CLINICAL-GAN: TRAJECTORY FORECASTING OF CLINICAL EVENTS USING TRANSFORMER AND GENERATIVE ADVERSARIAL NETWORKS

A Thesis
Submitted to the Faculty of Graduate Studies and Research
In Partial Fulfillment of the Requirements
for the Degree of
Master of Science
in
Computer Science
University of Regina

By
Vignesh Shankar
Regina, Saskatchewan
January 2022

Copyright © 2022: Vignesh Shankar
Vignesh Shankar, candidate for the degree of Master of Science in Computer Science, has presented a thesis titled, *Clinical-Gan: Trajectory Forecasting of Clinical Events Using Transformer and Generative Adversarial Networks*, in an oral examination held on December 17, 2021. The following committee members have found the thesis acceptable in form and content, and that the candidate demonstrated satisfactory knowledge of the subject material.

External Examiner: *Dr. Kin-Choong Yow, Software Systems Engineering*

Supervisor: *Dr. Alireza Manashty, Department of Computer Science*

Committee Member: *Dr. Howard Hamilton, Department of Computer Science*

Chair of Defense: *Dr. Farshid Torabi, Petroleum Systems Engineering*

*via ZOOM Conferencing*
Abstract

Predicting the trajectory of a disease at an early stage can aid physicians in offering effective treatment and prompt care to patients. Previous work has used Electronic Health Record (EHR) data and leveraged sequence models to forecast patients’ diagnosis, procedure, and medication codes. However, current deep learning models have difficulty learning from such EHR data—which comprises multivariate time series and multimodal data distribution. We propose a novel method called Clinical-GAN to tackle this learning issue, which combines Transformer and Generative Adversarial Networks (GAN) to forecast diagnosis, procedure, and prescription drugs while maintaining the interpretability of the model’s outcome. A Transformer mechanism is used as a Generator to learn from existing patients’ medical history and is trained adversarially against a Transformer-based Discriminator. In addition, we used multi-head attention of the Generator network to explain the model’s outcome.

We evaluated our method using a publicly available dataset, Medical Information Mart for Intensive Care IV (MIMIC-IV) v1.0, with more than 500,000 visits completed by around 196,000 adult patients over an 11 year period from 2008-2019. Based on the patient’s medical codes of each visit, Clinical-GAN achieved 57.04% in Mean Average Recall (MAR)@250 and 76.57% in Mean Average Precision (MAP)@250, significantly outperforming baseline methods in forecasting the patient’s diagnosis codes for subsequent visits. In addition, our model also outperformed the existing
work in sequential disease prediction by achieving a 65.27% in MAR@60. We used cosine similarity to calculate the top 20 associated codes of diagnosis based on the learned representation of embedded medical codes. Then we projected them onto a two-dimensional map using t-Distributed Stochastic Neighbour Embedding (t-SNE). We observed that our proposed method effectively learns the correlations between medical codes from the two-dimensional map. Overall, Clinical-GAN has achieved higher accuracy in forecasting the patient trajectory and sequential disease prediction than baseline and existing work, as demonstrated through various experiments.
Acknowledgments

I want to thank my supervisor, Dr. Alireza Manashty, for his support, guidance, and trust in my capacity. Furthermore, I am grateful to my supervisor, the Faculty of Graduate Studies and Research, the Department of Computer Science, and the Mitacs for financial assistance in the form of a Teaching Assistantship, a Scholarship award, and a MITACS Research award. I would like to express my gratitude to Dr. Kin-Choong Yow and Dr. Howard Hamilton for serving on my examining committee and providing valuable feedback. I would also like to thank the Natural Science and Engineering Research Council of Canada (NSERC) for providing funding to Dr. Alireza, through which Dayne Blair was recruited as an intern to assist me in this research project. Finally, I greatly appreciate the help I received from Elnaz Yousefi and Dayne for their contribution in preparing the journal paper.
Dedication

To my love, Deepika Teegapuram, who motivated and encouraged me to take challenges. To my parents and sister for their constant love and support.
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>i</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>iii</td>
</tr>
<tr>
<td>Dedication</td>
<td>iv</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>v</td>
</tr>
<tr>
<td>List of Tables</td>
<td>viii</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xi</td>
</tr>
<tr>
<td>Abbreviations</td>
<td>xvi</td>
</tr>
<tr>
<td>List of Symbols and Notations</td>
<td>xix</td>
</tr>
<tr>
<td>Chapter 1: Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Motivation</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Challenges</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Research Problem</td>
<td>3</td>
</tr>
<tr>
<td>1.4 Approach Overview and Contributions</td>
<td>3</td>
</tr>
<tr>
<td>1.5 Thesis Organization</td>
<td>5</td>
</tr>
<tr>
<td>1.6 Summary</td>
<td>6</td>
</tr>
</tbody>
</table>
Chapter 2: Background and Related Works

2.1 Background ......................................................... 7
  2.1.1 Deep Neural Networks ..................................... 7
  2.1.2 Recurrent Neural Networks ............................... 9
  2.1.3 RNN Encoder-Decoder ..................................... 13
  2.1.4 Word Embeddings .......................................... 14
  2.1.5 Cosine Similarity .......................................... 16
  2.1.6 t-SNE .......................................................... 17
  2.1.7 Transformer ................................................... 19
  2.1.8 Generative Adversarial Networks ....................... 22

2.2 Related Works .................................................. 24
  2.2.1 Trajectory Disease Forecasting ............................ 25
  2.2.2 Sequential Disease Prediction ............................. 25

2.3 Summary .......................................................... 29

Chapter 3: Clinical-GAN

3.1 Introduction ....................................................... 30

3.2 EHR Data and Notation ........................................ 32
  3.2.1 Input/Output Representation .............................. 33

3.3 Clinical-GAN ..................................................... 35
  3.3.1 Discriminator ................................................ 39
  3.3.2 Generator ..................................................... 40

3.4 Interpretability .................................................. 43

3.5 Summary .......................................................... 43

Chapter 4: Experimental Results
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 Datasets</td>
<td>45</td>
</tr>
<tr>
<td>4.2 Forecasting Tasks</td>
<td>48</td>
</tr>
<tr>
<td>4.3 Baseline Methods</td>
<td>51</td>
</tr>
<tr>
<td>4.4 Evaluation Metrics</td>
<td>52</td>
</tr>
<tr>
<td>4.5 Results and Discussion</td>
<td>53</td>
</tr>
<tr>
<td>4.6 Summary</td>
<td>68</td>
</tr>
</tbody>
</table>

**Chapter 5: Conclusion and Future Work**

References

Appendix A: Hyperparameter Details

Appendix B: Input and Output Samples of Clinical-GAN

Appendix C: Disease-Associated Medical Codes and their t-SNE Representation

Publications

vii
List of Tables

4.1 Statistics of the overall dataset .............................. 46
4.2 Average statistics for each type of medical code assigned per admission. 46
4.3 Statistics at each stage of data preprocessing .................. 48
4.4 An example of input and output samples generated from a patient record consisting of four visits (patient record = V1, V2, V3, V4) for the Trajectory Forecasting (TF) task. The resulting samples are formatted based on Eq. 3.12 and Eq. 3.13 as explained in Section 3.2.1. 49
4.5 An example of input and output samples generated from a patient record consisting of four visits (patient record = V1, V2, V3, V4) for the Sequential Disease Prediction (SDP) task. The resulting samples are formatted based on Eq. 3.12 and Eq. 3.13 as explained in Section 3.2.1. 50
4.6 The total number of records for the TF and SDP tasks, after formatting the data as mentioned above and removing records with a sequence length greater than 500. Before deletion, the total number of records was 35,770. 50
4.7 MAR and MAP of algorithms in all three scenarios of the trajectory forecasting task, where FD is forecasting Dx code only, FDP is forecasting Dx and Px codes, and FDR is forecasting Dx, Px, and Rx codes. The values are given in percentages 53
4.8 Approximate randomization test results for statistical significance testing between our method and Transformer in all three scenarios using MAR@250 and MAP@250. .................................................. 54

4.9 MAR of algorithms in the sequential disease prediction task ................................................. 55

4.10 Accuracy of Clinical-GAN on the five least common diseases in MIMIC-IV for the TF task in the \( F_D \) scenario. .................................................. 57

4.11 Accuracy of Clinical-GAN on the five most common diseases in MIMIC-IV for the TF task in the \( F_D \) scenario. .................................................. 58

4.12 Top 20 associated medical codes for Asthma calculated using cosine similarity from our final embedding. The valid medical codes are underlined. The code descriptions shown in the table are CCS and CCSR descriptions. Some of the diagnoses and procedures can be repeated because we are considering both ICD-9 and ICD-10 codes. . 60

4.13 Top 20 associated medical codes for Breast cancer, Essential hypertension, and Heart failure. The valid codes are underlined. .................................................. 62

4.14 Total number of model parameters and training time of algorithms in all three scenarios. .................................................. 68

A.1 Hyperparameter details ................................................................. 89

A.2 Hyperparameter search space .......................................................... 90

B.1 Example 1 - Input for the \( F_D \) scenario ................................................. 92

B.2 Example 1 - Ground truth and Clinical-GAN’s forecasted data for the input mentioned in the Table B.1 ................................................. 93

B.3 Example 2 - Input for the \( F_D \) scenario ................................................. 94

B.4 Example 2 - Ground truth and Clinical-GAN’s forecasted data for the input mentioned in the Table B.3 ................................................. 95
B.5 Example 3 - Ground truth and Clinical GAN’s forecasted data for the
input in the $F_{DP}$ scenario. ........................................ 96
B.6 Example 4 - Ground truth and Clinical GAN’s forecasted data for the
input in the $F_{DP}$ scenario. In this example, Clinical-GAN forecasted
only one visit instead of two compared to the ground truth. However,
the model captured all the unique codes assigned to that patient relative
to the ground truth. Surprisingly, it gave all the ground truth’s second
visit codes of a patient as the first visit codes in the forecasted data. . 97
B.7 Example 5 - Ground truth and Clinical GAN’s forecasted data for the
input in the $F_{DPR}$ scenario. ........................................ 98
B.8 Example 6 - Ground truth and Clinical GAN’s forecasted data for the
input in the $F_{DPR}$ scenario. ........................................ 99
C.1 Top 20 associated medical codes for Diabetes mellitus, Type 2. . . . 101
C.2 Top 20 associated medical codes for Anxiety and fear-related disorders. 103
C.3 Top 20 associated medical codes for HIV infection. ...................... 105
C.4 Top 20 associated medical codes for Pancreatic disorders (excluding
diabetes). ................................................................. 107
C.5 Top 20 associated medical codes for Dysrhythmia. ..................... 109
C.6 Top 20 associated medical codes for Mood disorders. .................. 111
List of Figures

1.1  Graphical illustration of the Clinical-GAN architecture forecasting the patient trajectory, given a sequence of the patient’s past medical history. 4

2.1  Graphical illustration of Deep Neural Networks (created using [27]), which has an input layer (eight nodes), an output layer (two nodes), and two hidden layers (six and four nodes). . . . . . . . . . . . . . . . 8

2.2  a) A simplified RNN diagram in which A represents a neural network with n layers that takes input data $X_t$ and produces output data $h_t$. The loop indicates the passing of each timestep’s output of the neural network to the next timestep. b) A graphical representation of the RNN unrolled version. Image courtesy of Christopher Olah [31]. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

