Towards Intelligent Healthcare: Predicting Future Diagnoses Using a Modified Bi-GRU Approach

Candidate numbers: 25018, 33608, 27508, 34896

Abstract

With the rise of Electronic Health Records (EHRs) and advancements in medical technology, healthcare is shifting towards data-driven approaches. Leveraging electronic medical data for disease diagnosis and prediction can improve patient management and treatment accuracy. Despite extensive research conducted in this field, accurately predicting future diseases and medical events remains a challenge. This paper presents a model to predict future diagnosis codes using patients' existing codes from the MIMIC-III dataset. Through extensive experiments, a modified Bi-GRU model achieves a recall@30 of 0.82. This study offers a reliable predictive model for real-world applications and insights for future research in medical artificial intelligence.

1. Introduction

In modern healthcare settings, Electronic Health Records (EHRs) have become the preferred choice for hospitals due to their efficiency in organizing patient records. EHR, containing digitized and systematized patient information along with medical and treatment histories, offer a comprehensive overview of patient care, thereby enhancing the hospital medical care through error minimization (HealthIT.gov, 2017; Shickel et al., 2018). Automated medical prognosis utilizing EHR can aid physicians in more accurate disease diagnosis and can greatly improve their working efficiency. Thus, a variety of research has been focused on developing deep-learning methodologies to improve the prognosis accuracy. For example, Andre Esteva & Dean (2019) employed natural language processing to extract clinical notes, and used auto-encoders to predict specific diagnoses.

Recent research, as summarized in Chuizheng Meng & Liu (2022), has focused on deep learning models within the context of a specific EHR dataset, MIMIC-III. This work delineates four models for predicting sequential clinical events: Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), Unsupervised Embedding, and Generative Adversarial Networks (GANs). Herein, we concentrate on the most influential model within this domain: RNNs and its variants, such as Gated Recurrent Units (GRUs) and Long Short-Term Memory (LSTM) networks. To contextualize our findings, we compare the prediction results with Minimal GRU (MGU), as proposed by LIGDoctor (Rodrigues-Jr et al., 2019), a simplified GRU structure designed by Zhou et al. (2016).

Additionally, as noted in Chuizheng Meng & Liu (2022), predictive models trained on the MIMIC-IV dataset display performance disparities among diverse demographic subgroups, such as ethnicity, gender, and age. To mitigate potential dataset biases and ensure algorithmic fairness, we focus solely on patients aged 15 years and older in our analysis.

This paper aims to improve the prediction accuracy of patients' future diagnoses using Recurrent Neural Networks (RNNs) within the MIMIC-III EHR dataset. Our main contributions are outlined as follows:

- We propose a new model architecture based on the basic GRU model, incorporating various deep learning techniques for predicting sequential clinical events. Unlike previous studies that use Theano package, our implementation is in TensorFlow, which increases effectiveness and readability.
- Our model demonstrates improved prediction rates for patients' next diagnoses, achieving a recall@30 score of 0.82, surpassing previous research conducted on the MIMIC-III dataset. Furthermore, our model exhibits robust performance in predicting other clinical events such as patients' next procedures.

The structure of the paper is organized as follows: Section 2 provides a review of related work, which serves as the foundation for our proposed approach; Section 3 details our dataset pre-processing methods and the key strategies employed in data manipulation; Section 4 elucidates the model architectures and vital components we used to enhance prediction performance; Section 5 presents the experimental setup and a comparative evaluation of results between baseline models and our proposed improvements;

Finally, Section 6 discusses the conclusions drawn from our study and outlines potential works for future research.

2. Related Work

2.1. DoctorAI

DoctorAI (Edward Choi & Sun, 2015) employed a Recurrent Neural Network (RNN) model, specifically Gated Recurrent Units (GRU), in conjunction with multi-label classification to simultaneously predict patients' subsequent diagnoses, procedures, and visit times using the dataset from Sutter Health Palo Alto Medical Foundation. The study found that an RNN model with two hidden layers and Skip gram vectors yielded the best performance, achieving a recall@30 of 0.80 for diagnosis prediction. However, our research is solely concentrated on diagnosis prediction. We compare various RNN models and MLP model, and explore the performance of additional techniques with the aim of enhancing the recall@30 prediction rates. Our goal is to surpass the benchmark set by Doctor AI and contribute to the advancement of predictive healthcare models.

2.2. LIGDoctor

LIGDoctor (Rodrigues-Jr et al., 2019) adopted certain aspects of the DoctorAI article and explained the process of translating ICD-9 encoding to Clinical Classifications Software (CCS) encoding in detail. This translation significantly reduced the number of classes in diagnoses and is more suitable for MIMIC-III dataset. Furthermore, the author implemented batch processing to handle sequences of patients' diagnoses in each admission, as opposed to using a one-hot vector per iteration. This approach largely simplified the network of models. MinGRU emerged as the top performer, achieving a recall@30 rate of 0.73 when predicting patients' subsequent diagnoses. In our research, we employ the data preprocessing methods introduced by LIGDoctor.

