# `RETAIN`: Interpretable Predictive Model in Healthcare using Reverse Time Attention Mechanism

Edward Choi*, Mohammad Taha Bahadori*, Andy Schuetz†, Walter F. Stewart†, Jimeng Sun*

* Georgia Institute of Technology       † Sutter Health

{mp2893,bahadori}@gatech.edu}, {schueta1,stewarwf}@sutterhealth.org, jsun@cc.gatech.edu

## Abstract

Accuracy and interpretation are two goals of any successful predictive models. Most existing works have to suffer the tradeoff between the two by either picking complex black box models such as recurrent neural networks (RNN) or relying on less accurate traditional models with better interpretation such as logistic regression. To address this dilemma, we present REverse Time AttentIoN model (`RETAIN`) for analyzing Electronic Health Records (EHR) data that achieves high accuracy while remaining clinically interpretable. `RETAIN` is a two-level neural attention model that can find influential past visits and significant clinical variables within those visits (e.g,. key diagnoses). `RETAIN` mimics physician practice by attending the EHR data in a reverse time order so that more recent clinical visits will likely get higher attention. Experiments on a large real EHR dataset of 14 million visits from 263K patients over 8 years confirmed the comparable predictive accuracy and computational scalability to the state-of-the-art methods such as RNN. Finally, we demonstrate the clinical interpretation with concrete examples from `RETAIN`.

## 1  Introduction

The broad adoption of Electronic Health Record (EHR) systems has opened the possibility of applying clinical predictive models to improve the quality of clinical services. Several systematic reviews have underlined the care quality improvement in hospitals using predictive analysis [7, 25, 5, 20]. EHR data are represented as temporal sequences of high-dimensional clinical variables (e.g., diagnoses, medications and procedures), where each sequence corresponds to all medical visits from a single patient. Abundance of such EHR data provide great machine learning opportunities for developing accurate while interpretable data driven models. However, the key challenge lies in how to model the temporality and high-dimensionality of these EHR sequences.

Accuracy and interpretation are two goals of any successful predictive models. There is a common belief that one has to trade accuracy for interpretation in favor to simpler models [6]. Traditional methods with great clinical interpretation can be divided into three groups: 1) identifying a set of rules (*e.g.* via decision trees [27]), 2) case-based reasoning by finding similar patients (*e.g.* via k-nearest neighbors [18] and distance metric learning [36]), and 3) identifying a list of risk factors (*e.g.* via LASSO coefficients [15]). While interpretable, all above models ignored the temporality of the EHR data, hence led to sub-optimal models. Latent-variable time-series models, such as [34, 35], do capture the temporality, but often have limited interpretation due to abstract state variables.

Recently, recurrent neural networks (RNN) have been successfully applied in modeling sequential EHR data to perform various predictive analysis such as learning to diagnose [30] and disease progression modeling [11, 14]. Despite the promising gain in accuracy, RNNs are notoriously difficult to interpret. While there have been several attempts at directly interpreting RNNs [19, 26, 8], none of them provide the level of interpretation that can serve the healthcare applications.

In this work, we propose `RETAIN`, a two-level neural attention model for sequential data that provides detailed interpretation of the prediction results while demonstrating the prediction accuracy comparable to RNN. This is made possible by our attention mechanism inspired by the behavior of human physicians. The
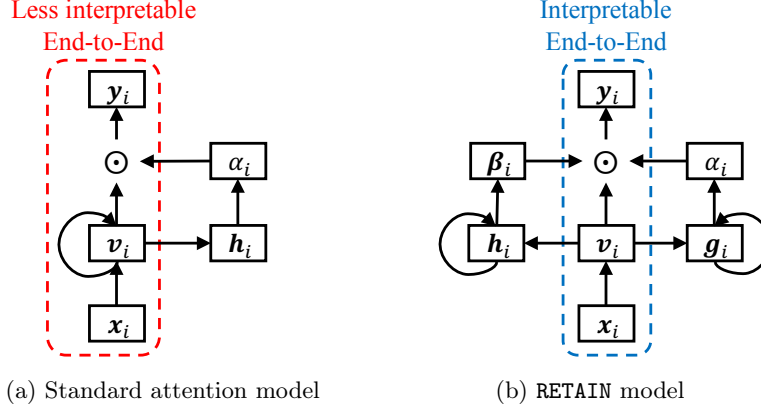
(a) Standard attention model        (b) `RETAIN` model

Figure 1: Common attention models vs. `RETAIN`, using folded diagrams of RNNs. (a) Standard attention mechanism: the recurrence on the hidden state vector $\mathbf{v}_i$ hinders interpretation of the model. (b) Attention mechanism in `RETAIN`: The recurrence is on the attention generation part ($\mathbf{h}_i$ or $\mathbf{g}_i$) while the hidden state $\mathbf{v}_i$ is generated by a simpler mechanism hence better interpretation.

key idea of `RETAIN` (see Figure 1) is to have a temporal attention generation mechanism, while keeping the representation learning part simple and interpretable. Similar to human doctors, `RETAIN` makes a prediction by looking at the patient's past visits in reverse time order, which computationally makes the attention generation mechanism more stable too. `RETAIN` not only considers which past visit is worth paying attention to, but also considers how much individual variables in each visit contribute to the prediction.

Using a large EHR data with 263K patients over 14m visits across 8 years, we compare `RETAIN` against several baseline methods including traditional machine learning methods and RNN variants on heart failure onset prediction. `RETAIN` achieves comparable performance to RNN in both accuracy and speed and outperforms significantly all the other baselines. Finally, we demonstrate `RETAIN` to provide intuitive interpretation through a concrete case study and visualization.

# 2 Methodology

In this section, we first describe the structure of sequential EHR data, our notation, and a general form of predictive analysis in healthcare using EHR. Then, we describe the details of `RETAIN`.

**EHR Structure and our Notation.** EHR data of each patient is modeled as a time-labeled sequence of multivariate observations. Assuming we use $r$ different variables, the $n$-th patient of total $N$ patients can be represented by a sequence of $T^{(n)}$ tuples $(t_i^{(n)}, \mathbf{x}_i^{(n)}) \in \mathbb{R} \times \mathbb{R}^r, i = 1, \ldots, T^{(n)}$. The timestamps $t_i^{(n)}$ denotes the time of the $i$-th visit of the $n$-th patient and $T^{(n)}$ the sequence length of the $n$-th patient. To avoid cluttered notation, we describe the algorithms for a single patient and drop the superscript $(n)$ whenever it is unambiguous. The goal of predictive modeling is to predict labels for each time step $\mathbf{y}_i \in \{0, 1\}^s$ or at the end of the sequence $\mathbf{y} \in \{0, 1\}^s$ (without time index). The number of labels $s$ can be more than 1.