2.3  An LSTM diagram. a) The forget gate $f_t$ examines the previous hidden state $h_{t-1}$ and the current input $x_t$ to determine which part of $h_{t-1}$ information should be kept in the previous cell state $C_{t-1}$. b) The input gate $i_t$ determines which portion of new hidden state $C_t$ should be used in the $C_{t-1}$ based on $h_{t-1}$ and $x_t$. c) The operation of $f_t$ and $i_t * C_t$ updates the $C_{t-1}$, resulting in a new cell state $C_t$. d) The output gate $o_t$ will determine which information from $C_t$ should be retained in order to generate a new hidden state $h_t$. Image courtesy of Christopher Olah [31]. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
2.4 A GRU diagram. \( \hat{h}_t \) is the updated hidden state computed as part of the Reset gate. \( z_t \) is a part of the Update gate operation, which decides what new information is to be stored in the hidden state. \( r_t \) decides how much of the previous information should be retained with respect to the current input \( x_t \). Image courtesy of Christopher Olah [31]. . . 12

2.5 One-hot encoding of the sentence "Deep learning is the future". Assuming there are nine unique words in a vocabulary, we create a zero vector for each word in the sentence of length nine and add one at the index of the corresponding word. . . . . . . . . . . . . . . . . . . . . 14

2.6 A graphical illustration of word embedding on the sentence "Deep learning is the future". Each unique word is converted to integers, which will be passed through a hidden layer of size four comprised of only a linear function. This process will generate a four-dimensional embedding vector that represents each word in the sentence. Intuitively, an embedding vector is a lookup table where each row represents a corresponding word. . . . . . . . . . . . . . . . . . . . . . . . . . . . 15

2.7 a) Two vectors \( a \) and \( b \) that have a smaller angle between them will have a high degree of similarity. b) Two vectors with a 90-degree angle will have a similarity of zero, indicating that both vectors are independent. c) Opposing vectors have a similarity value of -1, indicating that the vectors are unrelated. Image courtesy of [37]. . . . . . . . . . . . . . . . . . . . . . . . . . . . 17

2.8 A t-SNE plot of MNIST handwritten digit dataset [39]. t-SNE reveals different clusters representing each digit. Each cluster is encoded with a unique color. Image courtesy of Christopher Olah [40]. . . . . . . . . . . . . . . . . . . . . . . . . . . . 18
2.9 a) A trefoil knot visualized in three dimensional map. b) A t-SNE representation of the trefoil knot in a two-dimensional plane which reflects t-SNE's performance in capturing local and global structures. Image courtesy of [41].

2.10 A Transformer architecture [24].

2.11 Overview of GAN. The Discriminator and the Generator are both neural networks. The output of the Generator is directly connected to the Discriminator's input. The Discriminator's classification is used to update the Generator's weights via backpropagation. Image courtesy of [43].

2.12 A GAN’s Discriminator is only a classifier. It attempts to classify between real and generated data. During the Discriminator’s training, the Generator’s weights remain constant, and the Discriminator uses Discriminator loss to update its weights through backpropagation. Image courtesy of [44].

2.13 The Generator part of GAN takes random noise as input and generates fake data. The generated data goes through a Discriminator network to classify whether the generated data is real or fake. The Generator’s loss is then used to update the weights of the Generator network through backpropagation. Image courtesy of [45].

3.1 Data representation of a single patient with T visits, separated by an EOV token. Each visit begins with a SOH token and ends with an EOH token. The medical codes are grouped and arranged in sequential order for each patient’s visit.

3.2 The Generator architecture of Clinical-GAN. It comprises Encoder and Decoder models.
3.3 The Discriminator architecture of Clinical-GAN. It comprises an Encoder model with a linear layer as the last component.

3.4 Flow through the Clinical-GAN model during training. Our model consists of two components: Generator (G) and Discriminator (D). G takes historical medical data (X) and forecasts the medical codes (Ŷ).

D takes real (X, Y) and generates (X, Ŷ) sequences and produces the probability to distinguish between real and generated ones.

4.1 This figure shows a visualization of the trajectory forecasting of a patient’s visit record. The x-axis corresponds to the input of past medical history comprised of medical codes, and the y-axis corresponds to the forecasted medical codes. The contribution of each medical code in the input towards predicting the variable in the output is summarized by the percentage along the x-axis. The intensity of the green colour represents the strength of the contribution.

4.2 Both plots represent the MAR@250 and MAP@250 scores of the trajectory forecasting in scenario $F_D$ using our method on the test set. The left plot shows the accuracy as a function of input sequence length, where the x-axis corresponds to the input data sorted by their length.

The right plot shows the accuracy as a function of the number of visits in the input data, where the x-axis corresponds to the input data sorted by their number of visits to the hospital.

4.3 t-SNE representation of the associated medical codes for Asthma.

4.4 t-SNE representation of the associated medical codes for Breast cancer.

4.5 t-SNE representation of the associated medical codes for Essential hypertension.

4.6 t-SNE representation of the associated medical codes for Heart failure.
4.7 t-SNE representation of all the codes together, where each diagnosis code is associated with at least one or more procedure and medication codes. This demonstrates that our proposed method is effective at learning the relationships between medical codes.

C.1 t-SNE representation of the associated medical codes for Diabetes mellitus, Type 2 mentioned in the Table [C.1] .................................. 102

C.2 t-SNE representation of the associated medical codes for Anxiety and fear-related disorders mentioned in the Table [C.2] ...................... 104

C.3 t-SNE representation of the associated medical codes for HIV infection mentioned in the Table [C.3] ........................................ 106

C.4 t-SNE representation of the associated medical codes for Pancreatic disorders (excluding diabetes) mentioned in the Table [C.4] ....... 108

C.5 t-SNE representation of the associated medical codes for Dysrhythmia mentioned in the Table [C.5] ........................................... 110

C.6 t-SNE representation of the associated medical codes for Mood disorders mentioned in the Table [C.6] ........................................ 112
List of Abbreviations

AI  Artificial Intelligence. 2, 30

BIDMC  Beth Israel Deaconess Medical Center. 45

CCS  Clinical Classification Software. ix, 46, 47, 60, 68, 70, 100

CCSR  Clinical Classifications Software Refined. ix, 46, 47, 60, 68, 70, 100

DNN  Deep Neural Networks. 8, 9, 29, 42, 51

DTB  Danish Disease Trajectory Browse. 25

EHR  Electronic Health Record. i, vi, 1, 3, 25, 27, 28, 31, 32, 67

GAN  Generative Adversarial Networks. i, vi, xiii, 3, 5, 7, 22, 24, 28, 29, 31, 37, 42, 43, 67, 70

GMM  Gaussian Mixture Model. 27

GPUs  Graphical Processing Units. 7, 51

GRU  Gated Recurrent Unit. xii, 9, 11, 13, 26, 29, 51, 68

GSN  Generic Sequence Number. 45, 46

HCUP  Healthcare Cost and Utilization Project. 46
ICA  Independent Component Analysis. 27

ICD  International Classification of Disease. ix, 5, 26, 45, 47, 60, 67, 68, 70, 91, 100

ICU  Intensive Care Units. 1, 45

LM  Life Model. 27

LSTM  Long Short-Term Memory. xi, 9, 13, 26, 27, 29, 35, 51

MAP  Mean Average Precision. i, viii, ix, xiv, 52–55, 57, 59, 68

MAR  Mean Average Recall. i, iii, viii, ix, xiv, 52, 55, 57, 59, 68

MIMIC-III  Medical Information Mart for Intensive Care III. 27, 28, 31

MIMIC-IV  Medical Information Mart for Intensive Care IV. i, ix, 5, 28, 32, 45, 47, 57, 58, 67, 68, 70

MLE  Maximum Likelihood Estimation. 36

MTE  Mean Tolerance Error. 27

NDC  National Drug Code. 45, 46

NLP  Natural Language Processing. 4, 14, 15, 19, 29, 71

PCA  Principal Component Analysis. 27

RETAIN  Reverse Time Attention Model. 26

RNN  Recurrent Neural Network. xi, xix, xxii, 3, 4, 9, 11, 13, 25, 29, 37, 51

SDA  Stack Denoising Autoencoders. 27

xvii
**SDP**  Sequential Disease Prediction. viii, 48, 51, 54, 68, 70

**SGD**  Stochastic Gradient Descent. 88

**t-SNE**  t-Distributed Stochastic Neighbour Embedding. iii, vii, xii–xv, 17, 19, 29, 60, 61, 63–68, 70, 100, 102, 104, 106, 108, 110, 112

**TF**  Trajectory Forecasting. viii, ix, 48–54, 57, 58, 68, 70, 88

**US**  United States. 1

**WGAN**  Wasserstein Generative Adversarial Networks. 22, 39, 40, 71

**WHO**  World Health Organization. 47
List of Symbols and Notations

Symbols

∈ Belongs to a particular set.

تفاعل Concatenation.

E Expected value for a probability distribution.

∀ For all.

∞ Infinity.

R Real numbers.

∇ Gradients

θ Weight parameters of a model.

Notations

A Attention head computed by the multi-head attention.

a Attention vector computed by the [RNN] Encoder-Decoder framework.

B A total number of medical codes.

C Cell state.
$D$ Discriminator network.

d Dimension of a vector or matrix.

$D_x$ Diagnosis code.

$Dec_g$ Decoder framework of the Generator network in the Clinical-GAN.

$Enc_d$ Encoder framework of the Discriminator network in the Clinical-GAN.

$Enc_g$ Encoder framework of the Generator network in the Clinical-GAN.

$F_D$ Forecasting Diagnosis codes.

$F_{DP}$ Forecasting Diagnosis and Procedure codes.

$F_{DPR}$ Forecasting Diagnosis, Procedure, and Medication codes.

$G$ Generator network.

$h$ Hidden states.

$I$ Interpretability.

$i$ A single patient.

$K$ Key vector computed by the Transformer.

$k$ A total number of heads in the multi-head attention network.

$K_d$ A total number of iterations for the Discriminator network to run.

$K_g$ A total number of iterations for the Decoder network to produce the next predicted variable.

$L_G$ Overall loss of the Generator network in the Clinical-GAN.
$L_{\text{Dis}}$ Loss of the Discriminator network in the Clinical-GAN.

$L_{G_1}$ Loss of the Generator network in the Clinical-GAN.

$L_{G_2}$ Negative log-likelihood loss.

$M$ Medical codes.

$m$ Size of a minibatch.

$N$ A total number of patients.

$P$ A total number of medical codes.

$P_x$ Procedure code.

$Q$ Query vector computed by the Transformer.

$r$ Reset gate.

$R_{x}$ Medication code.

$S$ Sequence length.

$s$ Hidden states of the Decoder.

$t$ Time steps.

$V$ A patient’s visit to a hospital.

$W$ Weight parameter of the multi-head attention network.

$X$ Input data for the Clinical-GAN.

$x'$ Generated sample by the Generator network.

$Y$ Ground truth data for the Clinical-GAN.
\( \hat{Y} \) The forecasted output of the Clinical-GAN.

\( Z \) Contextual representation computed by the Transformer’s Encoder network.

\( z \) Contextual representation computed by the RNN’s Encoder network in the Encoder-Decoder framework.
Chapter 1

Introduction

This chapter provides an overview of the research problem addressed in this thesis. First, we discuss the motivations behind our work. In the following section, we define the research problem and the challenges it presents. Then, we discuss the overview of our approach to addressing this problem. Finally, we go over the document’s organizational structure.