2.3. Theano and TensorFlow

While the preceding papers are commendable, it's important to note that they all utilized the Theano package for coding[1]. However, TensorFlow has a significant advantage in several areas: it enjoys a large and active development community, and is designed with more user-friendly APIs and tools, which simplifies the process of building and training models. Furthermore, TensorFlow outperforms Theano in terms of efficiency, particularly when handling large-scale data and complex models (Chung et al., 2017). Therefore, in our work, we've chosen to employ structures from Keras 3.0 in our models, which greatly enhances both readability and comprehension.

[1] https://github.com/mp2893/doctorai; https://github.com/jfrjunio/LIG-Doctor

3. Data and Preprocessing

3.1. Dataset

In this study, we utilize the MIMIC-III clinical database, which comprises deidentified health-related data associated with over forty thousand patients in Beth Israel Deaconess Medical Center between 2001 and 2012 (Johnson et al., 2016). As our primary objective is to predict patients' subsequent diagnoses, our focus is solely on the admission and diagnosis tables.

3.2. Preprocessing

3.2.1. Cohort selection

Firstly, our analysis is limited to patients who are over 15 years old. This is because certain health conditions, such as chronic diseases, tend to be more common or better documented in adult populations (Gangavarapu et al., 2020). By focusing on data from patients older than 15, our model is able to concentrate on predicting diagnoses in adulthood, which could potentially result in more precise predictions.

Secondly, we exclude all patients who only have one admission record. A single record might indicate an unintended visit, and that the patient would not return, which are not meaningful for prediction purposes. Furthermore, since there are no subsequent visit diagnosis records for these patients, it poses a challenge for our model during training and testing. Therefore, it is crucial to disregard these cases.

3.2.2. ICD-9 and CCS mapping

The International Classification of Diseases, Ninth Revision (ICD-9) contains a numerical list of disease code numbers. In our dataset, there are 6985 unique ICD-9 codes, which is too large for the MIMIC-III dataset to handle for classification tasks (Edward Choi & Sun, 2015). The Clinical Classifications Software (CCS) for ICD-9 is a well-known diagnosis and procedure categorization scheme, which can group some ICD-9 codes together. We use an existing mapping table[2] to map the diagnosis codes from ICD-9 to CCS encoding. After implementing this transformation, the number of unique diagnosis codes is condensed to 128. This reduction could notably boost the performance of our model.

3.2.3. Batch processing

We have to merge the admission and diagnose tables together and transfer the sequences of CCS code into certain

[2] https://github.com/svenhalvorson/icd_ccs_elix/blob/master/Multi_Level_CCS_2015/ccs_multi_dx_tool_2015.csv

numerical language that our model could handle with. Here is an example that we did in our preprocessing step:

Table 1. Examples of patients' diagnoses

Patients	1st Admission	2nd	3rd
A	Bacterial infection[3] Eye disorders[87]	[5] [87]	[6]
B	Spinal cord injury [15]	[13]	/
C	Burns [110] Open wounds [115]	[114]	[114]

We change them into a large matrix:

```
[
[[3, 87],[5,87],[6]],
[[15],[13]],
[[110,115],[114],[114]]
]
```

The first line represents the diagnosis codes for three visits of Patient A, the second line is for Patient B, and so on.

Padding Sequences. Models based on Recurrent Neural Networks (RNNs) requires input sequences that maintain a uniform length. In the absence of padding, sequences with varying lengths could lead to inconsistent batch shapes, which would make it challenging to execute computations in parallel across multiple sequences (Goodfellow et al., 2016). In our dataset, the maximum number of visits by a single patient is recorded to be 28. Consequently, we implement left-sided padding on our time-related data, ensuring that the data for each patient is represented as a sequence of 28 visits:

```
[
[[3, 87],[5,87],[6],[0],.....,[0]],
[[15],[13],[0],.............,[0]],
[[110,115],[114],[114],[0],...,[0]],
...
]
```

One-hot Vector. Additionally, we want to use a one-hot vector to represent the diagnoses in each visit. Thus, we will convert a visit like [3, 87] into a 128-length vector since we have 128 unique CCS codes. This vector will contain all zeros, except the 3rd and 87th positions, which will be set to 1. For the original vector $\boldsymbol{z} = [z_1, z_2, ..., z_M]$ and the new vector $\boldsymbol{x} \in \mathbb{R}^{128}$, for any $j \in \{1, 2, 3, \ldots, 128\}$, $m \in \{1, 2, 3, \ldots, M\}$:

$$x_j = \begin{cases} 1 & \exists\, m,\ j = z_m \\ 0 & \exists\, m,\ j \neq z_m \end{cases} \tag{1}$$

4. Methods

This section describes the RNN variant models for sequential prediction of clinical events. We will introduce crucial components of the model, explore the entire RNNs architecture, and elucidate how we predict diagnoses and procedural events using the RNNs model.