For example, in disease progression modeling (DPM), we are given a sequence of visits where each visit is a set of varying number of medical codes $\{c_1, c_2, \ldots, c_n\}$. The code $c_j$ is the $j$-th code from the vocabulary $\mathcal{C}$. Therefore, in DPM, the number of variables $r = |\mathcal{C}|$ and input vector $\mathbf{x}_i \in \{0, 1\}^{|\mathcal{C}|}$ is a binary vector where the value one in the $j$-th coordinate indicates code $c_j$ is included in $i$-th visit. Given a sequence of visits $\mathbf{x}_1, \ldots, \mathbf{x}_T$, the goal of DPM is, for each time step $i$, to predict the codes occurring at the next visit $\mathbf{x}_2, \ldots, \mathbf{x}_{T+1}$, making the number of labels $s = |\mathcal{C}|$.

In the case of learning to diagnose (L2D) [30], the input vector $\mathbf{x}_i$ consists of measurements (possibly continuous) collected by monitoring devices. If there are $r$ different measurements, then $\mathbf{x}_i \in \mathbb{R}^r$. The goal of L2D is, given input sequence $\mathbf{x}_1, \ldots, \mathbf{x}_T$, to predict the occurrence of a specific disease ($s = 1$) or multiple

diseases ($s > 1$). Without loss of generality, we will describe the algorithm for DPM, as L2D can be seen as a special case of DPM where we make a single prediction at the end of the visit sequence.

In the rest of this section, we will use the abstract symbol RNN to denote any recurrent neural network variants that can cope with the vanishing gradient problem [3], such as LSTM [23], GRU [9], and IRNN [29], with any depth (number of hidden layers).

## 2.1 Preliminaries on Neural Attention Models

Attention based neural network models have recently gained much attraction in image processing [1, 32, 21, 37], natural language processing [2, 22, 33] and speech recognition [12]. The need for attention mechanism can be seen in the language translation task [2]: Representing the entire sentence with one fixed-size vector is inefficient; the neural translation machine usually finds it difficult to translate the given sentence represented by a single vector.

Intuitively, the attention mechanism for language translation works as follows: given a sentence of length $S$ in the original language, we generate $\mathbf{h}_1, \ldots, \mathbf{h}_S$, the representation of the words in the sentence, *e.g.* using an RNN. To find the $j$-th word in the target language, we generate attentions $\alpha_i^j$ for $i = 1, \ldots, S$ for each word in the original sentence. Then, we compute the context vector $\mathbf{c}_j = \sum_i \alpha_i^j \mathbf{h}_i$ and use it to predict the $j$-th word in the target language. In general, the attention mechanism allows the model to focus on a specific word (or words) in the given sentence when generating each word in the target language.

In this work, we define a temporal attention mechanism to provide interpretable prediction models in healthcare. Doctors generally pay attention to specific clinical information (e.g., key risk factors) and their timing when reviewing EHR data. We exploit this insight to develop a temporal attention model that mimics doctors' practice, which will be introduced next.

## 2.2 Reverse Time Attention Model RETAIN

Figure 2 shows the high-level overview of our model. One key idea is to delegate a considerable portion of the prediction responsibility to the attention weights generation process. RNNs become difficult to interpret due to the recurrent weights feeding past information to the hidden layer. Therefore, to consider both the visit-level and the variable-level (individual coordinates of $\mathbf{x}_i$) influence, we use a linear embedding of the input vector $\mathbf{x}_i$. That is, we define

$$\mathbf{v}_i = \mathbf{W}_{emb}\mathbf{x}_i, \tag{Step 1}$$

where $\mathbf{v}_i \in \mathbb{R}^m$ denotes the embedding of the input vector $\mathbf{x}_i \in \mathbb{R}^r$, $m$ the size of the embedding dimension, $\mathbf{W}_{emb} \in \mathbb{R}^{m \times r}$ the embedding matrix to learn. We can easily choose a more sophisticated but still interpretable representation such as multilayer perceptron (MLP) [13, 28] which has been used for representation learning in EHR data [10].

We use two sets of weights for the visit-level attention and the variable-level attention, respectively. The scalars $\alpha_1, \ldots, \alpha_i$ are the visit-level attention weights that govern the influence of each visit embedding $\mathbf{v}_1, \ldots, \mathbf{v}_i$. The vectors $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_i$ are the variable-level attention weights that focus on each coordinate of the visit embedding $v_{1,1}, v_{1,2}, \ldots, v_{1,m}, \ldots, v_{i,1}, v_{i,2}, \ldots, v_{i,m}$.

We use two RNNs, $\text{RNN}_\alpha$ and $\text{RNN}_{\boldsymbol{\beta}}$, to separately generate $\alpha$'s and $\boldsymbol{\beta}$'s as follows,

$$\mathbf{g}_i, \mathbf{g}_{i-1}, \ldots, \mathbf{g}_1 = \text{RNN}_\alpha(\mathbf{v}_i, \mathbf{v}_{i-1}, \ldots, \mathbf{v}_1),$$
$$e_j = \mathbf{w}_\alpha^\top \mathbf{g}_j + b_\alpha, \quad \text{for} \quad j = 1, \ldots, i$$
$$\alpha_1, \alpha_2, \ldots, \alpha_i = \text{Softmax}(e_1, e_2, \ldots, e_i) \tag{Step 2}$$
$$\mathbf{h}_i, \mathbf{h}_{i-1}, \ldots, \mathbf{h}_1 = \text{RNN}_{\boldsymbol{\beta}}(\mathbf{v}_i, \mathbf{v}_{i-1}, \ldots, \mathbf{v}_1)$$
$$\boldsymbol{\beta}_j = \tanh\left(\mathbf{W}_{\boldsymbol{\beta}}\mathbf{h}_j + \mathbf{b}_{\boldsymbol{\beta}}\right) \quad \text{for} \quad j = 1, \ldots, i, \tag{Step 3}$$

where $\mathbf{g}_i \in \mathbb{R}^p$ is the hidden layer of $\text{RNN}_\alpha$ at time step $i$, $\mathbf{h}_i \in \mathbb{R}^q$ the hidden layer of $\text{RNN}_{\boldsymbol{\beta}}$ at time step $i$ and $\mathbf{w}_\alpha \in \mathbb{R}^p, b_\alpha \in \mathbb{R}, \mathbf{W}_{\boldsymbol{\beta}} \in \mathbb{R}^{m \times q}$ and $\mathbf{b}_{\boldsymbol{\beta}} \in \mathbb{R}^m$ are the parameters to learn. The hyperparameters $p$
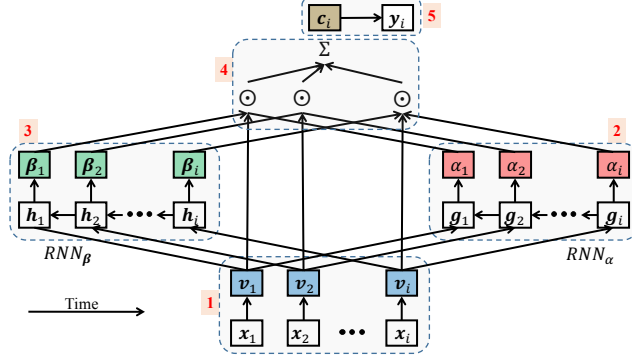
Figure 2: Unfolded view of `RETAIN`'s architecture: Given input sequence $\mathbf{x}_1, \ldots, \mathbf{x}_i$, we predict the label $\mathbf{y}_i$. **Step 1**: Embedding, **Step 2**: generating $\alpha$ values using $\text{RNN}_\alpha$, **Step 3**: generating $\boldsymbol{\beta}$ values using $\text{RNN}_{\boldsymbol{\beta}}$, **Step 4**: Generating the context vector using attention and representation vectors, and **Step 5**: Making prediction. Note that in Steps 2 and 3 we use RNN in the reversed time.

and $q$ determine the hidden layer size of $\text{RNN}_\alpha$ and $\text{RNN}_{\boldsymbol{\beta}}$, respectively. Note that for prediction at each timestamp, we generate a new set of attention vectors $\alpha$ and $\boldsymbol{\beta}$. For simplicity of notation, we do not include the index for predicting at different time steps. In Step 2, we can use Sparsemax [31] instead of Softmax to make the model more interpretable.