1.1 Motivation

At present, Intensive Care Units (ICU) in hospitals worldwide are collecting Electronic Health Record (EHR) data, which contain lab results, vital signs, medical codes (diagnosis, procedure, and medication codes), and other ICU-related information. This data could help healthcare professionals make proper recommendations and diagnoses to patients but is generally underutilized. Nevertheless, 12 million United States (US) citizens are affected by diagnostic errors every year. Furthermore, it is estimated that as many as 40,500 US ICU patients die each year due to misdiagnoses. The most common causes of diagnostic errors are premature case closure, clinical
assessment errors, incorrect interpretation of diagnostic tests, and rejection of physician-ordered diagnostic tests due to various circumstances [3, 4, 5, 6]. Moreover, delays in diagnosis can delay care, which can affect the patient’s health, finances, and mortality [7, 8].

In parallel, modern Artificial Intelligence (AI) has achieved significant performance in modeling image, audio, and text data with deep learning methodologies [9]. Thus, deep learning could be a promising solution to reduce diagnostic error by taking advantage of the vast amount of available EHR data to leverage forecasting of diagnosis, procedures, and medication codes of the subsequent visits (also known as forecasting patient trajectory). This will help physicians in the decision-making process when offering care to patients and can mitigate many diagnostic errors.

### 1.2 Challenges

Although deep learning models have been widely adopted for modeling EHR data applications and have shown significant results in mortality prediction, length of stay, and phenotype classification [10], less attention has been given to forecasting the patient trajectory. Despite advances in deep learning algorithms, there remain several challenges in modeling patient trajectories. The primary challenge is to make accurate predictive inferences from complex multi-modal data due to variance, insufficient diversity in sample data, and missing values. The second challenge is that the medical codes present in the dataset contribute to high granularity, which increases the hypothesis space. Additionally, learning the correlation between the diagnosis, procedure, and medication codes of multiple visits is complex and leads to longer sequences of input data.

Prior work in patient trajectory forecasting addresses the challenges discussed
above [11, 12, 13, 14, 15, 16, 17]. However, most of the research focuses on the Recurrent Neural Network (RNN) model [11, 12, 13, 14, 18], which has limitations in learning the long-term sequences [19]. Moreover, while existing approaches [11, 12, 13, 14] consolidate the medical codes and convert them into one hot encoding; they do not take the temporal characteristics of diagnosis, procedure, and medication codes into consideration while modeling the patient trajectory.

1.3 Research Problem

In this thesis, we address the following research problem:

"Given the past medical codes of a patient’s visits, how should the patient’s trajectory be forecasted?"

This research problem is further divided into three research questions:

1. How should the data be modelled such that the deep learning model is able to forecast the patient’s medical codes for the subsequent visits?

2. How should a deep learning architecture be built that will be well suited for the research problem?

3. How should the impact of input features on the model’s predictions be computed?

1.4 Approach Overview and Contributions

To address the challenges and limitations of previous works, we propose the use of Transformer-based Generative Adversarial Networks (GAN) to forecast the patient trajectory, as illustrated in Figure 1.1. GANs has shown effective performance in learning the complex multi-modal distribution of image and video-based data [20, 21]. Moreover, GANs has been used in generating synthetic EHR data and
imputing missing values \cite{22, 23}. The Transformer is a state-of-the-art methodology in Natural Language Processing (NLP) and has shown significant performance in machine translation compared to RNNs \cite{24}.

![Figure 1.1: Graphical illustration of the Clinical-GAN architecture forecasting the patient trajectory, given a sequence of the patient's past medical history.](image)

To address our first research question, we take the sequential order of diagnosis, procedure, and medication codes and group them together with three additional tokens to distinguish between visits of a patient (explained in Chapter 3). To answer our second research question, our proposed GAN uses a Transformer Encoder-Decoder as the Generator network and a Transformer Encoder as the Discriminator network. Additionally, we use adversarial loss along with negative log-likelihood loss, which enables the model to learn the distribution of the complex multi-modal data and thus helps in improving the accuracy of patient trajectory forecasting while handling long sequences. Finally, we utilize multi-head attention of the Transformer-based Generator to visualize the contribution of each input in predicting the forecasted codes. To the best of our knowledge, this work presents the first use of Transformer-based GANs in forecasting the patient trajectory, given a sequence of patient’s past diagnosis,
procedure, and medication codes.

Our contribution in this thesis is summarized as follows:

- We propose a novel architecture called Clinical-GAN, which combines Transformer and GAN methodology to forecast the medical codes for the patient’s subsequent visits.

- We introduce a methodology for representing the input features to feed the model, which aids the proposed method in learning the inherent relationship between medical codes and forecasting the patient trajectory. Additionally, we show the local interpretability of the model outcome using the Transformer-based Generator’s multi-head attention.

We evaluated our model using the real-world, recently published public dataset \textsc{MIMIC-IV} v1.0 \cite{johnson2019mimic} and considered \textit{International Classification of Disease} (ICD) version 9 and 10 codes. Our empirical experiments demonstrated that our proposed method achieved higher accuracy than baseline methods in patient trajectory forecasting and previous works in predicting the next visit diagnosis codes (sequential disease prediction). We have submitted our work as a research article in the \textit{Artificial Intelligence in Medicine} journal \cite{ours}.  

1.5 Thesis Organization

The remaining sections are organized as follows. Chapter 2 covers the background knowledge relevant to understand the rest of the thesis and briefly discusses the prior works on patient trajectory forecasting. Chapter 3 presents the input/output notations and describes the proposed methodology in detail. Chapter 4 discusses the dataset, various experiments conducted using the dataset, and quantitative and qualitative analyses of results. Finally, Chapter 5 discusses conclusions and future work.
1.6 Summary

In this chapter, we highlighted the severe impact of misdiagnosis errors by physicians, which can even result in a patient’s death. We proposed a deep learning methodology called Clinical-GAN to forecast the patient trajectory that will help physicians reduce misdiagnosis errors. We introduced our research problem and discussed the challenges to solving it. We also briefly described the overview of our approach and explicitly stated our main contributions to this work.
Chapter 2

Background and Related Works

This chapter discusses the background knowledge required to understand the rest of the thesis. Then, we will briefly describe the prior works on forecasting patient trajectory, GANs, and Transformer.

2.1 Background

2.1.1 Deep Neural Networks

Recently, deep learning approaches have become more popular due to a surge in the availability of data. In addition, computation power such as the Graphical Processing Units (GPUs) has advanced and become more accessible. Moreover, deep learning approaches can outperform existing traditional methods.

Neural networks consist mainly of three components: input, hidden, and output layers. Each layer consists of multiple nodes (neurons). Each node computes linear regression followed by an activation function to capture the non-linearity of the previous layer’s output. All the nodes have a weight parameter associated with them, which must be calculated based on the backpropagation error. The input layer consists of features that are selected from the dataset. The hidden layer learns the
new representation from the previous layer’s nodes. Finally, the output layer produces the desired output (see Figure 2.1). In this way, neural networks approximate the mapping function from input to output. When neural networks have more than three layers, the architecture is referred to as Deep Neural Networks (DNN) [26]. Below we will discuss the various deep learning approaches that are suitable for sequence to sequence mapping.

Figure 2.1: Graphical illustration of Deep Neural Networks (created using [27]), which has an input layer (eight nodes), an output layer (two nodes), and two hidden layers (six and four nodes).
2.1.2 Recurrent Neural Networks

Even though DNN can learn the meaningful mapping between input and output, there are still a few limitations. Firstly, DNN has fixed-sized input and output vectors. Secondly, DNN architecture does not support sequential learning. More specifically, they do not consider previous events in a sequence to predict the later ones.

RNN addresses the above mentioned shortcoming. RNNs do not consume all input data simultaneously. Rather than that, they consider one input data at a time. The RNN performs a series of computations at each step before providing an output. The result, referred to as the hidden state, is then added to the following input in the sequence to generate another output. This process continues till the last data in the input sequence is reached (see Figure 2.2). This method allows the model to learn the sequential nature of the input data. Moreover, the architectural design of RNN provides flexibility in handling variable input and output sequences depending on the problem statement.

A typical RNN has a drawback. When backpropagating through the computation graph, the gradients vanish (gradient approaches 0) or explode (gradient approaches extremely high values) due to the large number of layers. As a result, RNNs have difficulty learning long-term dependencies [28]. This shortcoming is addressed by Long Short-Term Memory (LSTM) [29] and Gated Recurrent Unit (GRU) [30] architectures.
Long Short-Term Memory

LSTM addresses the vanishing gradient problem by using a gating mechanism. There are three gates called the Forget gate, the Input gate, and the Output gate. These gates regulate the information flow in a memory cell or cell state. The memory cell is responsible for preserving the long-term memory or dependencies (past events in the sequence).

The Forget gate decides which portion of the previous hidden state to be kept in the memory cell based on the current input and previous hidden state. The Input gate is responsible for proposing a new hidden state based on the previous hidden state and current input. It also determines how much of the new hidden state will be
added to the memory cell. Finally, the output gate will produce a new hidden state based on an updated memory cell (see Figure 2.3).

![LSTM diagram](image)

(a) Forgot gate  
(b) Input gate  
(c) Updating cell state  
(d) Output gate

Figure 2.3: An LSTM diagram. a) The forget gate \( f_t \) examines the previous hidden state \( h_{t-1} \) and the current input \( x_t \) to determine which part of \( h_{t-1} \) information should be kept in the previous cell state \( C_{t-1} \). b) The input gate \( i_t \) determines which portion of new hidden state \( \tilde{C}_t \) should be used in the \( C_t \) based on \( h_{t-1} \) and \( x_t \). c) The operation of \( f_t \) and \( i_t \cdot \tilde{C}_t \) updates the \( C_t \), resulting in a new cell state \( C_t \). d) The output gate \( o_t \) will determine which information from \( C_t \) should be retained in order to generate a new hidden state \( h_t \). Image courtesy of Christopher Olah [31].

**Gated Recurrent Unit**

The **GRU** is another **RNN**-based sequential architecture. **GRU** also uses the gating mechanism to avoid the vanishing gradient problem. **GRU** only uses two gates, called the Update and Reset gates. The **LSTM** architecture is quite effective. However, it is
computationally expensive and takes a long time to train compared to GRU [32].

The Reset gate role is to propose a new hidden state by considering the previous hidden state and the current input. On the other hand, the Update gate is responsible for determining how much of the previous hidden state should be preserved based on the current input. Additionally, how much of the newly computed hidden state should be included in the final hidden state (see Figure 2.4).

In comparison to LSTM, GRU combines the responsibilities of the Forget and Input gates into the Update gate. Additionally, GRU does not have a dedicated memory cell for storing the long-term dependency; instead, the memory cell and hidden state are combined. In our work, we will use the GRU-based Encoder-Decoder framework as one of the baseline methods, which will be discussed in the next section.

Figure 2.4: A GRU diagram. $\tilde{h}_t$ is the updated hidden state computed as part of the Reset gate. $z_t$ is a part of the Update gate operation, which decides what new information is to be stored in the hidden state. $r_t$ decides how much of the previous information should be retained with respect to the current input $x_t$. Image courtesy of Christopher Olah [31].
2.1.3 RNN Encoder-Decoder

In this section, we briefly describe the framework introduced by Bahdanau et al. \cite{33} called the RNN Encoder-Decoder. There are three main components of this architecture called Encoder, Decoder, and Attention.

