODE, especially in predicting clinical codes that are rare in the training data. We also find that using GRAM embeddings for ICE-NODE does not achieve the hoped-for improvement over matrix embeddings in our current analysis. However, this initial conclusion might be due to the use of the simple CCS hierarchical coding scheme with its computationally manageable vocabulary (589 diagnosis codes). However, the use of such a simple coding scheme might have resulted in a loss of information. Therefore, ICE-NODE/G might still add value if more complex hierarchical coding schemes, such as *ICD-9* (15K diagnosis codes), *ICD-10* (70K diagnosis codes), or *SNOMED-CT* (383K diagnosis and procedure codes), were to be used. Employing such detailed coding schemes could enable the uncovering of new disease prognosis patterns, offering new opportunities for the predictive models to improve



(a) Experiment A (b) Experiment B



(c) Experiment C

Figure 3: **(a)-(c) Relative prediction of medical codes for the different models for Experiments A-C.** AUC values of all clinical codes are computed for each model and a DeLong test (*p-value*=0.01) is carried out to establish the statistical significance of the difference in prediction between each pair of models. Clinical codes predicted with AUC>0.9 by at least one model are assigned to the model with maximum AUC and to any other model with no significant difference (according to DeLong test). Frequencies of clinical codes in the training set are shown at the top. (a) In Experiment A, 27 codes are predicted with an AUC>0.9 by at least one model: 21 of those codes are predicted equivalently well by all 8 models, whereas the rest are predicted differently by some models. (b) In Experiment B, 84 codes are predicted with an AUC>0.9 by at least one model; of those, only 4 codes are predicted well by `LogReg`, whereas 39 codes are predicted equivalently well by the four models that incorporate temporal or sequence information. Some codes are predicted well by several models (39/2/11/2/18); other codes are predicted well only by one model (4/1/3). (c) In Experiment C, 50 codes are predicted with an AUC>0.9 by at least one model: 32 codes are predicted equivalently well by all models (but not by `LogReg`). There are 4 codes in (b) ('*Pulm hart dx*', '*Other ear dx*', '*Early labor*', '*Forceps del*'), and 1 code in (c) ('*Ac renal fail*') that are only predicted well by `ICE-NODE` (marked in red). The AUC values for the codes in red are shown in Fig. 7 (Appendix C).

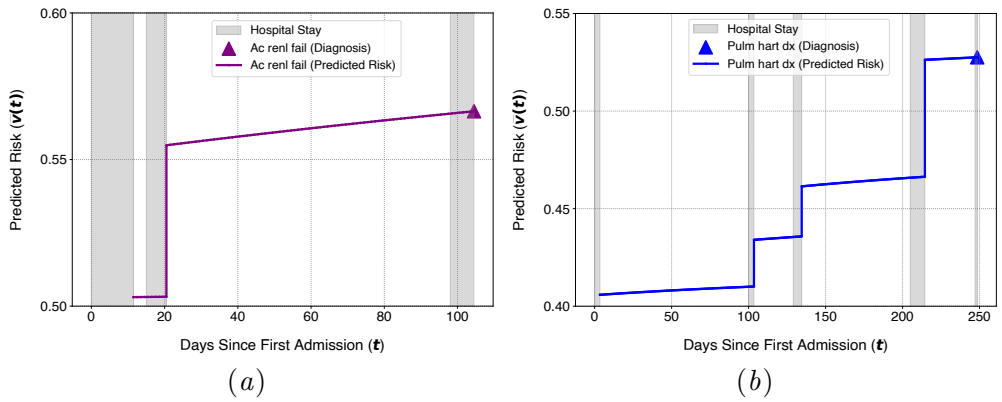


Figure 4: Two predicted risk trajectories for patients in the test set: (a) Predicted risk trajectory of code '*Ac renal fail*' for patient with `subject_id=3600` in the test set of MIMIC-III. The history of this patient consists of three hospital stays (i.e., admissions-discharges) with diagnosis of '*Ac renal fail*' at the third discharge. After the first discharge, `ICE-NODE` predicts a risk for this diagnosis with a probability slightly higher than 0.5. After the second discharge, the risk has jumped to above 0.55 with continuously increasing risk throughout roughly 80 days, before the last hospital stay; (b) Predicted risk trajectory of '*Pulm hart dx*' for the patient with `subject_id=13286711` in the test set of MIMIC-IV. In this case, the history consists of five hospital stays with diagnosis of '*Pulm hart dx*' at the fifth and last hospital stay. The risk for this condition kept increasing since the first discharge despite negative reporting for this code throughout.

their representation learning and hence their prediction performance. From this point, our discussion and subsequent analysis will focus only on models with matrix embeddings.