Another key idea in `RETAIN` is to generate the attention vectors by running the RNNs backward in time; i.e., $\text{RNN}_\alpha$ and $\text{RNN}_{\boldsymbol{\beta}}$ both take the visit embeddings in a reverse order $\mathbf{v}_i, \mathbf{v}_{i-1}, \ldots, \mathbf{v}_1$. This idea is inspired by the common behavior of physicians: When physicians try to diagnose based on the past records, they typically study the patient's most recent records first, and go back in time. Computationally, running the RNN in reversed time order has several advantages as well: The reverse time order allows us to generate $e$'s and $\boldsymbol{\beta}$'s that dynamically change their values when making predictions at different time steps $i = 1, 2, \ldots, T$. It ensures that the attention vectors will be different at each timestamp and makes the attention generation process computationally more stable.[1]

We generate the context vector $\mathbf{c}_i$ for a patient up to the $i$-th visit as follows,

$$\mathbf{c}_i = \sum_{j=1}^{i} \alpha_j \boldsymbol{\beta}_j \odot \mathbf{v}_j, \qquad \text{(Step 4)}$$

where $\odot$ denotes element-wise multiplication. We use the context vector $\mathbf{c}_i \in \mathbb{R}^m$ to predict the true label $\mathbf{y}_i \in \{0, 1\}^s$ as follows,

$$\widehat{\mathbf{y}}_i = \text{Softmax}(\mathbf{W}\mathbf{c}_i + \mathbf{b}), \qquad \text{(Step 5)}$$

where $\mathbf{W} \in \mathbb{R}^{s \times m}$ and $\mathbf{b} \in \mathbb{R}^s$ are parameters to learn. We use the cross-entropy to calculate the classification loss as follows,

$$\mathcal{L}(\mathbf{x}_1, \ldots, \mathbf{x}_T) = -\frac{1}{N} \sum_{n=1}^{N} \frac{1}{T^{(n)}} \sum_{i=1}^{T^{(n)}} \left( \mathbf{y}_i^\top \log(\widehat{\mathbf{y}}_i) + (\mathbf{1} - \mathbf{y}_i)^\top \log(\mathbf{1} - \widehat{\mathbf{y}}_i) \right) \qquad (1)$$

where we sum the cross entropy errors from all dimensions of $\widehat{\mathbf{y}}_i$. In case of real-valued output $\mathbf{y}_i \in \mathbb{R}^s$, we can change the cross-entropy in Eq. (1) to for example mean squared error.

Overall, our attention mechanism can be viewed as the inverted architecture of the standard attention mechanism for NLP [2] where the words are encoded using RNN and generate the attention weights using

---

[1]For example, feeding visit embeddings in the original order to $\text{RNN}_\alpha$ and $\text{RNN}_{\boldsymbol{\beta}}$ will generate the same $e_1$ and $\boldsymbol{\beta}_1$ for every time step $i = 1, 2, \ldots, T$. Moreover, in many cases, a patient's recent visit records deserve more attention than the old records. Then we need to have $e_{j+1} > e_j$ which makes the process computationally unstable for long sequences.

MLP. Our method, on the other hand, uses MLP to embed the visit information to preserve interpretation and uses RNN to generate two sets of attention weights, recovering the sequential information as well as mimicking the behavior of physicians.

## 3  Interpreting `RETAIN`

Finding the important visits for making a prediction can be derived based on the largest $\alpha_i$, which is simple. However, finding influential variables is slightly more involved which will be described next. Note that a clinical visit consists of multiple medical variables, each of which makes different amount of contribution. The contribution of each variable is determined by $\mathbf{v}$, $\boldsymbol{\beta}$ and $\alpha$, and interpreting $\alpha$ alone will tell which visit is influential in prediction but not why.

We propose a method to interpret the end-to-end behavior of `RETAIN`. By keeping $\alpha$ and $\boldsymbol{\beta}$ values fixed as the attention of doctors, we will analyze the changes in the probability of each label $y_{i,1}, \ldots, y_{i,s}$ in terms of the change in an original input $x_{1,1}, \ldots, x_{1,r}, \ldots, x_{i,1}, \ldots, x_{i,r}$. The $x_{j,k}$ that lead to the largest change in $y_{i,d}$ will be the input variable with highest contribution. More formally, given the sequence $\mathbf{x}_1, \ldots, \mathbf{x}_i$, we are trying to predict the probability of the output vector $\mathbf{y}_i \in \{0,1\}^s$, which can be expressed as follows

$$p(\mathbf{y}_i|\mathbf{x}_1, \ldots, \mathbf{x}_i) = p(\mathbf{y}_i|\mathbf{c}_i) = \text{Softmax}\left(\mathbf{W}\mathbf{c}_i + \mathbf{b}\right) \tag{2}$$

where $\mathbf{c}_i \in \mathbb{R}^m$ denotes the context vector. According to Step 4, $\mathbf{c}_i$ is the sum of the visit embeddings $\mathbf{v}_1, \ldots, \mathbf{v}_i$ weighted by the attentions $\alpha$'s and $\boldsymbol{\beta}$'s. Therefore Eq (2) can be rewritten as follows,

$$p(\mathbf{y}_i|\mathbf{x}_1, \ldots, \mathbf{x}_i) = p(\mathbf{y}_i|\mathbf{c}_i) = \text{Softmax}\left(\mathbf{W}\Big(\sum_{j=1}^{i} \alpha_j \boldsymbol{\beta}_j \odot \mathbf{v}_j\Big) + \mathbf{b}\right) \tag{3}$$