An Encoder takes an input sequence and generates a continuous representation by passing the input through a bi-directional RNN \cite{34}. RNN representation summarizes only preceding words, but bi-directional RNN summarizes both preceding and following words. Bi-directional RNN consists of one forward RNN and one backward RNN. The forward RNN reads the input sequence from left to right, while the backward RNN reads the input from right to left. Finally, the final hidden states of each word are computed by concatenating both forward and backward RNN’s hidden states.

The Decoder takes the previous hidden state computed using an RNN and the Encoder’s hidden states. Then, it computes an attention vector. The attention vector will be equal to the length of the input sequence where each element is between 0 and 1—intuitively, representing the probability of each word in the input sentence to predict the next word. More specifically, the attention mechanism signifies how much importance needs to be given to the hidden states of the Encoder with respect to the previous hidden state of the Decoder in computing the next hidden state and the next word in the sequence. We then pass the previous predicted word, attention vector, and new hidden state through a linear layer to predict the next word.

LSTM is computationally expensive compared to GRU and has comparable performance in terms of accuracy \cite{32}. As a result, we considered using the GRU instead of the LSTM network in the Encoder-Decoder framework.
2.1.4 Word Embeddings

<table>
<thead>
<tr>
<th>word1</th>
<th>Deep</th>
<th>word3</th>
<th>learning</th>
<th>word5</th>
<th>is</th>
<th>the</th>
<th>future</th>
<th>word9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 2.5: One-hot encoding of the sentence "Deep learning is the future". Assuming there are nine unique words in a vocabulary, we create a zero vector for each word in the sentence of length nine and add one at the index of the corresponding word.

In our work, we will treat the medical codes assigned to the patients during their admissions as a word token (explained in Chapter 3), similar to the language modeling in [NLP]. As a result, we will use word embeddings rather than one-hot encoding to represent the medical codes. Therefore, we will briefly describe the word embeddings in this section.

The machine learning models take inputs in the form of numbers. In [NLP], typically, the features are words. Hence, we need to convert our text into numbers. One of the common approaches is one-hot encoding. In one-hot encoding, each word in a text is represented by a zero vector of length equal to the vocabulary size, with a single one at the index of the corresponding word (as illustrated in Figure 2.5).

There are some significant drawbacks to this approach. Firstly, if the vocabulary is huge, the learnable parameters of the model will be extremely large. For instance, suppose a vocabulary has 10,000 words, and the input document contains 200 unique words. Then our input would be 200 * 10000 in size, corresponding to two million attributes. Secondly, most of the hidden layer’s nodes will be zero due to a high degree of sparseness in the input data. Finally, one-hot encoding only has information about
a specific word at each node (one and rest all zeros in a vector for a specific word),
which does not capture the similarity between the words. Word embeddings address
these limitations.

Word embedding is an important breakthrough in NLP. It is a distributed repre-
sentation of words in a text, where similar words have similar representations. More
specifically, an embedding is a dense vector of floating-point values for a specific word
in a corpus. The vector is not manually encoded. Instead, they are learned using
linear functions in a neural network. Hence, embeddings are the first hidden layer
with only linear functions in a neural network (see Figure 2.6).

Figure 2.6: A graphical illustration of word embedding on the sentence "Deep learning
is the future". Each unique word is converted to integers, which will be passed
through a hidden layer of size four comprised of only a linear function. This process
will generate a four-dimensional embedding vector that represents each word in the
sentence. Intuitively, an embedding vector is a lookup table where each row represents
a corresponding word.

The size of the embedding vector is the number of nodes in the hidden layer
specified as part of the model. This approach allows the model to learn similar vector
representations of similar words. Additionally, it reduces the high dimensionality of words by specifying fewer nodes in the hidden layer. Further work on embedding such as GloVe \cite{Pennington2014}, Word2Vec \cite{Mikolov2013} are pre-training the embedding on some tasks. Then, they use the learned embedding layers on other learning tasks. In our work, we focus only on learning the embedding from the task at hand.

2.1.5 Cosine Similarity

The similarity between words in a text is measured by cosine similarity. It measures the cosine angle between two vectors in a multi-dimensional space. Cosine similarity is computed as shown in Eq. \ref{eq:2.1}.

In the word embedding technique, each word in a text is represented by a fixed-size vector, which is learned by the model. Then, we apply cosine similarity between the learned word vectors to calculate the similarity. By doing so, we could validate whether the embedding vector has a similar vector representation for semantically similar words.

\[
\text{similarity} = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{2.1}
\]

where \(A_i\) and \(B_i\) are elements of vector \(A\) and \(B\) respectively. The value computed using Eq. \ref{eq:2.1} ranges between -1 and 1. The cosine similarity of -1 between two vectors indicates that they are opposite to each other. Two similar vectors will yield a cosine similarity of 1, otherwise 0, indicating that the vectors are independent to each other even though some degree of similarity could exist (see Figure \ref{fig:2.7}).
Figure 2.7: a) Two vectors $a$ and $b$ that have a smaller angle between them will have a high degree of similarity. b) Two vectors with a 90-degree angle will have a similarity of zero, indicating that both vectors are independent. c) Opposing vectors have a similarity value of -1, indicating that the vectors are unrelated. Image courtesy of [37].

2.1.6 t-SNE

t-Distributed Stochastic Neighbour Embedding (t-SNE) [38] is a way to visualize high-dimensional data on a two-dimensional or three-dimensional map. It converts similar data points to probabilities. Similar data points in the high-dimensional space are represented by Gaussian joint probabilities. The similar data points in the embedded space are represented by Student’s t-distributions. It then tries to minimize the difference between the calculated distributions using the gradient descent method.

With this approach, t-SNE transforms high-dimensional data into a lower-dimensional space. It finds patterns in the data by finding clusters of data points with various attributes similar to one another and groups them together in the two-dimensional map (see Figure 2.8 and 2.9).
Figure 2.8: A t-SNE plot of MNIST handwritten digit dataset \[39\]. t-SNE reveals different clusters representing each digit. Each cluster is encoded with a unique color. Image courtesy of Christopher Olah \[40\].

Figure 2.9: a) A trefoil knot visualized in three dimensional map. b) A t-SNE representation of the trefoil knot in a two-dimensional plane which reflects t-SNE’s performance in capturing local and global structures. Image courtesy of \[41\].
In our work, we use t-SNE to project the learned embeddings onto a two-dimensional map. Then, using qualitative validation, we assess whether or not the relevant associated codes are spatially closer to one another in the two-dimensional space. This indicates the model’s capability in learning the relationship between medical codes.

2.1.7 Transformer

The Transformer model was originally built for NLP tasks, where it outperformed state-of-the-art methods. In our work, we used a similar architecture as the original Transformer architecture [24]. We will briefly introduce the Transformer components and refer readers to [24] for a detailed description. Transformers consist of two modules called Encoder and Decoder (see Figure 2.10).

**Encoder**

The Encoder consists of several identical layers. Each layer is comprised of two sublayers: the multi-head self-attention mechanism and the feedforward connected network. We apply dropout, residual connection, and then layer normalization on both outputs of the two sublayers. The output of each encoding layer is then considered as input for the next Encoder layer.

Given an input vector $X$, the goal of the Encoder is to generate continuous representation $Z$ by passing the input vectors via the self-attention layer and the feedforward fully-connected network.
Multi-Head Self Attention

The self-attention layer’s purpose is to learn the dependency or relation between the tokens in the same sequence. This layer takes three input vectors called queries \(Q\), keys \(K\) and values \(V\) for each token in the sequence to compute the output. \(Q\) and \(K\) are of dimension \(d_k\) and \(V\) of dimension \(d_v\). The output is computed by using scaled dot-product attention as described in [24] and is represented as below:

\[
\text{Attention}(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V
\]  

(2.2)
where \( Q, K, V \) are a packed set of queries, keys, and values. \( T \) is the length of keys present in a sequence.

Multi-headed attention is when the self-attention is computed multiple times in parallel, and then each computed self-attention or head is concatenated together, followed by a fully connected neural network. This process makes the Encoder learn a much better representation than using a single head. Therefore,

\[
\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O
\]  

(2.3)

\[
\text{where head}_h = \text{Attention}(QW^Q_h, KW^K_h, VW^V_h)
\]  

(2.4)

where \( W^Q_h, W^K_h, W^V_h, W^O_h \) are parameters learned by the model.

**Input Embedding and Positional Encoding**

Given an input sequence \( X \), we use the word embedding technique to represent each token in the sequence as a fixed size vector. Then, we use a positional encoding that learns the order of sequence using learned embedding or static embedding. In our study, we have used learned embedding. After that, we add the input embedding and the positional embedding element-wise to get a final vector that contains the token and its sequence information. This final vector is considered the input for both initial Encoder and Decoder layers.

**Decoder**

The Decoder also consists of several identical layers. In addition to the two sublayers of an Encoder layer, the Decoder has a third sub-layer responsible for performing multi-head attention over the output of a top encoded layer, followed by
dropout, residual, and layer normalization same as the Encoder. The self-attention layer is modified by using a mask such that prediction for the next token can only depend on previous predicted/known outputs.

2.1.8 Generative Adversarial Networks

GAN consists of two models: Generator (G) and Discriminator (D), where G’s purpose is to generate sample $x'$ from the latent space $z$ as input. D takes the real data $x$ and the generated output $x'$, and classifies them as either real or fake. The objective of G is to minimize the probability of D correctly predicting real or fake, and D’s objective is to maximize the probability of correctly predicting which input is real or fake (see Figures 2.11, 2.12, and 2.13). Hence, the training objective is a min-max game with the below objective:

$$\min_G \max_D V(G, D) = \mathbb{E}_{x \sim p_{data}}[\log D(x)] + \mathbb{E}_{z \sim p(z)}[\log (1 - D(G(z)))] \quad (2.5)$$

However, GAN is unstable and often hard to train with, which may result in a mode collapse. Therefore, Wasserstein Generative Adversarial Networks (WGAN) [42] was introduced to resolve the instability and mode collapse. In our method, we prefer WGAN to the original GAN. Hence, the training objective is:

$$\min_G \max_D V(G, D) = \mathbb{E}_{x \sim p_{data}}[D(x)] - \mathbb{E}_{z \sim p(z)}[D(G(z))] \quad (2.6)$$

In Eq. 2.6, the role of D is to generate a number between $-\infty$ and $\infty$, rather than classifying the sequence as either real or fake. In WGAN, the Discriminator is called a Critic. However, we will use the term Discriminator and its notation D in place of Critic for ease and simplicity.
Figure 2.11: Overview of GAN. The Discriminator and the Generator are both neural networks. The output of the Generator is directly connected to the Discriminator’s input. The Discriminator’s classification is used to update the Generator’s weights via backpropagation. Image courtesy of [43].

Figure 2.12: A GAN’s Discriminator is only a classifier. It attempts to classify between real and generated data. During the Discriminator’s training, the Generator’s weights remain constant, and the Discriminator uses Discriminator loss to update its weights through backpropagation. Image courtesy of [44].
Figure 2.13: The Generator part of GAN takes random noise as input and generates fake data. The generated data goes through a Discriminator network to classify whether the generated data is real or fake. The Generator’s loss is then used to update the weights of the Generator network through backpropagation. Image courtesy of [45].