4.1. Problem setting

The input data for the model can be viewed as a series of patient observations. For each patient i, the observation comprises a list of visits , where each visit j consists of a list of codes t (e.g.diagnosis codes or procedure codes).

Then each visit list mentioned above is represented as a multi-hot vector $\mathbf{x}_{i,t}$, where i is the i^{th} patient, t is the time t^{th} admission, before training the RNN model. The multi-hot vector $\mathbf{x}_{i,t} \in \{0, 1\}^j$ represents the codes assigned at the t^{th} visit for the corresponding patient, where j is the total number of unique diagnose or procedure codes. We also pad j-dimensional zero vectors as visits to ensure that each patient has the same number of visits. We extract the mask matrix to indicate which visits are filled with meaningless vectors. This ensures the model can be implemented successfully.

The label variable $\boldsymbol{y_{i,t}}$ shares the same format as feature variable $\boldsymbol{x_{i,t}}$. However, the feature variable is one time step ahead of the label variable, that is $\boldsymbol{y_{i,t}} = \boldsymbol{x_{i,t+1}}$. This setup is designed for the model to predict the next visit.

4.2. Neural Network Architecture

We first introduce our decision of neural network and then elaborate on the vital architecture components, followed by a detailed demonstration of our determination of neuron nodes and layers in the hidden layer and output layer, as well as the loss and optimizer. Finally, we briefly summarize the optimal neural network architecture.

4.2.1. Macro Architecture Decision

We begin with MLP as the baseline, a standard feedforward neural network and a common benchmark model compared to other complicated models. Then, we explore various RNN architectures to enhance accuracy, leveraging their ability to handle sequential data with temporal dependencies. Initially, we implement LSTM to better capture long-term dependencies compared to simple RNNs. Next, we experiment with GRU for improved computational efficiency and effective long-term dependency modelling, offering fewer parameters and reducing overfitting risk. Due to the aforementioned benefits and better prediction performance, we use the GRU as our core architecture. Its equations are as follows (Chung et al., 2014):

$$
\begin{aligned}
\boldsymbol{r}_t &= \sigma(\boldsymbol{x}_t \boldsymbol{W}_{xr} + \boldsymbol{h}_{t-1} \boldsymbol{W}_{hr} + \boldsymbol{b}_r) \\
\boldsymbol{z}_t &= \sigma(\boldsymbol{x}_t \boldsymbol{W}_{xz} + \boldsymbol{h}_{t-1} \boldsymbol{W}_{hz} + \boldsymbol{b}_z) \\
\tilde{\boldsymbol{h}}_t &= \tanh(\boldsymbol{x}_t \boldsymbol{W}_{xh} + (\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1}) \boldsymbol{W}_{hh} + \boldsymbol{b}_h) \\
\boldsymbol{h}_t &= \boldsymbol{z}_t \odot \boldsymbol{h}_{t-1} + (1 - \boldsymbol{z}_t) \tilde{\boldsymbol{h}}_t
\end{aligned} \tag{2}
$$

where $\boldsymbol{r}_t$, $\boldsymbol{z}_t$, $\tilde{\boldsymbol{h}}_t$, and $\boldsymbol{h}_t$ respectively represent the reset gate, update gate, candidate hidden state, and the hidden state, all at time step t.

4.2.2. Vital Architecture Components

The most important parts of the model, which improve the prediction performance significantly, are the Bidirectional pass, Attention, and Time-distributed parts.

Bidirectional Pass. Beside the feed-forward-only GRU, we tested the bidirectional GRU, which combines the forward and backward GRU hidden states. In the forward GRU, the input sequence is fed following the order of time steps. At each time step, the hidden layer is calculated based on the current input and previous hidden state, so the overall forward hidden state contains information from the current and previous time steps. On the contrary, the input sequence is fed in a reverse time order in the backward GRU so that it can better capture the current and future time steps. Once complete, the forward and backward hidden states are concatenated into a single hidden layer. We restate the architecture mathematically in the following equations (Fang et al., 2021). A significant benefit of a bidirectional network compared to a feed-forward-only network is that it processes the time sequence bidirectionally, previous, current, and future patterns can be considered simultaneously.