We have used `ICE-NODE_UNIFORM` as a variant of `ICE-NODE` where all time-intervals between consecutive timestamps are fixed to one week, as a mean to evaluate within our framework the impact of incorporating the irregular intervals in the modeling, beyond using sequential information alone. Figure 3 shows that there is no clinical code predicted by `ICE-NODE_UNIFORM` with a significantly higher AUC than `ICE-NODE` in any of the three experiments (A-C), whereas `ICE-NODE` significantly outperforms `ICE-NODE_UNIFORM` in 17 clinical codes in Experiment B and 7 clinical codes in Experiment C. The results in Table 2 and Fig. 3 show that the inclusion of temporal information (sequential only or full timestamps) is strongly advantageous, as seen by the reduced performance of `LogReg`. Furthermore, the analysis of Experiment B in Fig. 3(b) shows that many clinical codes (18 out of 84) are predicted better by both `GRU` and `RETAIN` than by `ICE-NODE`. This suggests that for these clinical codes, the temporal-order information is sufficient without incorporating the irregular intervals, which, in turn, may have added noisy information in the learning of `ICE-NODE`, thus undermining its performance.

The results for Experiment C (training on MIMIC-IV and predicting on MIMIC-III as a test set) in relation to Experiment B (training and testing on splits of MIMIC-IV) in Fig. 3 also suggest better generalisation properties for `ICE-NODE` across datasets. Specifically, `ICE-NODE` is competent in predicting 62 clinical codes in Experiment B, and this number is reduced by 12 in Experiment C when predicting across datasets. For `RETAIN` and `GRU`, this number is reduced by 27 and 30, respectively.

Finally, we have shown that ICE-NODE produces time-continuous risk trajectories for each patient and each clinical code starting from the initial discharge from hospital. This feature may enable the use of the framework to gain data-informed insights to questions such as "*How will the risk of renal failure evolve within two weeks from now?*" or "*Will the patient undergo an alarming risk of renal failure? and when will the risk exceed 0.5?*". Appendix D shows additional exemplars of such predicted trajectories.

Future Work In relation to this last point, the user of ICE-NODE might be interested in studying time-continuous trajectories of multiple continuous simultaneously, to gain insights on how multiple medical conditions co-develop with time, and whether causal or correlational relationships might exist between conditions or other confounding conditions might explain these developments. This opens a future research direction to investigate whether the predicted, time-continuous trajectories can be reliable for causal discovery and/or causal inference tasks. This task becomes more challenging as the number of conditions studied increases but could help generate hypotheses on how medical conditions develop and interact supported by ICE-NODE data-based predictions.

Limitations Using neural ODEs can be sensitive to noisy information compared to the baseline models. This could explain why 18 out of 84 clinical codes in Experiment B are better predicted with two of the baseline models. The noise sources that can limit ICE-NODE learning capacity can be attributed to two reasons. First, for some clinical codes, the temporal-order could be sufficient for predictability, and incorporating the time-intervals is unnecessary and a source of variability. Second, mislabeled codes (false positives) and/or unreported codes (false negatives) in the ground-truth can misguide the prediction of the clinical codes trajectories. Since neural ODEs integrate trajectories based on noisy initial conditions, the sensitivity to noise increases with a longer time-interval until the next hospital discharge. Another limitation in this work concerns the training itself. We have observed that training ICE-NODE improves remarkably when using two learning rates (one for the parameters of the dynamics and another for the rest of the parameters). This burdens the task of searching for optimal hyperparameters, hence further improvements for ICE-NODE could be possible. At present, we do not have a sound explanation for this behaviour in the training.

When this research was finished, we learned of research by Peng et al. (2021) using neural ODEs to predict clinical codes of future visits. Even though that work exploits the temporal dimension, we discuss in Appendix E their design which is pragmatically optimised for predicting future codes without producing time-continuous trajectories for the patient state, in contrast to our work.

6. Conclusions

We have presented a new disease progression model (ICE-NODE) that learns from timestamped clinical codes using neural ODEs, and fully exploits the time dimension by incorporating the irregular time-intervals between the timestamps, as opposed to only exploiting the temporal-order. We have provided performance analyses by applying ICE-NODE (and the baselines) on MIMIC-III and MIMIC-IV. These analyses have identified a set of clinical codes that are predicted with improved performance using ICE-NODE, such as 'Ac renal fail'

and 'Pulm hart dx'. Our analyses also show better generalisation properties for ICE-NODE across datasets. We have finally discussed the implications of obtaining time-continuous risk trajectories for diseases, to improve the understanding of disease development and interactions. We have provided pointers for future directions of research on exploiting these trajectories to reveal potential causal relations between the medical conditions as they evolve with time.