2.2 Related Works

Our research primarily focuses on forecasting patient trajectories. More specifically, we are trying to forecast diagnoses, medication, and procedure codes for the subsequent visits of a patient. By forecasting diagnoses, medication, and procedure codes, the physician would be able to interpret the diseases accurately. Moreover, health insurance companies can take advantage of the forecasted medical codes to offer appropriate coverage to patients. However, most research focuses on forecasting diagnosis codes for the next visit of a patient and disease progression over time. Hence, the prior work is mainly categorized into sequential disease prediction and trajectory disease forecasting. Our study overlaps between the domains of sequence models, GAN, trajectory disease forecasting, and sequential disease prediction. As a result, we will briefly highlight the most relevant works in this domain, as well as their shortcomings.
2.2.1 Trajectory Disease Forecasting

Some research has been conducted in analyzing EHR datasets and patient disease trajectory prediction. For example, Danish Disease Trajectory Browse (DTB) utilizes existing medical data to create disease trajectories by calculating progression through a manual process of permutation and combination. However, this model faces the following drawbacks: the diagnosis pair is identified only through statistical significance and builds a linear trajectory of disease upon the identified diagnosis pair such that it does not forecast medical codes for subsequent visits. Hence, the results are not personalized but rather generalized based on statistics, and their inference time is high. The advantage of forecasting subsequent visits medical codes instead of the next visit is that it gives a comprehensive analysis of an individual patient’s status over time. Similar to DTB, other studies are conducted on modeling specific diseases using statistical methods such as Alzheimer’s, glaucoma, diabetes mellitus, and chronic kidney disease. Additionally, Disease-Atlas uses joint modeling, where the model simultaneously learns the temporal relation between variables and the shared representation between disease trajectories to predict the mortality of a patient diagnosed with cystic fibrosis. The authors used RNN based methods for their task and argued that their method was computationally effective; however, the authors did not provide the code, so it was not reproducible.

2.2.2 Sequential Disease Prediction

Numerous approaches have been proposed to predict future diagnoses. Typically, in most prior works, methods are based on variants of RNN. For instance, the Doctor AI model predicts a patient’s medical codes and estimated time until the next visit. Doctor-AI takes hospital visits by the patients’ as their temporal attribute. At
each timestep, medical codes assigned to a patient during a visit are considered as the features. They represent the features as a multi-hot label vector. Then, an embedding layer is used to convert the inputs to a lower-dimensional space by using the skip-gram pretraining technique. Then, GRU processes the lower-dimensional space to predict the next visit diagnosis codes along with the time duration of the next visit. However, their approach is limited to a fixed window width. Hence, it cannot be trained for subsequent visits.

Rodrigues-Jr et al. [12] extended the Doctor AI method [11] as LIG-Doctor. Their input representation is similar to Doctor-AI. They convert the medical codes of each visit into the multi-hot vector and use the visit as their temporal attribute. They incorporate a bidirectional recurrent neural network with an architecture of Minimal Gated Recurrent Unit (MGRU) [52]. Finally, they use a feed-forward layer to process the output from the bidirectional-MGRU network. Their approach aims to address the high granularity of ICD-9 codes. Similar to Doctor-AI, their approach is also limited to a fixed window width. Moreover, even though they handle the high granularity issue, they do not consider procedures, medications, and ICD-10 based coding mechanism.

In the Reverse Time Attention Model (RETAI N) [13] architecture, two RNNs are trained in reverse time order to learn the weights and importance of previous visits on input data. The role of the attention mechanism is to determine the feature significance for easier interpretability. DeepCare [53] uses LSTM [29] to accomplish three tasks: predicting the next visit diagnosis code, intervention recommendation, and future risk prediction. They conduct their experiment using only two groups of patients who are diagnosed with diabetes and mental health. Unfortunately, the RETAIN repository does not provide the codebase for sequential disease prediction. Similarly, DeepCare repository code is not executable as the authors provided incomplete code.
Manashty et al. [54] proposed a Life Model (LM) framework that efficiently represents temporal data in concise sequences to train RNN-based models efficiently. Additionally, LM uses Mean Tolerance Error (MTE) as its loss function and metric for sequence modeling. The authors used the MIMIC-III dataset to forecast the diagnosis and procedure codes of the patient’s next visit using the LSTM model. They converted the medical codes into one-hot encoding and modeled the input and output using LM. Miotto et al. [55] proposed an unsupervised deep learning approach called deep patient. First, they used Stack Denoising Autoencoders (SDA) to extract meaningful feature representations of patients from the EHR data. Then, they used learned patient representation from SDA to predict the occurrence of certain diseases in the future. They demonstrated that their method outperforms other feature extraction methods such as Principal Component Analysis (PCA), K-means, Gaussian Mixture Model (GMM), and Independent Component Analysis (ICA). However, the deep patient does not take temporal characteristics of the features, and it only predicts 78 diseases. Additionally, it performs poorly on some of the diseases in terms of accuracy.

The sequential disease prediction problem has also been tackled by other approaches not based on deep learning, some of which are based on Markov chain models and Bayesian networks [14, 16]. Wang et al. [14] utilized both Markovian and Bayesian concepts to model disease progression. The main disadvantages of these approaches are their high complexity and huge state-space, which are not practical for real-world applications. Incidentally, some approaches get benefits from other statistical methods such as Hawkes Processes [15]. Hawkes Processes generate models linearly while the non-linear version of this method incurs intense complexity.

In comparison with the above-mentioned methods, we are using the Transformer mechanism, which is better than traditional RNN-based methods in learning the order of sequence and the correlation between the features (medical codes) [24]. Our method
can forecast the patient’s medical codes for subsequent visits and is not limited to only forecasting medical codes for the next visit. Furthermore, we are using the MIMIC-IV dataset [10], which has more data than MIMIC-III [56]. Our approach represents the input data akin to language modeling techniques and uses special tokens to distinguish between visits and medical codes. Moreover, we take the sequential order of diagnosis, procedure, and medication codes into account instead of consolidating the medical codes and converting them into one-hot encoding.

2.2.3 GAN and Transformer

GAN has been used successfully for image synthesis [57, 58, 59, 60]. Moreover, GAN is also being used along with RNN-based models to generate synthetic EHR data [23, 61, 62] and imputing time series and non-time-series data [22, 63, 64]. Another application of GAN is the estimation of future trajectories based on previous observations [65, 66, 67, 68], where an RNN-based Encoder-Decoder Generator network is used in predicting socially acceptable human motion behaviour in various crowded and real-life scenes. Moreover, a study [69] used RNN-based GAN for forecasting applications using multivariate and univariate inputs. Another successful instance of applying GAN is in the generation of high-quality language descriptions [70]; the reports of RankGAN [70] showed that when generating human-like language, their method is superior to traditional methods, including maximum likelihood estimation.

Transformers [24] have been utilized in various applications. For instance, the performance of Transformer-based sequence-to-sequence architectures has been explored for summarizing medical conversations in [71]. Recently, some works have explored the usage of Transformers in GAN structure [72, 73]. The approach by Jiang et al. [73], called TransGAN, is a GAN that is based on Transformer architecture for generating images while being completely free of convolutions. The results of their evaluation
show that TransGAN achieves highly competitive performance in comparison with convolution-based GAN. In comparison with these approaches, we forecasted patient trajectory using Transformer-based GAN to learn the multi-modal distribution of the data and correlation between medical codes while handling long sequences.

2.3 Summary

This chapter briefly discussed several deep learning algorithms, including DNN, RNN, LSTM, GRU, RNN Encoder-Decoder framework with attention mechanism, Transformer, and GAN, along with a brief discussion of their limitations and benefits. We also covered some NLP concepts such as word embeddings, cosine similarity, and t-SNE. In addition, this chapter reviews the related work in forecasting patient trajectory, disease progression, and applications developed using GAN and Transformer. The next chapter introduces the proposed Clinical-GAN.
Chapter 3

Clinical-GAN

3.1 Introduction

In section 1.2, we described the challenges in modeling patient trajectories, and in section 1.3, we introduced the research questions. In this chapter, the architecture of the proposed model and the data representation methodology are provided. The idea behind the Clinical-GAN was developed when looking for a way to forecast possible future diagnoses such that a physician could look at the forecasted trajectory and verify their judgments. The goal of this research is to forecast the medical codes of a patient’s subsequent visits, given their past diagnosis, procedure, and medication history. For instance, if a patient has asthma, they should consult a physician. The physician should utilize a system that forecasts the potential diagnosis and associated procedure and medication codes for their subsequent visits. As a result, the physicians get a comprehensive view of an individual patient’s status over a period of time. Therefore, they may precisely understand the disease trajectory and treat the patients accordingly. However, we recommend the physician take advantage of the proposed approach after arriving at an initial conclusion about the patient’s status. Because solely relying on an AI-based system in a sensitive environment could lead to serious
consequences. Finally, unlike a black-box model, a physician needs a model that can explain its output, as this enables a physician to comprehend the cause of the projected medical codes. Furthermore, the physician can make a considerably more accurate diagnosis and avoid misdiagnosis errors.

However, many unique medical codes need to be predicted for each subsequent visit, making the data a complex multi-modal dataset. Also, EHR data has missing values and insufficient samples for a few codes in the data. As a result, the accuracy of the system could diminish. Moreover, prior works structured their input/output data representation to predict only the diagnoses or procedure codes of the next visit, as discussed in section 2.2. Hence, there is a void in research for forecasting the medical codes for subsequent visits accurately.

As our objective is to forecast the medical codes, we need to work with the sequence models and structure our data sequentially so that our model can be trained on it. Hence, we propose a novel architecture called Clinical-GAN, a Transformer-based GAN model that tackles the problem of misdiagnosis error by forecasting patient trajectory along with the local interpretability of the model’s outcome for each instance. Our method takes the Transformer Encoder-Decoder framework as the Generator network and the Transformer Encoder as the Discriminator network. Furthermore, both the networks are trained adversarially to forecast the patient trajectory.

Most previous studies [11, 13, 53, 55] have used private datasets, which means that their findings are not exactly replicable because their data and codebase are not publicly available. Some of the studies [12, 54] conducted their experiments on MIMIC-III, which has around 50,000 patients’ data. Moreover, none of the previous work utilized the sequential behavior of medical codes. Instead, they used one-hot encoding to represent the medical codes and used the visit as the temporal attribute. Due to this, they were not effective in learning the correlation between the medical
Therefore, we introduce a novel data representation method that helps in forecasting the medical codes for subsequent visits. We used the real-world, publicly available dataset, MIMIC-IV v1.0 (discussed in detail in Chapter 4) which contains more than 196,000 patients’ data. EHR data used in our method consists of sequential based data, where each patient has a sequence of visits. In a specific visit, multiple diagnosis ($D_x$), procedure ($P_x$) and medication ($R_x$) codes are assigned in sequential order. The $D_x$ codes are assigned a ranking or priority based on their importance. To illustrate, a patient diagnosed with sepsis must have assigned the first billed $D_x$ condition as an infectious agent and the second billed $D_x$ condition as sepsis during their hospital stay. Moreover, $P_x$ and $R_x$ codes are listed in order of occurrence during a patient’s hospital stay. This approach generates a sequence of $D_x$, $P_x$, and $R_x$ codes that are sequential in nature. We utilized language modeling approaches to represent the input and output data and used special tokens to differentiate between visits and medical codes. Clinical-GAN is designed to capture this sequential relation in the data and forecast the desired medical codes.

In the following sections, first, we introduce the EHR data and notation, and their input/output representations in our problem setting, followed by the details of Clinical-GAN and its interpretation.

### 3.2 EHR Data and Notation

In this section, we will address our first research question as stated below:

"How should the data be modelled such that the deep learning model is able to forecast the patient’s medical codes for the subsequent visits?"