$$
\begin{aligned}
\overrightarrow{\boldsymbol{h}}_t &= \overrightarrow{GRU}(\boldsymbol{x}_t, \overrightarrow{\boldsymbol{h}}_{t-1}) \\
\overleftarrow{\boldsymbol{h}}_t &= \overleftarrow{GRU}(\boldsymbol{x}_t, \overleftarrow{\boldsymbol{h}}_{t+1}) \\
\boldsymbol{h}_t^{bi} &= \overrightarrow{\boldsymbol{h}}_t \boldsymbol{W}_f + \overleftarrow{\boldsymbol{h}}_t \boldsymbol{W}_b + \boldsymbol{b}^{bi}
\end{aligned} \tag{3}
$$

where $\overrightarrow{\boldsymbol{h}}_t$ and $\overleftarrow{\boldsymbol{h}}_t$ are forward and backward candidate hidden state respectively.

Self-attention. The role of self-attention in RNN is to enhance the model's focus on different parts of the input sequence, enabling it to effectively handle long sequences and important information, thereby significantly improving the model's performance in sequence tasks. Regarding our dataset, incorporating a self-attention component allows the model to identify the significant connections between different admissions of a single patient. We reiterate the mathematical formulation of self-attention as follows.

$$
\begin{aligned}
\boldsymbol{h}_{t,t'} &= \tanh\left(\boldsymbol{x}_t^T \boldsymbol{W}_t + \boldsymbol{x}_{t'}^T \boldsymbol{W}_x + \boldsymbol{b}_t\right) \\
\boldsymbol{e}_{t,t'} &= \sigma\left(\boldsymbol{W}_a \boldsymbol{h}_{t,t'} + \boldsymbol{b}_a\right) \\
\boldsymbol{a}_t &= \text{softmax}(\boldsymbol{e}_t) \\
\boldsymbol{l}_t &= \sum_{t'} \boldsymbol{a}_{t,t'} \boldsymbol{x}_{t'}
\end{aligned} \tag{4}
$$

where $\boldsymbol{a}_t$ represents the attention score, indicating the importance of each time step relative to the target time step. $\boldsymbol{l}_t$ is the weighted sum, serving as the model's final response to the current query(CyberZHG).

Time Distributed. We also configure the GRU parameters with `return_sequences=True`, ensuring that it produces one hidden state $\boldsymbol{h}_t$ for each time step input. Additionally, we introduce a time-distributed Layer to every temporal slice, allowing each timestep to generate an output(Montaha et al., 2022). For instance, for one-dimensional input and output over $T = 3$ dates, where the output is one timestep ahead of the input, we can represent the input as $(\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2) \in \mathbb{R}^3$, and the output as $(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) \in \mathbb{R}^3$. And for each timestep, we can suppose that the equation is $\boldsymbol{x}_{t+1} = \tanh(\boldsymbol{W}_{x_{t+1}} \boldsymbol{x}_t + \boldsymbol{b}_{x_{t+1}})$. The model can be represented as follows:

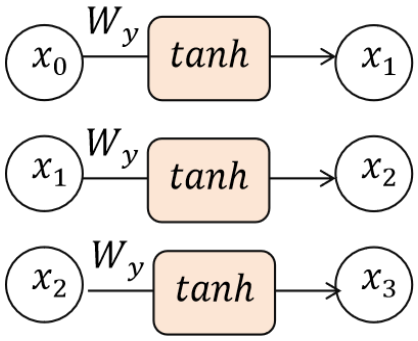

Figure 1. Time distributed structure

4.2.3. Determination of Micro Architecture

Hidden Layer. We did extensive testing regarding the number of neurons in each layer, activation functions, and the number of hidden layers. To find an optimal number of neurons in the hidden layer, we tested a variety of neuron numbers (e.g., 100, 200, 300, 400; 128, 256, 512, 1024). The best performance was observed at 256 - 2 times the number of 128 unique diagnosis codes. Next, we compared four activation functions in the hidden layer. The leaky Rectified Linear Unit (LReLU) function outperforms ReLU, Sigmoid, and Tanh functions, reaching the best performance. Additionally, we also tested two and three hidden layers, each layer with 128, 256, and 512 neuron nodes, but the performance deteriorates as the number of layers increases.