Using the fact that the visit embedding $\mathbf{v}_i$ is the sum of the columns of $\mathbf{W}_{emb}$ weighted by each element of $\mathbf{x}_i$, Eq (3) can be rewritten as follows,

$$
\begin{aligned}
p(\mathbf{y}_i|\mathbf{x}_1, \ldots, \mathbf{x}_i) &= \text{Softmax}\left(\mathbf{W}\Big(\sum_{j=1}^{i} \alpha_j \boldsymbol{\beta}_j \odot \sum_{k=1}^{r} x_{j,k} \mathbf{W}_{emb}[:,k]\Big) + \mathbf{b}\right) \\
&= \text{Softmax}\left(\sum_{j=1}^{i}\sum_{k=1}^{r} x_{j,k}\, \alpha_j \mathbf{W}\Big(\boldsymbol{\beta}_j \odot \mathbf{W}_{emb}[:,k]\Big) + \mathbf{b}\right)
\end{aligned}
\tag{4}
$$

where $x_{j,k}$ is the $k$-th element of the input vector $\mathbf{x}_j$. Eq (4) tells us that the calculation of the likelihood of $\mathbf{y}_i$ can be completely deconstructed down to the variables at each input $\mathbf{x}_1, \ldots, \mathbf{x}_i$. Therefore we can calculate the contribution $\omega$ of the $k$-th variable of the input $\mathbf{x}_j$ at time step $j \leq i$, for predicting $\mathbf{y}_i$ as follows,

$$\omega(\mathbf{y}_i, x_{j,k}) = \underbrace{\alpha_j \mathbf{W}(\boldsymbol{\beta}_j \odot \mathbf{W}_{emb}[:,k])}_{\text{Contribution coefficient}}\ \underbrace{x_{j,k}}_{\text{Input value}}\ , \tag{5}$$

where the index $i$ of $\mathbf{y}_i$ is omitted in the $\alpha_j$ and $\boldsymbol{\beta}_j$. As we have described in Section 2.2, we are generating $\alpha$'s and $\boldsymbol{\beta}$'s at time step $i$ in the visit sequence $\mathbf{x}_1, \ldots, \mathbf{x}_T$. Therefore the index $i$ is always assumed for $\alpha$'s and $\boldsymbol{\beta}$'s. Additionally, Eq (5) shows that when we are using a binary input value, the coefficient itself is the contribution. However, when we are using a continuous input value, we need to multiply the coefficient and the input value $x_{j,k}$ to correctly calculate the contribution.

## 4  Experiments

In this section, we demonstrate that while `RETAIN` is competitive with RNNs in terms of performance, we can interpret the knowledge learned by it. Due to lack of space, we only report the results on learning to diagnose (L2D) task and defer the results on disease progression modeling (DPM) to Appendix B. The source code of RETAIN is publicly available at https://github.com/mp2893/retain.

Table 1: Statistics of EHR dataset. (D:Diagnosis, R:Medication, P:Procedure)

| # of patients | 263,683 | Avg. # of codes in a visit | 3.03 |
|---|---|---|---|
| # of visits | 14,366,030 | Max # of codes in a visit | 62 |
| Avg. # of visits per patient | 54.48 | Avg. # of Dx codes in a visit | 1.83 |
| # of medical code groups | 615 (D:283, R:94, P:238) | Max # of Dx in a visit | 42 |

## 4.1 Experimental setting

**Source of data:** The dataset consists of Electronic Health Records collected by a non-profit health organization. The patients are middle-aged adults chosen for a heart failure study. From the encounter records, medication orders, procedure orders and problem lists, we extracted visit records consisting of diagnosis, medication and procedure codes. To reduce the dimensionality while preserving the clinical information, we used existing medical groupers to aggregate the codes into input variables. The details of the medical groupers are given in the Appendix A. The statistics of the dataset are provided in Table 1.

**Implementation details:** We implemented `RETAIN` with Theano 0.8 [4]. For training the model, we used Adadelta [38] with the mini-batch of 100 patients. The training was done in a machine equipped with Intel Xeon E5-2630, 256GB RAM, two Nvidia Tesla K80's and CUDA 7.5.

**Baselines:** We use the following baseline models.

- **Logistic regression (LR)**: We compute the counts of medical codes for each patient based on all her visits as input variables and normalize the vector to zero mean and unit variance. We use the resulting vector to train the logistic regression.

- **MLP**: We use the same feature construction as **LR**, but put a hidden layer of size 256 between the inpuputt and output.

- **RNN**: RNN with two hidden layers of size 256 implemented by the GRU. Input sequences $\mathbf{x}_1, \ldots, \mathbf{x}_i$ are used. Logistic regression is applied to the top hidden layer. We use two layers of RNN of to match the model complexity of `RETAIN`.

- **RNN+$\boldsymbol{\alpha}_M$**: One layer single directional RNN (hidden layer size 256) along time to generate the input embeddings $\mathbf{v}_1, \ldots, \mathbf{v}_i$. We use the MLP with a single hidden layer of size 256 to generate the visit-level attentions $\alpha_1, \ldots, \alpha_i$. We use the input embeddings $\mathbf{v}_1, \ldots, \mathbf{v}_i$ as the input to the MLP. This baseline corresponds to Figure 1a.

- **RNN+$\boldsymbol{\alpha}_R$**: This is similar to **RNN+$\boldsymbol{\alpha}_M$** but use the reverse-order RNN (hidden layer size 256) to generate the visit-level attentions $\alpha_1, \ldots, \alpha_i$. We use this baseline to confirm the effectiveness of generating the attentions using reverse time order.

The baselines are visually illustrated and compared in Appendix C. We use the same implementation, hyper-parameter exploration and training method for the baselines as described above. We describe in detail the hyper-parameters, regularization and drop-out strategies for the baseline models in Appendix A.

**Evaluation measures:** We use two types of metrics to measure the accuracy.

- **Negative log-likelihood on test set**: Negative log-likelihood measures the model loss on the test set. The loss can be calculated by Eq (1).

- **Area Under the ROC Curve (AUC)**: is the area under the receiver operating characteristic curve of comparing $\widehat{y}_i$ with with the true label $y_i$. AUC is more robust to imbalanced positive/negative prediction labels which makes is appropriate for evaluation of classification accuracy in the heart failure prediction task.

We also report the standard deviation of the performance measures by bootstrapping the results on the test set for 10,000 times.

Table 2: Heart failure prediction performance of `RETAIN` and the baselines

| Model | Test Neg Log Likelihood | AUC | Train Time / epoch | Test Time |
|---|---|---|---|---|
| LR | $0.3269 \pm 0.0105$ | $0.7900 \pm 0.0111$ | 0.15s | 0.11s |
| MLP | $0.2959 \pm 0.0083$ | $0.8256 \pm 0.0096$ | 0.25s | 0.11s |
| RNN | $0.2577 \pm 0.0082$ | $0.8706 \pm 0.0080$ | 10.3s | 0.57s |
| RNN+$\alpha_M$ | $0.2691 \pm 0.0082$ | $0.8624 \pm 0.0079$ | 6.7s | 0.48s |
| RNN+$\alpha_R$ | $0.2605 \pm 0.0088$ | $\mathbf{0.8717} \pm 0.0080$ | 10.4s | 0.62s |
| `RETAIN` | $\mathbf{0.2562} \pm 0.0083$ | $0.8705 \pm 0.0081$ | 10.8s | 0.63s |

## 4.2 Heart Failure Prediction

**Objective:** Given a visit sequence $\mathbf{x}_1, \ldots, \mathbf{x}_T$, we try to predict if the patient will be diagnosed with heart failure (HF). This can be seen as a special case of DPM where we make a prediction for a single disease at the end of the sequence. Since this is a binary prediction task, we use the logistic sigmoid function instead of the Softmax in Step 5.