Each patient $i$ has a sequence of visits $V_1, V_2, \ldots, V_T$ in which each visit contains
a varying number of medical codes $M$ which consist of all $D_x$, $P_x$, and $R_x$ codes. Furthermore, the total number of medical codes $B$ assigned at each visit is grouped together without changing the sequential order of each $D_x$, $P_x$, and $R_x$ code, resulting in a sequence length $S$. Therefore, we represent our input data $X_i$ for each patient as follows:

$$X_i = \{(V_t)^i \in \mathbb{R}^{1 \times S} \mid t = 1, \ldots, t_{obs}\} \text{ for } i \in \{1, \ldots, N\}$$  

(3.7)

where $N$ is the total number of patients and representing each visit $V_t$ as below:

$$V_t = \{M_j \in \mathbb{R}^{1 \times B} \mid j = 1, \ldots, B\} \text{ for } t \in \{1, \ldots, T\}$$

(3.8)

where $T$ is the total number of visits by an $i^{th}$ patient.

From Eq. 3.7 and Eq. 3.8 given the input sequence $X_i$ of medical codes for the observed visits from $t_1$ to $t_{obs}$ timesteps, the ground truth sequence $Y_i$ for each patient from $t_{obs+1}$ to $t_{pred}$ timestep is formulated as follows:

$$Y_i = \{(V_t)^i \in \mathbb{R}^{1 \times S} \mid t = t_{obs+1}, \ldots, t_{pred}\} \text{ for } i \in \{1, \ldots, N\}$$

(3.9)

### 3.2.1 Input/Output Representation

In this section, we will describe how our input $X$ and output $Y$ are formatted to feed into the model and further simplify our notation.
Each unique medical code is mapped to a unique numerical value; then, all visits are grouped together by using three additional tokens. A start token ("start of history" <SOH>) is added at the beginning of the first visit of each patient, a separator token ("end of visit" <EOV>) is added at the end of each visit to distinguish between visits, and an end token ("end of history" <EOH>) is added at the end of the last visit of each patient (see Figure 3.1). These additional tokens are mapped to unique numerical values as well. Hence, after formatting, our input/output representation is as below:

\[
\Delta V^i_x = \{<\text{SOH}> \sim V_1 \sim <\text{EOV}> \sim V_2 \sim <\text{EOV}> \sim \ldots V_{t_{\text{obs}}} \sim <\text{EOH}>\}
\]  
(3.10)

\[
\Delta V^i_y = \{<\text{SOH}> \sim V_{t_{\text{obs}}+1} \sim <\text{EOV}> \sim V_{t_{\text{obs}}+2} \sim <\text{EOV}> \sim \ldots V_{t_{\text{pred}}} \sim <\text{EOH}>\}
\]  
(3.11)

\(\Delta V^i_x\) and \(\Delta V^i_y\) are records of an input and output data of a given \(i^{th}\) patient. Hence, from Eq. 3.7 and Eq. 3.9 we obtain the input \(X\) and output \(Y\) for \(N\) patients as
follows:

\[ X = \{(\Delta V^i_x) \in \mathbb{R}^{N \times S}\} \text{ for } \forall i \in \{1, \ldots, N\} \]  

\[ Y = \{(\Delta V^i_y) \in \mathbb{R}^{N \times S}\} \text{ for } \forall i \in \{1, \ldots, N\} \]  

The length of the sequences \( S \) will vary depending on the patient, therefore we pad short sequences with zeros at the end.

The idea behind the data representation is motivated by Zaremba et al. \[74\]. Zaremba used LSTM to perform the addition of two nine-digit numbers. First, they used character-level representation for every digit and individual operator. Then, they trained them using LSTM to perform the addition task. As a result, the model effectively learned the meaning of the "+" operator and achieved 99% accuracy. Similar to the "+" operator, in our work, we use three special tokens: <SOH>, <EOV>, and <EOH> to distinguish between different visits of a patient. Thus, our model will learn the meaning of these special tokens and forecast the patient’s medical codes for subsequent visits.

3.3 Clinical-GAN

In this section, we will describe the answer to our second research question (research problem described previously in section 1.3) as stated below:

"How should a deep learning architecture be built that will be well suited for the research problem?"

We are representing medical codes using the word-level representation of language models. In other words, we are considering the medical code as a time-ordered sequence of discrete tokens. Language models are then trained on the time-ordered sequence to accomplish certain tasks.

Data representations of language models are confined to an autoregressive structure
such that, given a sequence of tokens \( x = [x_1, \ldots, x_T] \), we compute the unknown distribution \( p^*(x) \) and learn the parameters of the model \( \theta \) as below:

\[
p_\theta(x) = \prod_{i=1}^{T} p_\theta(x_t | x_1, \ldots, x_{t-1}) \tag{3.14}
\]

Typically, the autoregressive models are trained using Maximum Likelihood Estimation (MLE) objective:

\[
L_{mle} = -\mathbb{E}_{x^r \sim p_r} [\log p_\theta(x^r)] \tag{3.15}
\]

During the training stage, the autoregressive models follow a teacher forcing strategy [75], where the ground truth samples are fed into the model to predict the next token. However, during the inference stage, the model needs to predict the next token given the previously generated samples.

The teacher forcing algorithm leads to an issue called exposure bias [76]. This issue occurs due to the intrinsic difference between the training and inference stages of the autoregressive models. The effect of the exposure bias results in poor prediction, repetitive outputs, and incoherent sequence generation by the model. This effect becomes more serious in the case of long-term sequences. Bengio et al. [77] proposed a methodology called scheduled sampling to resolve the exposure bias problem by using teacher forcing and free generation as a prefix, based on the random variable. They argued that their approach reduces the distribution difference between the training and inference stages. However, [76] proved that scheduled sampling is ineffective in resolving the exposure bias issue.

Furthermore, [76, 78, 79, 80] addresses the exposure bias problem by incorporating the adversarial loss into the MLE objective. Additionally, they show that their approach performs better in handling long-term dependencies. Moreover, [81, 82]
showed that GAN does not have exposure bias problem in the text generation tasks. The Transformer architecture has emerged as the state-of-the-art in language modeling tasks. Hence, we use the Transformer-based GAN as our architecture.

We consider patient forecasting data to be non-stationary. Since the disease can occur due to an unexpected incident, and the observation of clinical events (medical codes) at different admissions could be correlated. Traditional machine learning algorithms typically need the data to be stationary \[83, 84\]. However, sequence models such as RNN-based methods, and Transformer can efficiently handle non-stationary multivariate time series data \[85, 86\]. They effectively capture long-term dependencies in the input sequence, and the inherent architecture predicts the next token based on the learned context of the input data. Thus, the model can learn the hypothesis space by handling the non-stationary data. In addition, Transformer has been shown to outperform RNN-based architectures significantly \[24\]. This further motivated our choice to select Transformer-based GAN as our architecture.

The proposed novel model uses a GAN framework composed of the Generator (G) and Discriminator (D) networks (see Figures 3.2 and 3.3), where G is composed of Transformer-based Encoder (Enc\(_g\))-Decoder (Dec\(_g\)) framework and D of Transformer-based Encoder (Enc\(_d\)) except the last layer of the Encoder is followed by a fully connected layer. G takes \(X\) as input and generates future visits medical codes \(\hat{Y}\) of the patients. D takes a pair consisting of real data \((X, Y)\) and generated data \((X, \hat{Y})\) and measures the distance between their probability distribution. Additionally, we update the model parameters based on the backpropagation error using adversarial loss and a negative log-likelihood loss during the training phase.
Figure 3.2: The Generator architecture of Clinical-GAN. It comprises Encoder and Decoder models.
3.3.1 Discriminator

Firstly, we optimized the D with a fixed G by drawing a sample \((X, Y)\) from a minibatch of size \(m\) along with the generated output \(\hat{Y}\) of size \(m\) using G. Then, the \(\text{Enc}_d\) network takes real data \((X, Y)\) and generated data \((X, \hat{Y})\) as inputs and learns to distinguish between them. As discussed in section 2.1.8, we considered WGAN to avoid mode collapse issue. Hence, from Eq. 2.6 we define the loss of D according to

\[ \text{loss}_D = -\mathbb{E}_{(X, Y) \sim \mathcal{D}} \log D(X, Y) + \mathbb{E}_{(X, \hat{Y}) \sim \mathcal{G}} \log (1 - D(X, \hat{Y})) \]
the objective of \textbf{WGAN}:

\[
L_{\text{Dist}} = \mathbb{E}_{X \sim p_r} [\text{Enc}_d (X, G(X))] - \mathbb{E}_{X,Y \sim p_r} [\text{Enc}_d (X,Y)]
\]  

(3.16)

\subsection*{3.3.2 Generator}

Secondly, we optimized G along with the updated D by drawing a sample \((X)\) from a minibatch of size \(m\), which is then passed through the \text{Enc}_g. That action will generate a fixed-size continuous value representation \(Z\), which will be used as an input along with the output \(\hat{Y}\) generated by \text{Dec}_g. In addition to that, as discussed earlier, G has two objectives to alleviate the exposure bias. First, maximize the likelihood of the data, and second, fool the discriminator. Hence, we incorporate the negative log-likelihood loss with the adversarial loss. Using Eq. 2.6 and Eq. 3.15, we define the objective of G as follows:

\[
L_G = L_{\text{mle}}[G_{\theta}] + \alpha L_{\text{gen}}
\]

(3.17)

where

\[
L_{\text{gen}} = \mathbb{E}_{X \sim p_r} [-\text{Enc}_d(X, G(X))]
\]

(3.18)

and \(\alpha\) is a hyperparameter.

We provide the pseudocode in Algorithm 1 and Figure 3.4 for an overview of the training procedure.
Algorithm 1: Training procedure of Clinical-GAN

\[ \theta_{\text{Enc}}, \theta_{\text{Dec}}, \theta_{\text{Dis}} \leftarrow \text{weight initialization using Xavier} \]

\[ K_g, K_d \leftarrow \text{number of iterations, 5} \]

Def \( G(\text{Enc}_g, \text{Dec}_g, X) \):

- Apply input embedding and positional encoding on \( X \)
- \( Z \leftarrow \text{Enc}_g(X) \)
- \( K, V \leftarrow Z \)
- \( \hat{Y} = \left[ \langle \text{SOH} \rangle \right] \)

for \( K_g \) iterations do
  \( \tilde{Y} = \text{Dec}_g(K, V, \hat{Y}) \)
  \( \hat{Y} \leftarrow \hat{Y} + \tilde{Y} \)
  if \( \tilde{Y} == \langle \text{EOH} \rangle \) then
    break
  end if
end for

return \( \hat{Y} \)

while not converged do
  \( X, Y \leftarrow \text{Draw random sample from minibatch size } M \)

  // Training Discriminator
  for \( K_d \) iterations do
    \( \hat{Y} \leftarrow G(\text{Enc}_g, \text{Dec}_g, X) \)
    // Update the discriminator according to gradients
    \( \theta_{\text{Dis}} \leftarrow -\nabla_{\theta_{\text{Dis}}} L_{\text{Dis}} \)
  end for

  // Training Generator
  \( \hat{Y} \leftarrow G(\text{Enc}_g, \text{Dec}_g, X) \)
  // Update the generator according to gradients
  \( \theta_{\text{Enc}} \leftarrow -\nabla_{\theta_{\text{Enc}}} L_G \)
  \( \theta_{\text{Dec}} \leftarrow -\nabla_{\theta_{\text{Dec}}} L_G \)
end while
Figure 3.4: Flow through the Clinical-GAN model during training. Our model consists of two components: Generator (G) and Discriminator (D). G takes historical medical data ($X$) and forecasts the medical codes ($\hat{Y}$). D takes real ($X, Y$) and generates ($X, \hat{Y}$) sequences and produces the probability to distinguish between real and generated ones.