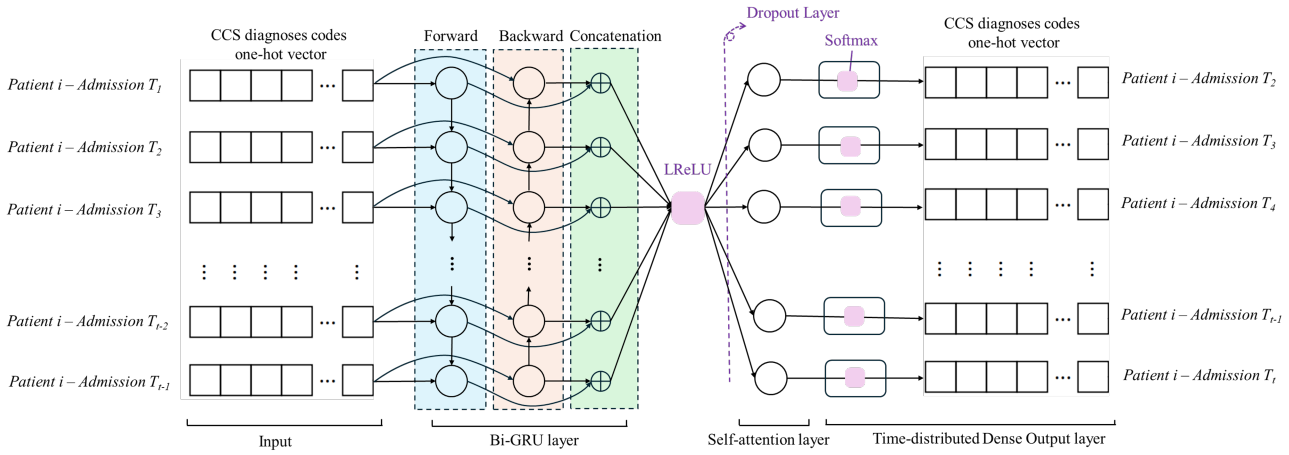

Figure 2. The Proposed architecture

This may be because a relatively small number of MIMIC-III dataset instances and simplified CCS diagnosis codes do not sustain a more complex network. This result aligns with LIGDoctor (Edward Choi & Sun, 2015). As a result, our final model ends up with one bidirectional GRU hidden layer. Before feeding the sequences to the output layer, we use a dropout technique with a rate of 0.1 to prevent the overfitting problem.

Output Layer. Since our task is to predict which of the 128 diagnosis codes each patient belongs to at the next admission, we use a dense layer with 128 neurons to map the sequence to a probability output space and use the softmax activation function to output the probabilities of belonging to those diagnosis codes. Note that the time-distributed function is added to this layer to generate an output for each time step or each admission of the patients.

Loss and Optimizer. We use categorical cross entropy as the loss function because our classification task is a multi-class classification problem. Say $y_{i,t,j}$ represents the ground truth diagnosis code for patient i at time steps t, and $\hat{y}_{i,t,j}$ represents the predicted probability of the observation belonging to the diagnosis code j. Then, the cross-entropy loss is formulated as follows. Note that $\boldsymbol{y}_t = \boldsymbol{x}_{t+1}$.

$$Loss(\boldsymbol{y}_{i,t}, \hat{\boldsymbol{y}}_{i,t}) = -\sum_{j=1}^{C} y_{i,t,j} \log(\hat{y}_{i,t,j}) \tag{5}$$

We choose the Adam optimizer because it automatically adjusts the learning rate during training, enabling to handle different learning rates for different parameters and speed up the convergence process (Kingma & Ba, 2017).

4.2.4. Summary of the Optimal Neural Network

Our best model (see Figure 2.) includes four layers: one input layer, one bidirectional GRU layer, one self-attention layer and one time-distributed dense output layer. In the bidirectional GRU layer, we assigned 256 neuron nodes (i.e., two times the number of unique CCS diagnosis codes) with the LReLU activation function and `return_Sequence=True`, followed by a 0.1 rate of dropout and self-attention layer. As per the output layer, a time-distributed dense layer with 128 neurons and softmax activation was employed. After building the model, we optimize all parameters using Adam to minimize the categorical cross-entropy loss.

5. Experiments

In this section, we will present the evaluation matrices and elaborate the details of experiment procedures and result proofs for the methodology part. These results are crucial for determining our best model to predict medical records for the next admission based on existing records. The codes and data of our experiments are provided in GitHub[3].

5.1. Experiment setup

To compare the models, we conducted 10 epochs of training for each model, when the model converges well. The dataset was divided into training and test sets, with a split ratio of 85% for training and 15% for testing. Our model was implemented using Keras with TensorFlow in our experiments.

[3] https://github.com/lse-st456/st456-wt2024-project-aidefeatshuman/tree/main/Submission

5.2. Evaluation metrics

We evaluate our model performance in predicting diagnosis CCS codes by using the top-k recall.

Recall@k/ Top-k recall. It is a commonly used metric to evaluate the performance of recommendation systems or classification models in tasks where the objective is to retrieve the top k relevant categories for a given instance. It measures the proportion of relevant items correctly identified within the first k recommendations out of the total number of relevant items in the dataset (Aher, 2023). In essence, this metric indicates the model's ability to successfully locate relevant items.