**Cohort construction:** From the source dataset, 3,884 cases are selected and approximately 10 controls are selected for each case (28,903 controls). The case/control selection criteria are fully described in the supplementary section. Cases have index dates to denote the date they are diagnosed with HF. Controls have the same index dates as their corresponding cases. We extract diagnosis codes, medication codes and procedure codes in the 18-months window before the index date.

**Training details:** The patient cohort was divided into the training, validation and test set in a 0.75:0.1:0.15 ratio. The validation set was used to determine the values of the hyper-parameters. The details of hyper-parameter tuning is provided in the supplementary section.

**Results:** Table 2 compares the prediction performance of `RETAIN` against the baselines. There is a large gap in the performance of logistic regression, MLP, and temporal learning algorithms including RNN variants and our `RETAIN`, because the temporality is modeled. We see that while `RETAIN` is quite competitive with the best performing RNN variants, it beats RNN in terms of test negative log-likelihood and is very close to the best one in AUC measure. Given the interpretation benefit, `RETAIN` can be a great choice for clinical decision support applications.

Note that RNN+$\alpha_R$ model are a degenerated version of `RETAIN` with only scalar attention measures, which is still a competitive model as shown in table 2. This confirms the efficiency of generating attention weights using the RNN. However, RNN+$\alpha_R$ model only provides scalar visit-level attention, which is not sufficient for healthcare applications. Patients often receives several medical codes at a single visit, and it will be important to distinguish their relative importance to the target. We show such a case study in section 4.3.

Table 2 also shows the scalability of `RETAIN`. The training time is the number of seconds to train the model over the entire training set once, i.e., training time for 1 epoch. The test time is the number of seconds to generate the prediction output for the entire test set. We use the mini-batch of 100 patients in both cases. Note that RNN takes longer time than RNN+$\alpha_M$ because of its two-layer structure, whereas RNN+$\alpha_M$ uses a single layer RNN. The models that use two RNNs (RNN, RNN+$\alpha_R$, `RETAIN`)[2] take similar time to train for one epoch. However, each model takes different number of epochs to converge. RNN typically takes approximately 10 epochs, RNN+$\alpha_M$ and RNN+$\alpha_R$ 15 epochs and `RETAIN` 30 epochs. Lastly, training the attention models (RNN+$\alpha_M$, RNN+$\alpha_R$ and `RETAIN`) for DPM would take considerably longer than L2D, because they will generate context vectors at each time step. RNN, on the other hand, does not require additional computation other than embedding the visit to its hidden layer to predict target labels at each time step. Therefore, in DPM, the training time of the attention models will linearly increase by the length of the input sequence.

---

[2]RNN uses two layers of RNN, RNN+$\alpha_R$ uses one for visit embedding and one for generating $\alpha$, `RETAIN` uses each for generating $\alpha$ and $\boldsymbol{\beta}$
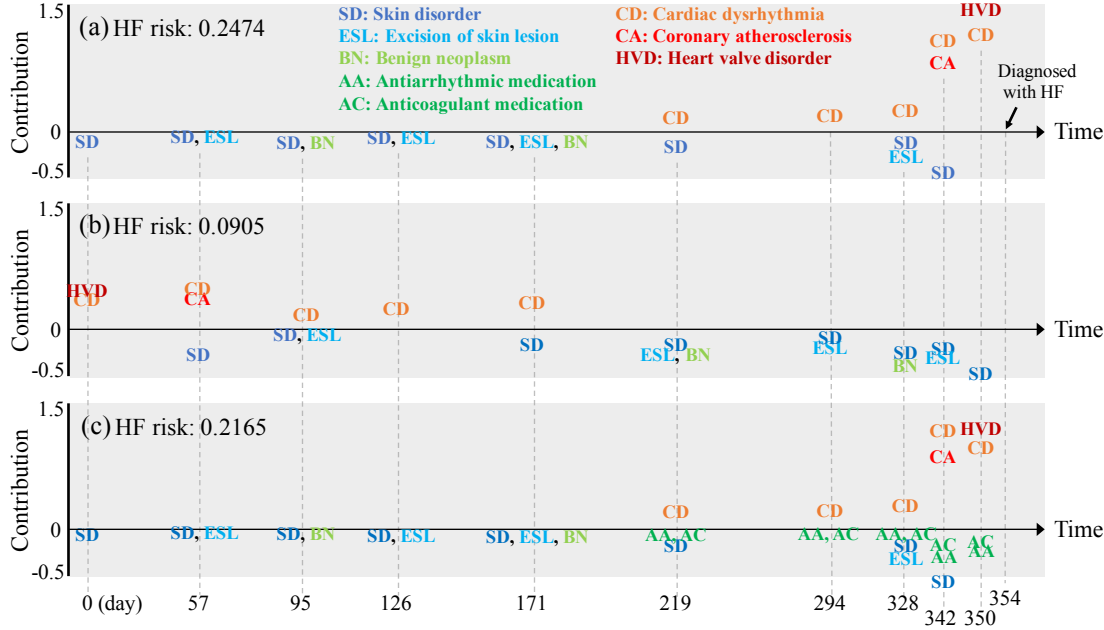
Figure 3: (a) Visualizing the visit records of a heart failure (HF) case patient and analyze the contribution of the variables (diagnosis codes) for making the binary prediction. $x$-axis represents the time and $y$-axis represents the amount of contributions of each code. (b) We reverse the order of the visit sequence to see if RETAIN can properly adjust to the new visit sequence. (c) We add medication codes to the visit record to see how it changes the behavior of RETAIN.

## 4.3    Model Interpretation for Heart Failure Prediction

We demonstrate the interpretability of RETAIN by studying its behavior in the HF prediction task. We choose a HF patient from the test set and calculate the contribution of the variables (medical codes in this case) for making the binary prediction. Figure 3a is the visualization of the contributions of the variables in each visit. The patient suffered from skin problems, *skin disorder* (SD), *benign neoplasm* (BN), *excision of skin lesion* (ESL), for some time before showing symptoms of HF, *cardiac dysrhythmia* (CD), *heart valve disease* (HVD) and *coronary atherosclerosis* (CA), then being diagnosed with HF at the end. We can see that skin-related codes from the earlier visits made little contribution to HF prediction as expected. RETAIN properly puts much attention to the HF-related codes that occurred in recent visits.

In order to evaluate RETAIN's ability to generate attentions considering the temporality, we reverse the visit sequence of Figure 3a and feed it to RETAIN. Figure 3b shows the contribution of the medical codes of the reversed visit record. HF-related codes in the old visits are still making positive contributions, but not as much as they did in Figure 3a. Figure 3b also emphasizes RETAIN's superiority to interpretable, but stationary models such as logistic regression. Stationary models often aggregate past information and remove the temporality from the input data, which can mistakenly lead to the same risk prediction for Figure 3a and 3b. RETAIN, however, can correctly digest the sequential information and calculates the HF risk score of 9.0%, which is significantly lower than that of Figure 3a.