Acknowledgments

This work was supported by the UKRI CDT in AI for Healthcare https://ai4health.io (Grant No. EP/S023283/1) and the NIHR Imperial Biomedical Research Centre (BRC), Imperial Clinical Analytics, Research and Evaluation (iCARE). MB acknowledges support from the EPSRC under grant EP/N014529/1 funding the EPSRC Centre for Mathematics of Precision Healthcare at Imperial.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

Melina Arnold, Mark J Rutherford, Aude Bardot, Jacques Ferlay, Therese ML Andersson, Tor Åge Myklebust, Hanna Tervonen, Vicky Thursfield, David Ransom, Lorraine Shack, et al. Progress in cancer survival, mortality, and incidence in seven high-income countries 1995–2014 (icbp survmark-2): a population-based study. *The Lancet Oncology*, 20(11): 1493–1505, 2019.

Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic Differentiation in Machine Learning: a Survey. *Journal of machine learning research*, 18, 2018.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011. URL https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Edward De Brouwer, Jaak Simm, Adam Arany, and Yves Moreau. GRU-ODE-Bayes: Continuous modeling of sporadically-observed time series. 5 2019. URL https://arxiv.org/abs/1905.12374v2.

Xiangrui Cai, Jinyang Gao, Kee Yuan Ngiam, Beng Chin Ooi, Ying Zhang, and Xiaojie Yuan. Medical Concept Embedding with Time-Aware Attention. *IJCAI International Joint Conference on Artificial Intelligence*, 2018-July:3984–3990, 6 2018. URL http://arxiv.org/abs/1806.02873.

Cancer Research UK. Cancer statistics for the UK, 2015. URL https://www.cancerresearchuk.org/health-professional/cancer-statistics-for-the-uk.

Ricky T Q Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural Ordinary Differential Equations. In S Bengio, H Wallach, H Larochelle, K Grauman, N Cesa-Bianchi, and R Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, pages 6571–6583. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf.

Daniel R Cherry, Qinyu Chen, and James Don Murphy. A novel prediction model to identify patients with early-stage pancreatic cancer. *Journal of Clinical Oncology*, 38(15_suppl): e16801–e16801, 5 2020. ISSN 0732-183X. doi: 10.1200/jco.2020.38.15{_}suppl.e16801.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Edward Choi, Mohammad Taha Bahadori, Joshua A. Kulas, Andy Schuetz, Walter F. Stewart, and Jimeng Sun. RETAIN: An Interpretable Predictive Model for Healthcare using Reverse Time Attention Mechanism. *Advances in Neural Information Processing Systems*, pages 3512–3520, 8 2016a. URL http://arxiv.org/abs/1608.05745.

Edward Choi, Mohammad Taha Bahadori, Elizabeth Searles, Catherine Coffey, Michael Thompson, James Bost, Javier Tejedor-Sojo, and Jimeng Sun. Multi-layer representation learning for medical concepts. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 13-17-August-2016, pages 1495–1504, New York, NY, USA, 8 2016b. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939823. URL https://dl.acm.org/doi/10.1145/2939672.2939823.

Edward Choi, Mohammad Taha Bahadori, Le Song, Walter F. Stewart, and Jimeng Sun. GRAM: Graph-based attention model for healthcare representation learning. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume Part F129685, pages 787–795, New York, NY, USA, 8 2017. Association for Computing Machinery. ISBN 9781450348874. doi: 10.1145/3097983.3098126. URL https://dl.acm.org/doi/10.1145/3097983.3098126.

Edward Choi, Cao Xiao, Walter F. Stewart, and Jimeng Sun. MiME: Multilevel Medical Embedding of Electronic Health Records for Predictive Healthcare. *Advances in Neural Information Processing Systems*, 2018-December:4547–4557, 10 2018. URL http://arxiv.org/abs/1810.09593.

Edward Choi, Zhen Xu, Yujia Li, Michael Dusenberry, Gerardo Flores, Emily Xue, and Andrew Dai. Learning the Graphical Structure of Electronic Health Records with Graph Convolutional Transformer. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(01):606–613, 4 2020. ISSN 2374-3468. doi: 10.1609/aaai.v34i01.5400. URL https://aaai.org/ojs/index.php/AAAI/article/view/5400.

Elizabeth R DeLong, David M DeLong, and Daniel L Clarke-Pearson. Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach. *Biometrics*, pages 837–845, 1988.

John R Dormand and Peter J Prince. A family of embedded runge-kutta formulae. *Journal of computational and applied mathematics*, 6(1):19–26, 1980.

Jacques Ferlay, Murielle Colombet, Isabelle Soerjomataram, Donald M Parkin, Marion Piñeros, Ariana Znaor, and Freddie Bray. Cancer statistics for the year 2020: An overview. *International Journal of Cancer*, 149(4):778–789, 2021.

Patrizia Ferroni, Fabio M. Zanzotto, Silvia Riondino, Noemi Scarpato, Fiorella Guadagni, and Mario Roselli. Breast cancer prognosis using a machine learning approach. *Cancers*, 11(3), 3 2019. ISSN 20726694. doi: 10.3390/cancers11030328. URL /pmc/articles/PMC6468737/?report=abstracthttps://www.ncbi.nlm.nih.gov/pmc/articles/PMC6468737/.

Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *circulation*, 101(23):e215–e220, 2000.

Alistair Johnson, Tom Pollard, and Roger Mark. Mimic-iii clinical database (version 1.4), 2016a. URL https://doi.org/10.13026/C2XW26.

Alistair Johnson, Lucas Bulgarelli, Tom Pollard, Steven Horng, Leo Anthony Celi, and Roger Mark. Mimic-iv (version 1.0), 2021. URL https://doi.org/10.13026/s6n6-xd98.

Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-Wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9, 2016b.

Jacob Kelly, Jesse Bettencourt, Matthew James Johnson, and David Duvenaud. Learning Differential Equations that are Easy to Solve. 7 2020. URL https://arxiv.org/abs/2007.04504v2.

Patrick Kidger. On neural differential equations. *CoRR*, abs/2202.02435, 2022. URL https://arxiv.org/abs/2202.02435.

Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural Controlled Differential Equations for Irregular Time Series. *arXiv*, 5 2020. URL http://arxiv.org/abs/2005.08926.

Hyunjik Kim, George Papamakarios, and Andriy Mnih. The lipschitz constant of self-attention. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5562–5571. PMLR, 18–24 Jul 2021. URL http://proceedings.mlr.press/v139/kim21i.html.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR, 2014.

Xueping Peng, Guodong Long, Tao Shen, Sen Wang, Jing Jiang, and Michael Blumenstein. Temporal self-attention network for medical concept embedding. In *Proceedings - IEEE International Conference on Data Mining, ICDM*, volume 2019-November, pages 498–507. Institute of Electrical and Electronics Engineers Inc., 11 2019. ISBN 9781728146034. doi: 10.1109/ICDM.2019.00060.

Xueping Peng, Guodong Long, Tao Shen, Sen Wang, and Jing Jiang. Sequential diagnosis prediction with transformer and ontological representation. *arXiv preprint arXiv:2109.03069*, 2021.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL http://www.aclweb.org/anthology/D14-1162.

Yulia Rubanova, Ricky T.Q. Chen, and David Duvenaud. Latent ODEs for irregularly-sampled time series. 7 2019. URL https://arxiv.org/abs/1907.03907v1.

Martin G. Seneviratne, Juan M. Banda, James D. Brooks, Nigam H. Shah, and Tina M. Hernandez-Boussard. Identifying Cases of Metastatic Prostate Cancer Using Machine Learning on Electronic Health Records. *AMIA ... Annual Symposium proceedings. AMIA Symposium*, 2018:1498–1504, 2018. ISSN 1942597X. URL /pmc/articles/PMC6371284/?report=abstracthttps://www.ncbi.nlm.nih.gov/pmc/articles/PMC6371284/.

Ying Wang, Xiao Xu, Tao Jin, Xiang Li, Guotong Xie, and Jianmin Wang. Inpatient2Vec: Medical Representation Learning for Inpatients. In *Proceedings - 2019 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2019*, pages 1113–1117. Institute of Electrical and Electronics Engineers Inc., 11 2019. ISBN 9781728118673. doi: 10.1109/BIBM47256.2019.8983281.

Rebecka Weegar and Karin Sundström. Using machine learning for predicting cervical cancer from Swedish electronic health records by mining hierarchical representations. *PLOS ONE*, 15(8):e0237911, 8 2020. ISSN 1932-6203. doi: 10.1371/journal.pone.0237911. URL https://dx.plos.org/10.1371/journal.pone.0237911.

Cao Xiao, Edward Choi, and Jimeng Sun. Opportunities and challenges in developing deep learning models using electronic health records data: A systematic review. *Journal of the American Medical Informatics Association*, 25(10):1419–1428, 10 2018. ISSN 1527974X.

doi: 10.1093/jamia/ocy068. URL https://academic.oup.com/jamia/article/25/10/1419/5035024.

Yungang Xu, Zhiguo Zhou, Donghan Yang, Kai Wang, Yunyun Zhou, Li Wang, Qinghua Wang, Heming Bai, Cong Liu, Wei Liu, Yuanpeng Zhang, Lei Jiang, and Huji Xu. EHR2Vec: Representation Learning of Medical Concepts From Temporal Patterns of Clinical Notes Based on Self-Attention Mechanism. *Frontiers in Genetics — www.frontiersin.org*, 1:630, 2020. doi: 10.3389/fgene.2020.00630. URL www.frontiersin.org.

Jinghe Zhang, Kamran Kowsari, James H. Harrison, Jennifer M. Lobo, and Laura E. Barnes. Patient2Vec: A Personalized Interpretable Deep Representation of the Longitudinal Electronic Health Record. *IEEE Access*, 6:65333–65346, 2018. ISSN 21693536. doi: 10.1109/ACCESS.2018.2875677.