The concept of top-k recall rate is similar to the approach used by doctors in differential diagnosis. During this approach, doctors list the most likely diagnoses and customize treatments according to the patient's condition (Edward Choi & Sun, 2015) . Therefore, a high top-k recall rate indicates effective diagnostic skills comparable to those of doctors. For our problem, the top-k recall rate is therefore a useful evaluation metric. The formula can be shown as:

$$\text{Top-k recall} = \frac{\text{\# of True Positive in the top } k \text{ predictions}}{\text{\# of True Positive}} \tag{6}$$

Our data consist of 128 CCS diagnostic codes. The output of our model is a one-dimensional vector of length 128, where each value in the vector represents the probability of the input data belonging to one of the 128 label categories, ranging from 0 to 1. As a result, if the model predicts ten codes as output responses, the labels with the highest 10 probability values in the vector can be chosen as the top ten predicted results. In our evaluation part, the Top-10, Top-20, and Top-30 evaluation metrics are used for each model respectively.

5.3. Comparison of macro architectures

The initial round of experiments can be referred to in Section 4.2.1, where we compared the top-k recall of Multilayer Perceptron (MLP), Feed Forward GRU(FF-GRU), Feed Forward LSTM(FF-LSTM), Bidirectional GRU(Bi-GRU), and Bidirectional LSTM(Bi-LSTM). We used the same hyperparameters approach for each model (see Section 4.2.3) to ensure homogeneity. All models were trained using the same training data (as previously stated). The aggregated results are represented in the Table 2, which shows that the Bi-GRU has the best performance, with a recall rate of 77% for the top 30. Notably, the Bi-LSTM has comparable performance, with a recall rate of 77%. This may due to the addition of a bi-directional layer allows the output layer to acquire information about both past and future states. However, the Bi-GRU outperforms the Bi-LSTM in both top-10 and top-20 recall rates. We were surprised to find that the performance of MLP was comparable to that of FF-LSTM and FF-GRU, with top-30 recall accuracy as high as 67%. The reason may be that most of the patients in the MIMIC-III dataset had only two admissions, in which case the function of temporal perception had little effect on the results (Rodrigues-Jr et al., 2019). After observing these results, we decided to choose three models, MLP, Bi-LSTM and Bi-GRU, for further investigation.

Table 2. Top-k recall rates for different models

Model	**Top-10**	**Top-20**	**Top-30**
MLP	0.36	0.56	0.67
FF-GRU	0.37	0.56	0.67
FF-LSTM	0.36	0.54	0.65
Bi-GRU	**0.49**	**0.67**	**0.77**
Bi-LSTM	0.47	0.66	0.77

5.4. Comparison of neurons and layers

We continued to explore various combinations of layer depths and neuron counts alongside the initially defined hyperparameters. Our criterion for comparing model performance was the top-30 recall. We used with up to three layers and 128, 256, 512 neurons for each model. The number of neurons is a multiple of the size of the unique CCS diagnosis codes, as explained in Section 4.2.3. Table 3 shows that, as stated before, model performance decreases with the number of layers, especially for the Bi-LSTM model. We also found that the required processing time increases with the number of neurons. According to the Table 3, for each model, adding more neurons from 128 to 256 improves the top-30 recall noticeably. However, increasing the number of neurons from 256 to 512 produces a very tiny performance gain and and it took significantly more time to train the models. These findings suggest that the Bi-GRU model consistently outperforms others for a given layer and neuron configuration. After taking all of these factors into account, We decided to use a single hidden layer with 256 neuron units consistently and chose Bi-GRU as the baseline model for our architecture.

5.5. Comparison of micro architectures

After the determination of the number of neurons and layers, as well as the Bi-GRU as the baseline model, more advanced techniques were applied to enhance performance. We tried using different activation functions. According to Table 4, we found that only LReLU improved the model performance, with a top-10 recall higher than any other models. Based on this architecture, we also tried other techniques,

Table 3. Comparisons between different layers and neurons

Layers	Neurons	MLP	Bi-GRU	Bi-LSTM
1	128	0.66	0.73	0.72
	256	0.67	**0.77**	0.77
	512	0.68	0.77	0.77
2	128	0.66	0.72	0.72
	256	0.68	**0.76**	0.72
	512	0.67	0.75	0.73
3	128	0.65	0.72	0.66
	256	0.66	**0.75**	0.66
	512	0.67	0.74	0.67

including adding another self attention layer. By focusing on different parts of the input sequence, the model can better understand the relationship between the elements, thus improving the performance. As demonstrated in Table 4, the enhanced model performs much better, with top-30 recall reaching 82% and top-10 and top-20 recall increasing by almost 10% respectively.