Figure 3c shows how the contributions of the codes change when we gave the patient proper medications in the early stage. We added two medications from day 219: *antiarrhythmics* (AA) and *anticoagulants* (AC), both of which are used to treat *cardiac dysrhythmia* (CD). We can see that the two medications are making negative contributions as expected, especially towards the end of the record. The medications helped decrease the positive contributions of *heart valve disease* and *cardiac dysrhythmia* in the last visit. Indeed, the HF risk prediction (.2165) of Figure 3c is lower than that of Figure 3a (.2474). This suggests that taking proper medications can help the patient in reducing the HF risk.

8

# 5 Conclusion

We present the need for interpretable and accurate predictive model in healthcare applications. Given the power of recurrent neural networks for analyzing sequential data, we proposed `RETAIN`, which preserves RNN's predictive power while allowing a higher degree of interpretation. The key idea of `RETAIN` is to improve the prediction accuracy through a sophisticated attention generation process, while keeping the representation learning part simple for interpretation, making the entire algorithm accurate and interpretable. `RETAIN` trains two RNN in a reverse time order to efficiently generate the appropriate attention variables. For future work, we plan to developing an interactive visualization system for `RETAIN` and evaluating `RETAIN` in healthcare applications.

# References

[1] J. Ba, V. Mnih, and K. Kavukcuoglu. Multiple object recognition with visual attention. In *ICLR*, 2015.

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

[3] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166, 1994.

[4] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of SciPy*, 2010.

[5] A. D. Black, J. Car, C. Pagliari, C. Anandan, K. Cresswell, T. Bokun, B. McKinstry, R. Procter, A. Majeed, and A. Sheikh. The impact of ehealth on the quality and safety of health care: a systematic overview. *PLoS Med*, 8(1):e1000387, 2011.

[6] R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm, and N. Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *KDD*, 2015.

[7] B. Chaudhry, J. Wang, S. Wu, M. Maglione, W. Mojica, E. Roth, S. C. Morton, and P. G. Shekelle. Systematic review: impact of health information technology on quality, efficiency, and costs of medical care. *Annals of internal medicine*, 144(10):742–752, 2006.

[8] Z. Che, S. Purushotham, R. Khemani, and Y. Liu. Distilling knowledge from deep networks with applications to healthcare domain. *arXiv preprint arXiv:1512.03542*, 2015.

[9] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*, 2014.

[10] E. Choi, M. T. Bahadori, E. Searles, C. Coffey, and J. Sun. Multi-layer representation learning for medical concepts. In *KDD*, 2016.

[11] E. Choi, M. T. Bahadori, and J. Sun. Doctor ai: Predicting clinical events via recurrent neural networks. *arXiv preprint arXiv:1511.05942*, 2015.

[12] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio. Attention-based models for speech recognition. In *NIPS*, pages 577–585, 2015.

[13] D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 2009.

[14] C. Esteban, O. Staeck, Y. Yang, and V. Tresp. Predicting clinical events by combining static and dynamic information using recurrent neural networks. *arXiv preprint arXiv:1602.02685*, 2016.

[15] A. S. Fleisher, B. B. Sowell, C. Taylor, A. C. Gamst, R. C. Petersen, L. J. Thal, and f. t. A. D. C. Study. Clinical predictors of progression to Alzheimer disease in amnestic mild cognitive impairment. *Neurology*, 68(19):1588–1595, May 2007.

[16] A. for Healthcare Research and Quality. Clinical classifications software (ccs) for icd-9-cm. `https://www.hcup-us.ahrq.gov/toolssoftware/ccs/ccs.jsp`. Accessed: 2016-04-01.

[17] A. for Healthcare Research and Quality. Clinical classifications software for services and procedures. `https://www.hcup-us.ahrq.gov/toolssoftware/ccs_svcsproc/ccsvcproc.jsp`. Accessed: 2016-04-01.

[18] B. Gallego, S. R. Walter, R. O. Day, A. G. Dunn, V. Sivaraman, N. Shah, C. A. Longhurst, and E. Coiera. Bringing cohort studies to the bedside: framework for a 'green button' to support clinical decision-making. *Journal of Comparative Effectiveness Research*, pages 1–7, May 2015.

[19] J. Ghosh and V. Karamcheti. Sequence learning with recurrent networks: analysis of internal representations. In *Aerospace Sensing*, pages 449–460. International Society for Optics and Photonics, 1992.

[20] C. L. Goldzweig, A. Towfigh, M. Maglione, and P. G. Shekelle. Costs and benefits of health information technology: new trends from the literature. *Health affairs*, 28(2):w282–w293, 2009.

[21] K. Gregor, I. Danihelka, A. Graves, and D. Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

[22] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. Teaching machines to read and comprehend. In *NIPS*, pages 1684–1692, 2015.

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

[24] W. K. C. D. Information. Medi-span electronic drug file (med-file) v2. `http://www.wolterskluwercdi.com/drug-data/medi-span-electronic-drug-file/`. Accessed: 2016-04-01.

[25] A. K. Jha, C. M. DesRoches, E. G. Campbell, K. Donelan, S. R. Rao, T. G. Ferris, A. Shields, S. Rosenbaum, and D. Blumenthal. Use of electronic health records in us hospitals. *N Engl J Med*, 2009.

[26] A. Karpathy, J. Johnson, and F.-F. Li. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

[27] A. N. Kho, M. G. Hayes, L. Rasmussen-Torvik, J. A. Pacheco, W. K. Thompson, L. L. Armstrong, J. C. Denny, P. L. Peissig, A. W. Miller, W.-Q. Wei, S. J. Bielinski, C. G. Chute, C. L. Leibson, G. P. Jarvik, D. R. Crosslin, C. S. Carlson, K. M. Newton, W. A. Wolf, R. L. Chisholm, and W. L. Lowe. Use of diverse electronic medical record systems to identify genetic risk for type 2 diabetes within a genome-wide association study. *JAMIA*, 19(2):212–218, Apr. 2012.

[28] Q. V. Le. Building high-level features using large scale unsupervised learning. In *ICASSP*, 2013.

[29] Q. V. Le, N. Jaitly, and G. E. Hinton. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015.

[30] Z. C. Lipton, D. C. Kale, C. Elkan, and R. Wetzell. Learning to Diagnose with LSTM Recurrent Neural Networks. In *ICLR*, 2016.

[31] A. F. Martins and R. F. Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *ICML*.

[32] V. Mnih, N. Heess, A. Graves, et al. Recurrent models of visual attention. In *NIPS*, 2014.

[33] A. M. Rush, S. Chopra, and J. Weston. A neural attention model for abstractive sentence summarization. In *EMNLP*, 2015.