Guo-Bing Zhou, Jianxin Wu, Chen-Lin Zhang, and Zhi-Hua Zhou. Minimal gated unit for recurrent neural networks. *CoRR*, abs/1603.09420, 2016. URL http://arxiv.org/abs/1603.09420.

Weicheng Zhu and Narges Razavian. Graph Neural Network on Electronic Health Records for Predicting Alzheimer's Disease. *arXiv*, 12 2019. URL http://arxiv.org/abs/1912.03761.

Appendix A. Consort Diagrams

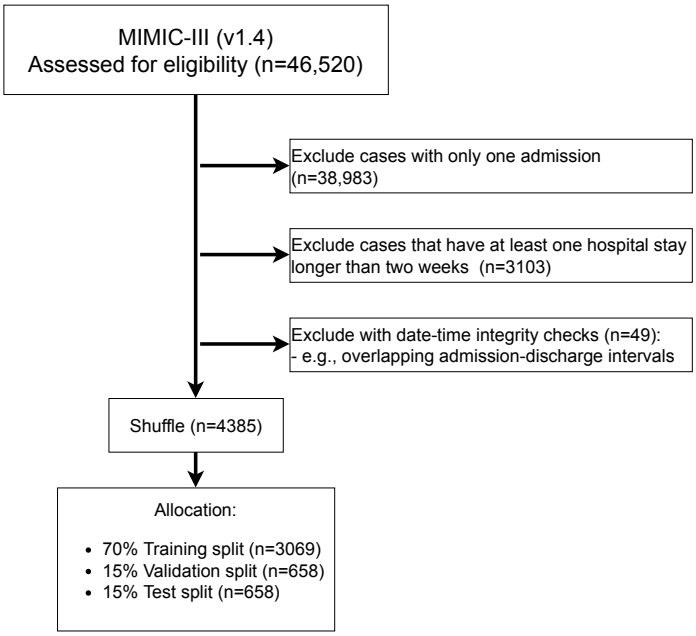


Figure 5: Consort diagram for the cohort extracted from MIMIC-III for our experiments.

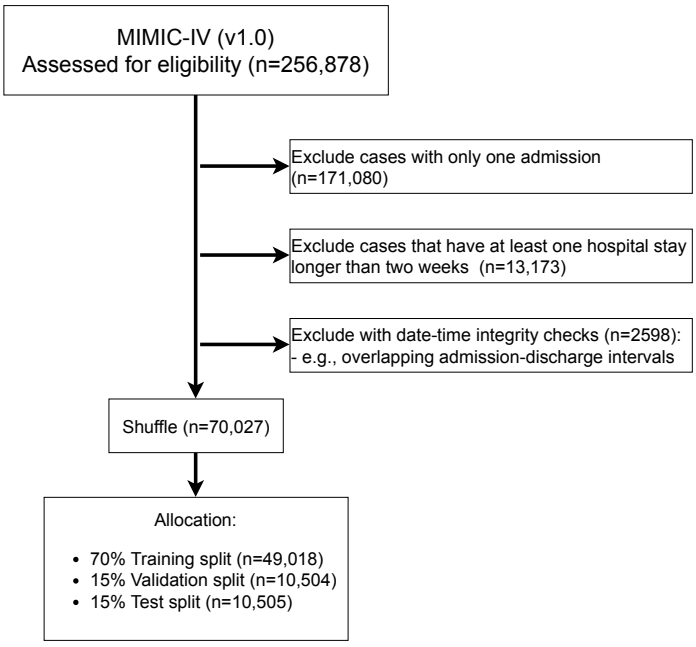


Figure 6: Consort diagram for the cohort extracted from MIMIC-IV for our experiments.

Appendix B. ICE-NODE: Implementation Details & Optimal Configuration

This section provides implementation details and description for the hyperparameter search task to find the optimal configuration of ICE-NODE and ICE-NODE/G, in addition to their corresponding variants ICE-NODE_UNIFORM and ICE-NODE_UNIFORM/G. However, the source code in https://github.com/barahona-research-group/ICE-NODE can be consulted for a better comprehension and the additional details of the optimal configuration finding for GRU, GRU/G, RETAIN, and LogReg. The implementation of our method and the baselines relies on the *automatic differentiation* engine of JAX library (Bradbury et al., 2018), which also provides the scalable gradient computation of neural ODEs using the *adjoint method* (Chen et al., 2018).

We used the optuna library (Akiba et al., 2019) to run the hyperparameter optimisation harnessing distributed runs across multiple nodes. We used the *Tree-structured Parzen Estimator* algorithm (Bergstra et al., 2011) with the default parameters as in optuna[5]. We specified the averaged *visit AUC* metric on the validation partition of MIMIC-III as the hyperparameter optimisation objective.