Table 4. Top-k recall rates for improved models

Model	Top-10	Top-20	Top-30
Bi-GRU + tanh	0.49	0.67	0.77
Bi-GRU + sigmoid	0.4	0.59	0.7
Bi-GRU + ReLU	0.45	0.62	0.73
Bi-GRU + LReLU	0.49	0.67	0.76
Bi-GRU + LReLU + Self Attention	**0.55**	**0.73**	**0.82**

Figure 3 illustrates the impact of the self-attention layer on performance enhancement. In the initial experiment, we trained the Bi-GRU model for 20 epochs (indicated by the red curve). Then in the second experiment, we augmented the Bi-GRU model with a self-attention layer and trained the enhanced model for 20 epochs (indicated by the blue curve). We observed a significant enhancement in prediction performance upon adding the self-attention layer, resulting in over a 5% improvement in top-30 recall rate.

5.6. Comparison to literature

Overall, our model builds upon the foundation of the Doctor AI (Edward Choi & Sun, 2015), incorporating bidirectional layers and introducing a self-attention mechanism. This enhancement has led to improved performance in predicting the diagnosis of patients upon their next hospital admission. In the subsequent section, we only compared our model with the final optimized model from LIGDoctor (Rodrigues-Jr et al., 2019), utilizing the same MIMIC-III dataset. As

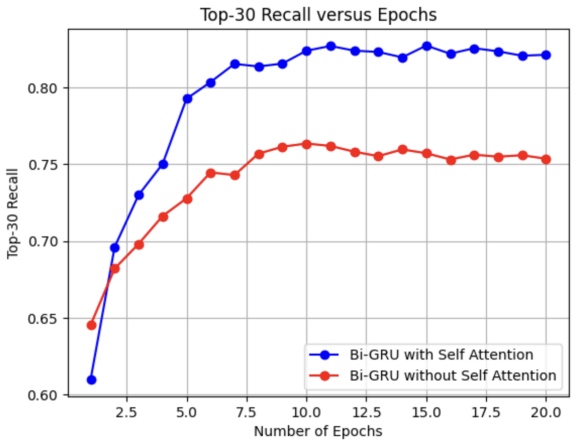

Figure 3. Top-30 recall versus epochs

shown in Table 5, our enhanced model outperforms LIGDoctor's model in terms of top-10, top-20, and top-30 recall rates.

Table 5. Top-k recall rates of LIGDoctor model versus our model

Model	Top-10	Top-20	Top-30
LIGDoctor	0.51	0.68	0.76
Our Model	**0.56**	**0.74**	**0.82**

5.7. Our best model on predicting procedures

To assess the robustness of our model, we used it to predict the procedure code of the patient's next hospitalization. Its core idea is similar to our diagnosis prediction, which is to forecast the CCS code of the procedure using previous admission records. The findings are presented as follows:

Table 6. Prediction result of patients' next procedures using our best model

	Top-10	Top-20	Top-30
Our model	0.50	0.65	0.72

Specifically, the model reaches 72% at top-30 recall rate, which indicates that the model is fairly accurate in forecasting the procedure code, and its robustness has been demonstrated.

6. Conclusion

Our study presents a sequential prediction model for clinical events, specifically designed to forecast patients' future diagnoses based on their existing diagnosis codes. Using the comprehensive and openly accessible medical dataset

MIMIC-III, which provides sequential diagnosis data for thousands of patients, we developed a modified Bi-GRU model, intricately built upon the classic GRU architecture. The proposed architecture is equipped with advanced deep learning methodologies, including Bidirectional processing to deal with the sequential data in both forward and backward directions. Moreover, the integration of a self-attention mechanism enables the model to figure out which admissions of one patient are important to each other. Additionally, the utilization of Time Distributed component enables the model to generate outputs at each time step, ensuring its predictive capabilities. Moreover, we did extensive experiments to find the optimal hyper-parameters to improve the prediction performance. Finally, our proposed architecture reaches a recall@30 of 0.82. Our model not only provides clinically meaningful diagnosis predictions, offering valuable reminders to patients, but also enhances the diagnostic process for human doctors. Additionally, it offers insights into the future application of artificial intelligence in medical events prediction, paving the way for personalized healthcare recommendations automated for individuals.

There is still much space left for the future research:

- Firstly, this study utilized existing diagnosis codes for predicting future diagnoses. Integrating additional data sources like textual analysis of physicians' diagnostic notes could enhance predictive power (Juhn & Liu, 2020).
- Secondly, our study employed CCS, a well-established disease classification system. Future research could delve into more granular disease diagnosis codes. While the MIMIC-III dataset used provided valuable patient cases, its scale is not be suitable for predicting finer disease diagnoses (e.g., ICD9), that is also the reason we use CCS codes. Future studies could use larger clinical datasets for more granular predictions.
- Thirdly, while our study predicted future diseases, it didn't explore potential onset times. Future research could investigate onset times based on patient admission times for better health monitoring and management.
- Lastly, our data examined individuals aged 15 and above broadly. Future studies could refine population stratification by age, income, ethnicity, or geography for more accurate predictions.