[34] S. Saria, D. Koller, and A. Penn. Learning individual and population level traits from clinical temporal data. In *NIPS, Predictive Models in Personalized Medicine workshop*, 2010.

[35] P. Schulam and S. Saria. A probabilistic graphical model for individualizing prognosis in chronic, complex diseases. In *AMIA*, volume 2015, page 143, 2015.

[36] J. Sun, F. Wang, J. Hu, and S. Edabollahi. Supervised patient similarity measure of heterogeneous patient records. *ACM SIGKDD Explorations Newsletter*, 14(1):16–24, 2012.

[37] K. Xu, J. Ba, R. Kiros, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.

[38] M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

# A    Details of the experiment settings

## A.1    Hyper-parameter Tuning

We used the validation set to tune the hyper-parameters: visit embedding size $m$, $RNN_\alpha$'s hidden layer size $p$, $RNN_\beta$'s hidden layer size $q$, $L_2$ regularization coefficient, and drop-out rates.

$L_2$ regularization was applied to all weights except the ones in $RNN_\alpha$ and $RNN_\beta$. Two separate drop-outs were used on the visit embedding $\mathbf{v}_i$ and the context vector $\mathbf{c}_i$. We performed the random search with predefined ranges $m, p, q \in \{32, 64, 128, 200, 256\}$, $L_2 \in \{0.1, 0.01, 0.001, 0.0001\}$, $dropout_{\mathbf{v}_i}, dropout_{\mathbf{c}_i} \in \{0.0, 0.2, 0.4, 0.6, 0.8\}$. We also performed the random search with $m$, $p$ and $q$ fixed to 256.

The final value we used to train `RETAIN` for heart failure prediction is $m, p, q = 128$, $dropout_{\mathbf{v}_i} = 0.6$, $dropout_{\mathbf{c}_i} = 0.6$ and 0.0001 for the $L_2$ regularization coefficient.

## A.2    Code Grouper

Diagnosis codes, medication codes and procedure codes in the dataset are respectively using International Classification of Diseases (ICD-9), Generic Product Identifier (GPI) and Current Procedural Terminology (CPT).

Diagnosis codes are grouped by Clinical Classifications Software for ICD-9[16] which reduces the number of diagnosis code from approximately 14,000 to 283. Medication codes are grouped by Generic Product Identifier Drug Group[24] which reduces the dimension to from approximately 151,000 to 96. Procedure codes are grouped by Clinical Classifications Software for CPT[17], which reduces the number of CPT codes from approximately 9,000 to 238.

## A.3    Training Specifics of the Basline Models

- **LR**: We use 0.01 $L_2$ regularization coefficient for the logistic regression weight.

- **MLP**: We use drop-out rate 0.6 on the output of the hidden layer. We use 0.0001 $L_2$ regularization coefficient for the hidden layer weight and the logistic regression weight.

- **RNN**: We use drop-out rate 0.6 on the outputs of both hidden layers. We use 0.0001 $L_2$ regularization coefficient for the logistic regression weight. The dimension size of both hidden layers is 256.

- **RNN+$\boldsymbol{\alpha}_M$**: We use drop-out rate 0.4 on the output of the hidden layer and 0.6 on the output of the context vector $\sum_i \alpha_i \mathbf{v}_i$. We use 0.0001 $L_2$ regularization coefficient for the hidden layer weight of the MLP that generates $\alpha$'s and the logistic regression weight. The dimension size of the hidden layers in both RNN and MLP is 256.

- **RNN+$\boldsymbol{\alpha}_R$**: We use drop-out rate 0.4 on the output of the hidden layer and 0.6 on the output of the context vector $\sum_i \alpha_i \mathbf{v}_i$. We use 0.0001 $L_2$ regularization coefficient for the hidden layer weight of the RNN that generates $\alpha$'s and the logistic regression weight. The dimension size of the hidden layers in both RNNs is 256.

## A.4    Heart Failure Case/Control Selection Criteria

Case patients were 40 to 85 years of age at the time of HF diagnosis. HF diagnosis (HFDx) is defined as: 1) Qualifying ICD-9 codes for HF appeared in the encounter records or medication orders. Qualifying ICD-9 codes are displayed in Table 3. 2) a minimum of three clinical encounters with qualifying ICD-9 codes had to occur within 12 months of each other, where the date of diagnosis was assigned to the earliest of the three dates. If the time span between the first and second appearances of the HF diagnostic code was greater than 12 months, the date of the second encounter was used as the first qualifying encounter. The date at which HF diagnosis was given to the case is denoted as HFDx. Up to ten eligible controls (in terms of sex, age, location) were selected for each case, yielding an overall ratio of 9 controls per case. Each control was also

Table 3: Qualifying ICD-9 codes for heart failure

| ICD-9 Code | Description |
| --- | --- |
| 398.91 | Rheumatic heart failure (congestive) |
| 402.01 | Malignant hypertensive heart disease with heart failure |
| 402.11 | Benign hypertensive heart disease with heart failure |
| 402.91 | Unspecified hypertensive heart disease with heart failure |
| 404.01 | Hypertensive heart and chronic kidney disease, malignant, with heart failure and with chronic kidney disease stage I through stage IV, or unspecified |
| 404.03 | Hypertensive heart and chronic kidney disease, malignant, with heart failure and with chronic kidney disease stage V or end stage renal disease |
| 404.11 | Hypertensive heart and chronic kidney disease, benign, with heart failure and with chronic kidney disease stage I through stage IV, or unspecified |
| 404.13 | Hypertensive heart and chronic kidney disease, benign, with heart failure and chronic kidney disease stage V or end stage renal disease |
| 404.91 | Hypertensive heart and chronic kidney disease, unspecified, with heart failure and with chronic kidney disease stage I through stage IV, or unspecified |
| 404.93 | Hypertensive heart and chronic kidney disease, unspecified, with heart failure and chronic kidney disease stage V or end stage renal disease |
| 428.0 | Congestive heart failure, unspecified |
| 428.1 | Left heart failure |
| 428.20 | Systolic heart failure, unspecified |
| 428.21 | Acute systolic heart failure |
| 428.22 | Chronic systolic heart failure |
| 428.23 | Acute on chronic systolic heart failure |
| 428.30 | Diastolic heart failure, unspecified |
| 428.31 | Acute diastolic heart failure |
| 428.32 | Chronic diastolic heart failure |
| 428.33 | Acute on chronic diastolic heart failure |
| 428.40 | Combined systolic and diastolic heart failure, unspecified |
| 428.41 | Acute combined systolic and diastolic heart failure |
| 428.42 | Chronic combined systolic and diastolic heart failure |
| 428.43 | Acute on chronic combined systolic and diastolic heart failure |
| 428.9 | Heart failure, unspecified |

assigned an index date, which is the HFDx of the matched case. Controls are selected such that they did not meet the operational criteria for HF diagnosis prior to the HFDx plus 182 days of their corresponding case. Control subjects were required to have their first office encounter within one year of the matching HF case patient's first office visit, and have at least one office encounter 30 days before or any time after the case's HF diagnosis date to ensure similar duration of observations among cases and controls.