The motivation behind the Clinical-GAN architecture is from GAIN [22], Time-GAN [23], and TransGAN [73]. GAIN uses a DNN-based GAN framework to impute non-time series data. Time-GAN uses an RNN-based GAN to generate time-series data. Both architectures used adversarial loss and other loss functions to accomplish their tasks and outperformed previous work. On the other hand, TransGAN used a Transformer-based GAN, a free of convolution network for generating images. As a result, TransGAN reported higher accuracy than the state-of-the-art GAN.

In contrast to the GAIN and Time-GAN architectures, in our work, we used Transformer-based GAN. Additionally, we considered the adversarial loss along with the negative log-likelihood loss during training. Thus, it enables our model to learn from complex multi-modal data efficiently, resulting in a more accurate model while handling long sequences. Below we will describe the training procedure of the Clinical-GAN in detail.
3.4 Interpretability

In this section, we will describe the answer to our third research question as stated below:

"How should the impact of input features on the model’s predictions be computed?"

We demonstrate the local interpretability of the Clinical-GAN by using the multi-head attention of the Decoder network in the Generator. We took the average of all the attention heads returned by Dec\textsubscript{g} and used the resulting matrix to check which input sequence variable is influential in contributing to each predicted output variable.

We define $A \in \mathbb{R}^{k \times d_t \times d_i}$, which represents a single self-attention matrix computed from multi-head attention in the Dec\textsubscript{g}, as mentioned in Eq. 2.3. Hence, the interpretability $I$ is defined as below:

$$I = \frac{1}{k} \sum_{i=1}^{k} A_i$$

(3.19)

where $I \in \mathbb{R}^{d_t \times d_i}$ is a matrix with $d_i$ and $d_t$ as the dimensions of the input and predicted sequence of a single patient. $k$ is the total number of heads in multi-head attention computed by the Dec\textsubscript{g}.

Our approach is a post-hoc-based local interpretation of an individual instance [87]. For every instance, we use feature summary statistics—a weighted sum to explain the influence of input features on the prediction of an individual instance. Then, we visualize the feature summary statistics using a heat map (discussed in section 4.5).

3.5 Summary

In this chapter, we described our proposed novel architecture called Clinical-GAN. Clinical-GAN uses a GAN framework where the Generator network is composed of the
Transformer Encoder-Decoder framework, and the Discriminator is composed of the Transformer Encoder network. When updating the weights of the Generator network through backpropagation error, we employ adversarial loss along with the negative log-likelihood loss to capture the multi-modality of the data.

We introduced a novel data representation methodology that can aid in forecasting the patient’s medical codes for subsequent visits. We also described the notations and discussed how medical codes are stored in a sequential manner. In addition, we briefly described the formatting of input and output features. Finally, we showed how to explain the outcome of the model by using the multi-head attention in the Generator network.
Chapter 4

Experimental Results

This section describes the dataset used, data preprocessing steps, and the various experiments conducted on the proposed method. The source code of Clinical-GAN is publicly available at [88].

4.1 Datasets

We conducted all experiments on the MIMIC-IV v1.0 dataset [10]. This dataset is publicly available and comprises de-identified clinical data on patients in the ICU at Beth Israel Deaconess Medical Center (BIDMC) between 2008 and 2019. The data consists of various information for each ICU stay, such as the patients’ de-identified demographic information, diagnosis, medication, procedure, clinical notes, vital sign measurements, lab test results, and in-hospital mortality.

MIMIC-IV contains 196,043 unique adult patients, 521,111 distinct hospital admissions, 3,861 unique laboratory measurements, 27,192 unique diagnosis codes, and 13,016 procedure codes for both ICD-9 and ICD-10 versions assigned to the patients. In addition, there are 6,303 Generic Sequence Number (GSN) and 5,911 National Drug Code (NDC) unique coding identifiers for drugs prescribed to the patients during
their hospital stay. In this work, we are using NDC instead of GSN because GSN data is not properly formatted in the dataset. The statistics of this dataset are provided in Tables 4.1 and 4.2.

Table 4.1: Statistics of the overall dataset.

<table>
<thead>
<tr>
<th>General statistics</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Adult patients</td>
<td>196,043</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Admissions</td>
<td>521,111</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unique codes</td>
<td>46,119*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>-</td>
<td>2.35 (per patient)</td>
<td>45.81 (per admissions)</td>
</tr>
</tbody>
</table>

* $D_x: 27,192, P_x: 13,016, R_x: 5,911$

Table 4.2: Average statistics for each type of medical code assigned per admission.

<table>
<thead>
<tr>
<th>Codes per visit</th>
<th>Procedure</th>
<th>Medication</th>
<th>Diagnosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>2.79</td>
<td>32.89</td>
<td>10.13</td>
</tr>
</tbody>
</table>

We extracted the diagnosis, procedure, and medication codes from each patient visit. To reduce the granularity of ICD-9 and ICD-10 version codes, we used medical grouping databases such as Clinical Classification Software (CCS) [89] and Clinical Classifications Software Refined (CCSR) [90] provided by the Healthcare Cost and Utilization Project (HCUP) [91].

CCS and CCSR are created by domain experts and represented in the tabular form. They classify the ICD codes into meaningful categories for the ease of analytical purposes. For example, tuberculosis has 426 ICD-9 and 102 ICD-10 codes. CCS consolidates all 426 ICD-9 codes into a single code. Similarly, CCSR groups all 102
ICD-10 codes into a single code. This approach considerably reduces the granularity of ICD codes at the expense of comprehensive description.

CCS classifies 15,293 unique $D_x$ and $P_x$ codes based on ICD-9 into 285 CCS codes. Similarly, CCSR classifies 95,051 $D_x$ codes based on ICD-10 into 530 CCSR codes. Additionally, it classifies 81,311 $P_x$ codes based on ICD-10 into 320 CCSR codes. This mapping mechanism yields a total of 1,135 codes from a total of 191,601 codes.

From 1970 to 2014, hospitals employed ICD-9 based $D_x$ and $P_x$ codes as a standard encoding technique. Since 2015, the World Health Organization (WHO) has phased out the ICD-9 version and replaced it with the ICD-10 version as the encoding mechanism for $D_x$ and $P_x$ codes. However, there is no mechanism for converting ICD-9 to ICD-10 codes. As MIMIC-IV v1.0 consists of data between the period 2008 and 2019, there are ICD-9 codes and ICD-10 codes. Hence, we consider both ICD-9 and ICD-10 version data.

After mapping the ICD-9 and ICD-10 codes to the CCS and CCSR codes in the MIMIC-IV v1.0 dataset, the cumulative number of $D_x$ and $P_x$ codes was reduced to 1,267 codes from 40,208 ICD codes. As a result, the overall number of unique $D_x$ codes is 758, whereas the total number of unique $P_x$ codes is 509, as shown in Stage 4 of Table 4.3.

We need to preprocess the data before feeding the data into our proposed model. There are six stages of data preprocessing in our proposed method. In the first stage, we removed the neonatal data from the dataset. We excluded patients who do not have all three diagnoses, procedures, and medication information for admission in the second stage. In the third stage, we kept only the patient records with more than one admission to the hospital. In the fourth stage, we transformed the ICD-9 and ICD-10 codes to CCS and CCSR codes. In the fifth stage, for each visit, we limited the $D_x$, $P_x$, and $R_x$ codes individually by using a maximum threshold of 80 to reduce
the memory usage. Finally, in the sixth stage, we excluded medical codes with fewer than five occurrences in the data. Table 4.3 summarizes the statistics of each data preprocessing stage.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Patients</th>
<th>Admissions</th>
<th>Diagnosis codes</th>
<th>Procedure codes</th>
<th>Medication codes (NDC codes)</th>
<th>Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>196,043</td>
<td>521,111</td>
<td>27,192</td>
<td>13,016</td>
<td>5,911</td>
<td>46,119</td>
</tr>
<tr>
<td>2</td>
<td>79,189</td>
<td>114,959</td>
<td>19,188</td>
<td>10,197</td>
<td>5,236</td>
<td>34,621</td>
</tr>
<tr>
<td>3</td>
<td>21,525</td>
<td>57,295</td>
<td>14,470</td>
<td>7,732</td>
<td>4,906</td>
<td>27,108</td>
</tr>
<tr>
<td>4</td>
<td>21,525</td>
<td>57,295</td>
<td>758</td>
<td>509</td>
<td>4,906</td>
<td>6,173</td>
</tr>
<tr>
<td>5</td>
<td>21,525</td>
<td>57,295</td>
<td>758</td>
<td>509</td>
<td>4,906</td>
<td>6,058</td>
</tr>
<tr>
<td>6</td>
<td>21,525</td>
<td>57,295</td>
<td>697</td>
<td>469</td>
<td>3,430</td>
<td>4,596</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stage</th>
<th>Admissions/patient</th>
<th>Procedure codes/admission</th>
<th>Diagnosis codes/admission</th>
<th>Medication codes/admission</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.35</td>
<td>2.79</td>
<td>10.13</td>
<td>32.89</td>
</tr>
<tr>
<td>2</td>
<td>1.45</td>
<td>3.12</td>
<td>11.18</td>
<td>43.70</td>
</tr>
<tr>
<td>3</td>
<td>2.66</td>
<td>3.11</td>
<td>11.82</td>
<td>43.17</td>
</tr>
<tr>
<td>4</td>
<td>2.66</td>
<td>3.11</td>
<td>12.43</td>
<td>43.17</td>
</tr>
<tr>
<td>5</td>
<td>2.66</td>
<td>3.05</td>
<td>12.43</td>
<td>35.13</td>
</tr>
<tr>
<td>6</td>
<td>2.66</td>
<td>3.05</td>
<td>11.39</td>
<td>32.40</td>
</tr>
</tbody>
</table>

### 4.2 Forecasting Tasks

To determine Clinical-GAN’s efficacy and compare its performance with other baseline methods in forecasting subsequent visits, we performed two experimental tasks: the **Trajectory Forecasting (TF)** and the **Sequential Disease Prediction (SDP)**. In both tasks, given a patient’s past visits medical codes, the goal is to forecast the patient’s medical codes for \( t \) visits.

Firstly, we performed the **TF** task where the model learns to forecast patient trajectories. Given the input sequence of medical codes for the observed visits from \( t_1 \) to \( t_{obs} \) for each patient, the task is to forecast the medical codes from \( t_{obs+1} \) to \( t_{pred} \).
visits. To allow the model to learn the patient trajectory, we broke down each patient’s visits based on a window width \( W_d \) which increments sequentially from 1 to \( t - 1 \) visits to form a new set of input and output samples (see Table 4.4). This methodology lets the proposed model forecast medical codes in subsequent visits relative to the number of visits in the input. We provide a few input and output samples in Appendix B.

Table 4.4: An example of input and output samples generated from a patient record consisting of four visits (patient record = \( V_1, V_2, V_3, V_4 \)) for the Trajectory Forecasting (TF) task. The resulting samples are formatted based on Eq. 3.12 and Eq. 3.13 as explained in Section 3.2.1.