7. Contributions

Everyone contributed equally to this report. Everyone did the literature review together and decided the direction of this research. Person 1 (ID: 25018) and Person 2 (ID: 34896) focused on the problem analysis, initial model architecture design, and data processing. Person 3 (ID: 33608) and Person 4 (ID: 27508) contributed to refining the model architectures, adjusting layers and parameters for enhanced performance, and comparing model performances. Additionally, all team members contributed to the writing and development of their respective sections of the article, ensuring a comprehensive and cohesive presentation of the research findings.

References

Aher, P. Evaluation Metrics for Recommendation Systems — An overview. 8 2023. URL https://towardsdatascience.com/evaluation-metrics-for-recommendation\-systems-an-overview-71290690ecba.

Andre Esteva, Alexandre Robicquet, B. R. V. K. M. D. K. C. C. C. G. C. S. T. and Dean, J. A guide to deep learning in healthcare. *Nature Medicine*, 25:24–29, 2019. doi: 10.1038/s41591-018-0316-z.

Chuizheng Meng, Loc Trinh, N. X. J. E. and Liu, Y. Interpretability and fairness evaluation of deep learning models on mimic-iv dataset. *Scientific Reports*, 12(1): 7166, May 2022. doi: 10.1038/s41598-022-11012-2.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Chung, Y., Ahn, S., Yang, J., and Lee, J. Comparison of deep learning frameworks: about theano, tensorflow, and cognitive toolkit. *Journal of Intelligence and Information Systems*, 23(2):1–17, 2017.

CyberZHG. keras-self-attention/README.zh-CN.md at master · CyberZHG/keras-self-attention. URL https://github.com/CyberZHG/keras-self-attention/blob/master/README.zh-CN.md.

Edward Choi, Mohammad Taha Bahadori, A. S. W. F. S. and Sun, J. Doctor AI: Predicting clinical events via recurrent neural networks. *arXiv (Cornell University)*, 1 2015. doi: 10.48550/arxiv.1511.05942. URL https://arxiv.org/abs/1511.05942.

Fang, Y., Yang, S., Zhao, B., and Huang, C. Cyberbullying detection in social networks using bi-gru with self-attention mechanism. *Information*, 12(4):171, 2021.

Gangavarapu, T., Krishnan, G. S., Kamath, S., and Jeganathan, J. Farsight: long-term disease prediction using unstructured clinical nursing notes. *IEEE Transactions on Emerging Topics in Computing*, 9(3):1151–1169, 2020.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

HealthIT.gov. Benefits of electronic health records, 2017. URL https://www.healthit.gov/topic/health-it-and-health-information-\exchange-basics/benefits-ehrs.

Johnson, A. E., Pollard, T. J., Shen, L., Lehman, L.-w. H., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Anthony Celi, L., and Mark, R. G. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9, 2016.

Juhn, Y. and Liu, H. Artificial intelligence approaches using natural language processing to advance ehr-based clinical research. *Journal of Allergy and Clinical Immunology*, 145(2):463–469, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization, 2017.

Montaha, S., Azam, S., Rafid, A. R. H., Hasan, M., Karim, A., and Islam, A. Timedistributed-cnn-lstm: A hybrid approach combining cnn and lstm to classify brain tumor on 3d mri scans performing ablation study. *IEEE Access*, 10:60039–60059, 06 2022. doi: 10.1109/ACCESS.2022.3179577.

Rodrigues-Jr, J. F., Spadon, G., Brandoli, B., and Amer-Yahia, S. Patient trajectory prediction in the mimic-iii dataset, challenges and pitfalls. 2019. URL https://arxiv.org/abs/1909.04605.

Shickel, B., Tighe, P. J., Bihorac, A., and Rashidi, P. Deep ehr: A survey of recent advances in deep learning techniques for electronic health record (ehr) analysis. *IEEE Journal of Biomedical and Health Informatics*, 22(5): 1589–1604, 2018. doi: 10.1109/JBHI.2017.2767063.

Zhou, G., Wu, J., Zhang, C.-L., and Zhou, Z. Minimal gated unit for recurrent neural networks. *International journal of automation and computing*, 13(3):226–234, 6 2016. doi: 10.1007/s11633-016-1006-2. URL https://doi.org/10.1007/s11633-016-1006-2.