The hyperparameter space is divided into four categories:

- the clinical embeddings module,
- the dynamics function,
- the decoder function,
- and the training settings.

Two main tasks are conducted: the first task is focused on optimising the configuration of ICE-NODE (with the matrix embeddings). The optimal configuration found for ICE-NODE is reused for ICE-NODE_UNIFORM. The second task is focused on ICE-NODE/G, but the optimisation here is focused only on optimising the configuration of the GRAM embeddings, while the other configurations are fixed to the optimal configuration found for ICE-NODE in the first task. Similarly, the final optimal configuration of ICE-NODE/G is reused for ICE-NODE_UNIFORM/G.

B.1. The Clinical Embeddings Module

Matrix Embeddings for ICE-NODE The only hyperparameter of the matrix embeddings is the embeddings vector size, i.e. d_e in Equation (4). The search domain of d_e is $[30, 60, 90, \ldots, 300]$. The optimal value is found to be $d_e = 300$.

GRAM Embeddings for ICE-NODE/G The optimisation of the configuration of GRAM embeddings considered three hyperparameters:

hyperparameter	type	domain	optimal value
d_e: embedding dimension	integer	$[30, 60, 90, \ldots, 300]$	300
attention method	categorical	[tanh(7), l2(8)]	tanh
ℓ: hidden layer dimension	integer	$[50, 100, \ldots, 300]$	200

5. https://optuna.readthedocs.io/en/stable/reference/generated/optuna.samplers.TPESampler.html

We use the *GloVe* initialisation of the basic embeddings that has been described in (Choi et al., 2017).

B.2. The Dynamics Function

The configuration of the dynamics function considered two hyperparameters

hyperparameter	type	domain	optimal value
d_m: memory state size	integer	$[10, 20, 30, \ldots, 100]$	30
architecture	categorical	[`mlp2`, `mlp3`, `gru`]	`mlp3`

While each architecture is defined as following:

- `mlp2`: is an MLP with two layers, where each layer has (i) the dimensionality $d_h = d_m + d_e$, (ii) the activation function tanh, and (iii) no bias term.
- `mlp3`: is same as `mlp2` but with three layers.
- `gru`: is a minimal GRU described in (Brouwer et al., 2019, Appendix G), and originally authored by Zhou et al. (2016).

B.3. The Decoder Function

The decoder function is implemented as a MLP, where: (i) hidden layers has the dimensionality of d_e and followed by `LeakyReLU` activation, while (ii) the output layer has the dimensionality C (i.e. the coding scheme vocabulary size) and followed by *sigmoid* activation. The only hyperparameter here is the number of layers. The domain of search was limited to $[2, 3]$, and the optimal number of layers found is 2.

B.4. The Update Function

The update function of `ICE-NODE` in Equation (12) adjusts the memory state to accommodate the new information at the new timestamp, and it is implemented as following:

$$\begin{aligned}\boldsymbol{h}_m(t_k^+) &= f_U(\boldsymbol{h}_m(t_k^-), \boldsymbol{h}_e(t_k^-), \boldsymbol{g}(t_1); \boldsymbol{\theta}_U) \\ &= \texttt{GRU}(\boldsymbol{W}_U \begin{bmatrix} \boldsymbol{h}_m(t_k^-) \\ \boldsymbol{g}(t_k) \end{bmatrix} + \boldsymbol{b}_U, \boldsymbol{h}_m(t_k^-); \boldsymbol{\theta}_U),\end{aligned}$$

where $\boldsymbol{W}_U \in \mathbb{R}^{d_e \times 2d_e}$ and $\boldsymbol{b}_U \in \mathbb{R}^{d_e}$ updates the concatenation $[\boldsymbol{h}_m(t_k^-); \boldsymbol{g}(t_k)]$ to a space with the same dimensionality R^{d_e}. $\texttt{GRU} : R^{d_e} \times R^{d_m} \mapsto R^{d_m}$ is an implementation of a GRU cell (Cho et al., 2014)[6], which computes the updated memory state. No new hyperparameter to tune for the update function, as the dimensions d_e and d_m are already included above.