# B    Results on disease progression modeling

**Objective:** Given a sequence of visits $\mathbf{x}_1, \ldots, \mathbf{x}_T$, the goal of DPM is, for each time step $i$, to predict the codes occurring at the next visit $\mathbf{x}_2, \ldots, \mathbf{x}_{T+1}$. However, as we are interested in the disease progression, we create a separate set of labels $\mathbf{y}_1, \ldots, \mathbf{y}_T$ that do not contain non-diagnosis codes such as medication codes or procedure codes. Therefore $\mathbf{y}_i$ will contain diagnosis codes from the next visit $\mathbf{x}_{i+1}$.

    **Dataset:** We divide the entire dataset described in Table 1 into 0.75:0.10:0.15 ratio, respectively for training set, validation set, and test set.

    **Baseline:** We use the same baseline models we used for HF prediction. However, since we are predicting 283 binary labels now, we replace the logistic regression function with the Softmax function. The drop-out and $L_2$ regularization policies remain the same.

    For LR and MLP, at each step $i$, we aggregate maximum ten past input vectors[3] $\mathbf{x}_{i-9}, \ldots, \mathbf{x}_i$ to create a pseudo-context vector $\widehat{\mathbf{c}}_i$. LR applies the Softmax function on top of $\widehat{\mathbf{c}}_i$. MLP places a hidden layer on top of $\widehat{\mathbf{c}}_i$ then applies the Softmax function.

    **Evaluation metric:** We use the negative log likelihood Eq (1) on the test set to evaluate the model performance. We also use Recall@$k$ as an additional metric to measure the prediction accuracy.

- **Recall@$k$:** Given a sequence of visits $\mathbf{x}_1, \ldots, \mathbf{x}_T$, we evaluate the model performance based on how

---

[3]We also tried aggregating all past input vectors $\mathbf{x}_1, \ldots, \mathbf{x}_i$, but the performance was slightly worse than using just ten.

Table 4: Disease progression modeling performance of `RETAIN` and the baselines

| Model | Negative Likelihood | Recall@5 | Recall@10 |
|---|---|---|---|
| LR | 0.0288 | 43.15 | 55.84 |
| MLP | 0.0267 | 50.72 | 65.02 |
| RNN | **0.0258** | **55.18** | **69.09** |
| RNN+$\alpha_M$ | 0.0262 | 52.15 | 65.81 |
| RNN+$\alpha_R$ | 0.0259 | 53.89 | 67.45 |
| `RETAIN` | 0.0259 | 54.25 | 67.74 |

accurately it can predict the diagnosis codes $\mathbf{y}_1, \ldots, \mathbf{y}_T$. We use the average Recall@$\boldsymbol{k}$, which is expressed as below,

$$\frac{1}{N} \sum_{n=1}^{N} \frac{1}{T^{(n)}} \sum_{i=1}^{T^{(n)}} \text{Recall@}\boldsymbol{k}(\widehat{\mathbf{y}}_i), \quad where \quad \text{Recall@}\boldsymbol{k}(\widehat{\mathbf{y}}_i) = \frac{|\text{argsort}(\widehat{\mathbf{y}}_i)[:k] \cap nonzero(\mathbf{y}_i)|}{|nonzero(\mathbf{y}_i)|}$$

where $argsort$ returns a list of indices that will decrementally sort a given vector and $nonzero$ returns a list of indices of the coordinates with non-zero values. We use Recall@$\boldsymbol{k}$ because of its similar nature to the way a human physician performs the differential diagnostic procedure, which is to generate a list of most likely diseases for an undiagnosed patient, then perform medical practice until the true disease, or diseases are determined.

**Prediction accuracy:** Table 4 displays the prediction performance of `RETAIN` and the baselines. We use $k = 5, 10$ for Recall@$\boldsymbol{k}$ to allow a reasonable number of prediction trials, as well as cover complex patients who often receive multiple diagnosis codes at a single visit.

RNN shows the best prediction accuracy for DPM. However, considering the purpose of DPM, which is to assist doctors to provide quality care for the patient, black-box behavior of RNN makes it unattractive as a clinical tool. On the other hand, `RETAIN` performs as well as other attention models, only slightly inferior to RNN, provides full interpretation of its prediction behavior, making it a feasible solution for clinical applications.

The interesting finding in Table 4 is that MLP is able to perform as accurately as RNN+$\alpha_M$ in terms of Recall@10. Considering the fact that MLP uses aggregated information of past ten visits, we can assume that DPM depends more on the frequency of disease occurrences rather than the order in which they occurred. This is quite different from the HF prediction task, where stationary models (LF, MLP) performed significantly worse than sequential models.

## C   Illustration and comparison of the baselines

Figure 4 illustrates the baselines used in the experiments and shows the relationship among them.
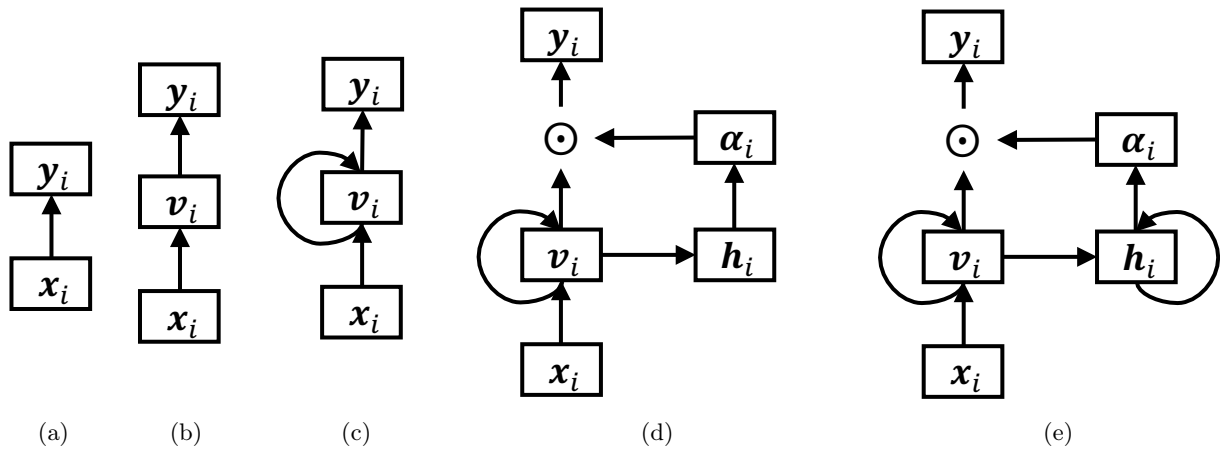
Figure 4: Graphical illustration of the baselines: (a) Logistic regression (LR), (b) Multilayer Perceptron (MLP), (c) Recurrent neural network (RNN), (d) RNN with attention vectors generated via an MLP (RNN+$\boldsymbol{\alpha}_M$), (e) RNN with attention vectors generated via an RNN (RNN+$\boldsymbol{\alpha}_R$). RETAIN is given in Figure 1b.