B.5. The Training settings

For each training iteration, we randomly sample, with replacement, a fixed number of patients B from the total of N patients in the training partition. With a slight abuse of terminology, we use the term 'epoch' to refer to a number of training iterations equals

6. https://dm-haiku.readthedocs.io/en/latest/api.html#gru

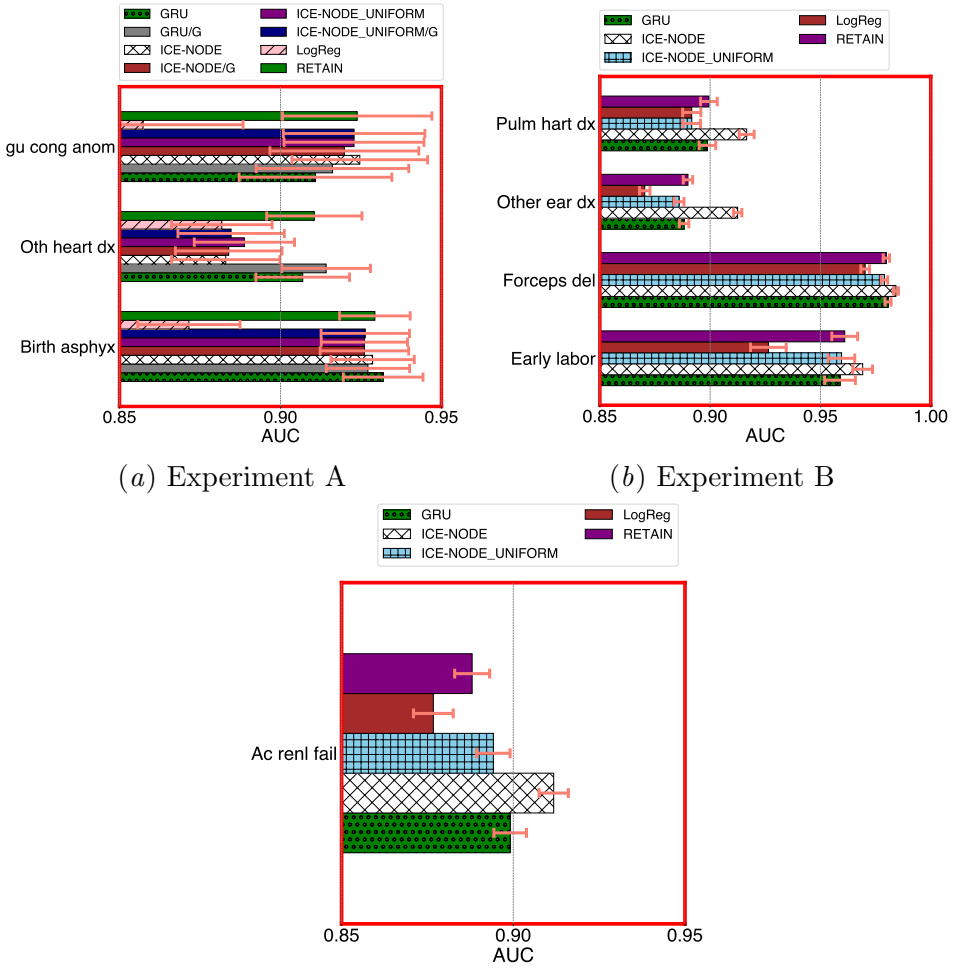
to $\lceil N/B \rceil$, where $\lceil . \rceil$ is the ceiling operator. The number of 'epochs' is fixed to 60. The hyperparameters of the training settings is listed below:

hyperparameter	type	domain	optimal value
optimiser	categorical	[`adam`, `adamax`, `sgd`]	`adam`
η_1: dynamics learning rate	float	`LogUniform`$[10^{-5}, 10^{-2}]$	7.15×10^{-5}
η_2: the other learning rate	float	`LogUniform`$[10^{-5}, 10^{-2}]$	1.14×10^{-3}
`decay_rate`	float	`LogUniform`$[0.1, 0.9]$	0.3
B: batch size	integer	$[2, 4, 8, 16, \ldots, 256]$	256

Early stopping for model selection through the training iterations Since we rely on iterative methods for training our models we adopt the *early stopping*[7] strategy to avoid overfitting our model after a large, fixed number of iterations. Throughout the training iterations, we evaluate the averaged *visit AUC* on the EHRs of the validation set. The *visit AUC* is evaluated for each visit after the first discharge in the patient history, and it estimates the probability of assigning risk scores to the actual clinical codes issued at that visit with higher risk values than those of the unreported clinical codes. This metric is used to guide the model selection throughout the training iterations.

7. https://en.wikipedia.org/wiki/Early_stopping

Appendix C. AUC values for clinical codes highlighted in Figure 3



(*a*) Experiment A (*b*) Experiment B

(*c*) Experiment C

Figure 7: AUC values for clinical codes highlighted in red in Figure 3. The 3 codes in (a) are detected equally well by all methods that use either sequence or full temporal information but not by `LogReg`, which ignores time altogether. The codes in (b) and (c) are only predicted well by `ICE-NODE`, but not by methods that use only sequence information or by `LogReg`.

Appendix D. Further Examples of ICE-NODE Predicted Risk Trajectories

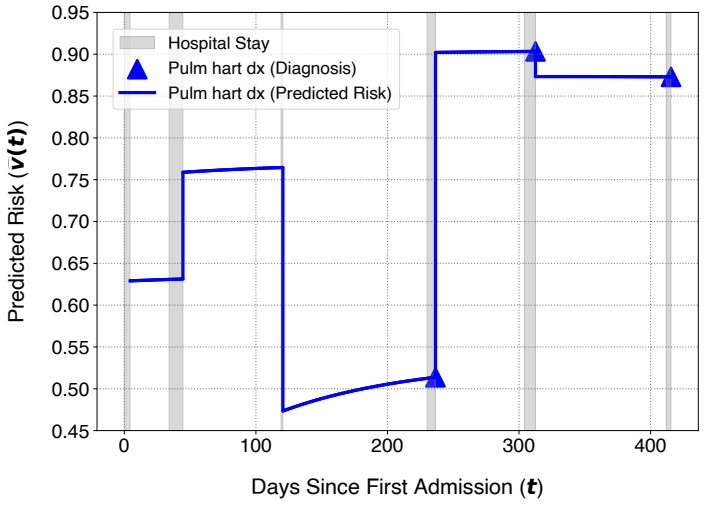


Figure 8: The predicted risk trajectory of '*Pulm hart dx*' for the patient with `subject_id=50093` in the test partition of MIMIC-III. The history of this patient consists of six hospital stays (i.e. admissions-discharges) and the patient was diagnosed with '*Pulm hart dx*' for the first time at the fourth hospital stay onward. After the initial discharge, `ICE-NODE` has predicted a risk for this diagnosis with a probability slightly lower than 0.65. The risk increased after the second discharge despite a negative reporting of '*Pulm hart dx*', but decreased after another negative reporting at the third discharge, after which the risk alarmingly increased with time until the fourth hospital stay.

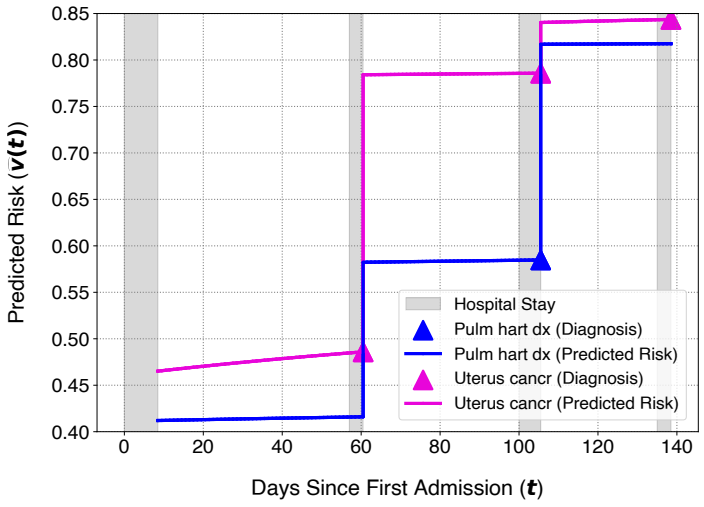


Figure 9: The predicted risk trajectory of '*Pulm hart dx*' together with '*Uterus cancr*' for the patient with `subject_id=11052692` in the test partition of MIMIC-IV. The history of this patient consists of four hospital stays and the first diagnosis of '*Uterus cancr*' was made at the second hospital stay, while '*Pulm hart dx*' at the third hospital stay.

Appendix E. Discussion on recent related work by Peng et al. (2021)

When this research was finished, we learned that a solution based on neural ODEs and transformers has been employed by Peng et al. (2021) to capture information about irregular intervals within the EHRs. Briefly, their approach aims to produce a representation $\boldsymbol{v}_t^o$ for each t-th visit. Using f to denote multiple computational modules, their representation is generated as

$$\boldsymbol{v}_t^o = f(\boldsymbol{v}_t^{\text{code}}, \boldsymbol{v}_t^{\text{LoS}}, \boldsymbol{v}_t^{\text{Interval}})$$

where $\boldsymbol{v}_t^{\text{code}}$ is the clinical code representation of that visit; $\boldsymbol{v}_t^{\text{Interval}}$ is the final solution of a certain IVP from the discharge timestamp of the previous visit to the current admission timestamp, and $\boldsymbol{v}_t^{\text{LoS}}$ (LoS is short for length-of-stay) is the final solution of a certain IVP from the visit admission timestamp until the discharge timestamp. The visit representation $\boldsymbol{v}_t^o$ is then used to predict the clinical codes of the next visit using one-layer *softmax*.

Whereas this approach exploits the temporal dimension pragmatically to produce accurate predictions for next visits, it fuses three different representations, including the ones that capture the irregular intervals in the EHR. Hence it lacks the advantage of producing intermediate representations spanning the time intervals between visits. In contrast, `ICE-NODE` can retrieve the patient state at any arbitrary time, allowing for the retrieval of time-continuous trajectories.