<table>
<thead>
<tr>
<th>Samples</th>
<th>Window width ( (W_d) )</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample-1</td>
<td>1</td>
<td>( V_1 )</td>
<td>( V_2, V_3, V_4 )</td>
</tr>
<tr>
<td>Sample-2</td>
<td>2</td>
<td>( V_1, V_2 )</td>
<td>( V_3, V_4 )</td>
</tr>
<tr>
<td>Sample-3</td>
<td>3</td>
<td>( V_1, V_2, V_3 )</td>
<td>( V_4 )</td>
</tr>
</tbody>
</table>

Additionally, we compare the trajectory forecasting of the clinical events task in three scenarios:

- \( F_D \): Forecasting only \( D_x \) codes
- \( F_{DP} \): Forecasting \( D_x \) and \( P_x \) codes
- \( F_{DPR} \): Forecasting \( D_x \), \( P_x \), and \( R_x \) codes

Secondly, previous researchers conducted their experiments only on the SDP task where the model learns to predict the next visit diagnosis codes; therefore, we conducted this task to ensure that our model performs comparatively. Given the input sequence of medical codes for the observed visits from \( t_1 \) to \( t_{obs} \), we forecasted the diagnosis codes for each patient at the \( t_{obs+1} \) visit. We used a window width similar
to the TF task for each patient as our input, then removed medication and procedure codes from $t_{obs+1}$ visit for our proposed method (see Table 4.5).

Table 4.5: An example of input and output samples generated from a patient record consisting of four visits (patient record = $V_1, V_2, V_3, V_4$) for the Sequential Disease Prediction (SDP) task. The resulting samples are formatted based on Eq. 3.12 and Eq. 3.13, as explained in Section 3.2.1.

<table>
<thead>
<tr>
<th>Samples</th>
<th>Window width ($W_d$)</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample-1</td>
<td>1</td>
<td>$V_1$</td>
<td>$V_2$</td>
</tr>
<tr>
<td>Sample-2</td>
<td>2</td>
<td>$V_1, V_2$</td>
<td>$V_3$</td>
</tr>
<tr>
<td>Sample-3</td>
<td>3</td>
<td>$V_1, V_2, V_3$</td>
<td>$V_4$</td>
</tr>
</tbody>
</table>

After formatting the data as mentioned above, we removed the patient’s records if their input or output sequence length exceeds 500 for both the trajectory forecasting and sequential disease prediction tasks to reduce the memory consumption. Table 4.6 shows the statistics of both tasks.

Table 4.6: The total number of records for the TF and SDP tasks, after formatting the data as mentioned above and removing records with a sequence length greater than 500. Before deletion, the total number of records was 35,770.

<table>
<thead>
<tr>
<th></th>
<th>TF</th>
<th>SDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_D$</td>
<td>35204</td>
<td>35205</td>
</tr>
<tr>
<td>$F_{DP}$</td>
<td>35205</td>
<td>35205</td>
</tr>
<tr>
<td>$F_{DPR}$</td>
<td>35451</td>
<td></td>
</tr>
</tbody>
</table>

| Total Records | 35204 | 35205 | 35205 | 35451 |
4.3 Baseline Methods

First, we compared our model to other baseline models for the TF task as described below:

- **GRU Encoder-Decoder**: This baseline model is developed based on the architecture of [33] with input/output representation using Eq. 3.12 and Eq. 3.13.

- **Transformer**: The Transformer model is similar to the original Transformer architecture. However, we used a learned positional encoding to learn the position of each medical code instead of a fixed positional encoding [24].

We did not consider DNNs because of their inability to map sequence-to-sequence mappings. We also did not consider any RNN-based models such as LSTM and GRU due to their ineffectiveness at learning long sequences compared to the RNN-based Encoder-Decoder [19].

To evaluate the performance of our model, we also compared our method with other competitors, including Doctor-AI [11] and LIG-Doctor [12] for SDP. The details of the hyperparameters for all the models are given in Appendix A.

**Training details**: The data were randomly divided into training, test, and validation sets in an 90%:5%:5% ratio for training all models. We implemented all models using PyTorch [92]. We used the Ray Tune [93] library to find the optimal hyperparameters for both the tasks and evaluated the models on the test set. We performed the training of the models on a machine equipped with NVIDIA TITAN V and GeForce RTX 2070 GPUs.
4.4 Evaluation Metrics

We used Mean Average Recall (MAR) and Mean Average Precision (MAP) as recommendation metrics, where recall and precision are defined as below:

\[
\text{Recall}@k = \frac{\text{Total number of recommendations that are relevant in the top } k}{\text{Total number of all the possible relevant items}}
\]  
\[
\text{Precision}@k = \frac{\text{Total number of recommendations that are relevant in the top } k}{\text{Total number of predicted items}}
\]  
(4.20)  
(4.21)

We averaged over Eq. 4.20 and Eq. 4.21 to evaluate the performance of the model. MAP measures the relevance and order of the predicted medical codes, while MAR measures the most likely clinical events (such as \(D_x\), \(P_x\), and \(R_x\) codes).

As shown in Table 4.3 after data preprocessing, the average number of \(D_x\), \(P_x\), and \(R_x\) codes assigned per visit are 11.39, 3.05, and 32.40, respectively. The average number of medical codes (including \(D_x\), \(P_x\), and \(R_x\) codes) assigned at each patient visit is 46.84. Moreover, the average number of admissions per patient is 2.66. Hence, for TF tasks, we will set \(k\) to be 250 (i.e., around five times the average number of medical codes assigned per visit). This would signify how our model performed for the forecasted patients’ medical codes for the first five consecutive visits. Also, deep cut-offs will ensure that simple and complex cases are considered for evaluating the model’s performance.

Similarly, for SDP tasks we only forecast diagnosis codes for the next visit, the average number of diagnoses assigned per visit is 11.39. Therefore, we set \(k\) to be 60.
4.5 Results and Discussion

Table 4.7: **MAR** and **MAP** of algorithms in all three scenarios of the trajectory forecasting task, where $F_D$ is forecasting $D_x$ code only, $F_{DP}$ is forecasting $D_x$ and $P_x$ codes, and $F_{DPR}$ is forecasting $D_x$, $P_x$, and $R_x$ codes. The values are given in percentages.

| Algorithm                  | $F_D$ (MAR | MAP)       | $F_{DP}$ (MAR | MAP)       | $F_{DPR}$ (MAR | MAP)       |
|----------------------------|-------------|-------------|-------------|-------------|-------------|
|                            | $K = 100$   | $K = 150$   | $K = 250$   | $K = 100$   | $K = 150$   | $K = 250$   |
| GRU Encoder-Decoder        | 46.27 | 67.11 | 46.43 | 64.51 | 46.71 | 61.13  |
| Transformer                | 54.51 | 74.47 | 54.57 | 74.40 | 54.58 | 74.40  |
| Clinical-GAN               | **57.03** | **76.59** | **57.04** | **76.57** | **57.04** | **76.57** |
| GRU Encoder-Decoder        | 40.86 | 63.35 | 41.89 | 61.52 | 43.44 | 60.13  |
| Transformer                | 51.83 | 70.49 | 51.93 | 70.40 | 51.94 | 70.39  |
| Clinical-GAN               | **55.92** | **72.87** | **56.08** | **72.76** | **56.10** | **72.75** |
| GRU Encoder-Decoder        | 16.33 | 62.65 | 22.68 | 54.19 | 25.31 | 53.24  |
| Transformer                | 36.52 | 63.10 | 37.75 | 63.72 | 38.25 | 63.70  |
| Clinical-GAN               | **38.50** | **63.26** | **39.67** | **63.84** | **40.19** | **63.90** |

$D_x$: diagnosis, $P_x$: procedure, $R_x$: medication

Table 4.7 summarizes the results of the three scenarios in TF task. Clinical-GAN has achieved 76.57%, 72.75%, and 63.90% **MAP@250** and 57.04%, 56.10%, and 40.19% **MAP@250**. 
MAR@250 for the TF task in the $F_D$, $F_{DP}$, and $F_{DPR}$ scenarios respectively, which is comparable to the Transformer method. In each scenario, values of 100, 150, and 250 were tried for $k$. The values of $k$ at 100, 150, and 250 indicate the performance of the forecasted medical codes for the first two visits, three visits, and five visits, respectively. For all values of $k$, the Clinical-GAN algorithm had the highest MAP and MAR scores in all three scenarios.

To check whether the difference in precision and recall scores is statistically significant between the Transformer and Clinical-GAN, we conducted an approximate randomization test [94] between the two models on the three scenarios. We used the implementation as described in [95]. We used MAP@250 and MAR@250 as our evaluation and set the significance level as 0.05. From Table 4.8, all p-values from the approximate randomization tests are smaller than 0.05, demonstrating that our method’s performance over Transformer is statistically significant. Furthermore, a high precision score suggests that our proposed method can learn sequence relationships among the medical codes.

Table 4.8: Approximate randomization test results for statistical significance testing between our method and Transformer in all three scenarios using MAR@250 and MAP@250.

<table>
<thead>
<tr>
<th>p-value</th>
<th>$F_D$</th>
<th>$F_{DP}$</th>
<th>$F_{DPR}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAR@250</td>
<td>0.002</td>
<td>0.002</td>
<td>0.039</td>
</tr>
<tr>
<td>MAP@250</td>
<td>0.033</td>
<td>0.041</td>
<td>0.045</td>
</tr>
</tbody>
</table>

As shown in Table 4.9, our method outperformed other approaches by achieving 65.27% MAR@60 for the SDP task. It is clear from Table 4.7 that MAR@250 and
MAP@250 are higher when forecasting $D_x$ than when forecasting $D_x, P_x$, and $R_x$ together. Lower scores occur because the hypothesis space is larger when forecasting all medical codes jointly than when forecasting only diagnosis codes.

Table 4.9: MAR of algorithms in the sequential disease prediction task

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$K = 20$</th>
<th>$K = 40$</th>
<th>$K = 60$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doctor-AI</td>
<td>35.45%</td>
<td>44.89%</td>
<td>51.19%</td>
</tr>
<tr>
<td>LIG-Doctor</td>
<td>33.45%</td>
<td>46.95%</td>
<td>53.34%</td>
</tr>
<tr>
<td>Clinical-GAN</td>
<td>58.11%</td>
<td>63.39%</td>
<td>65.27%</td>
</tr>
</tbody>
</table>

**Model interpretation:** As described in Eq. 3.19, we calculated the influence for each input datum. The result is a two-dimensional matrix, where each row represents the percentage of each input’s contribution associated with the predicted variable.

The visualization shown in Figure 4.1 demonstrates that HEP010-24.7%, DIG017-16.6%, NEO070-6.2%, and DR_64253033335-13.6% contributed the most in predicting the NEO070 code for the next visit. Thus, we can infer from the visualization which input medical codes are influential in generating the output sequences from this methodology. We provide detailed examples of Clinical-GAN output in Appendix B.
Figure 4.1: This figure shows a visualization of the trajectory forecasting of a patient’s visit record. The x-axis corresponds to the input of past medical history comprised of medical codes, and the y-axis corresponds to the forecasted medical codes. The contribution of each medical code in the input towards predicting the variable in the output is summarized by the percentage along the x-axis. The intensity of the green colour represents the strength of the contribution.
Furthermore, we examined the model performance of the five most and least commonly diagnosed diseases in MIMIC-IV and summarized the results in Tables 4.10 and 4.11. We found that our model performs well on most of the diseases.

Table 4.10: Accuracy of Clinical-GAN on the five least common diseases in MIMIC-IV for the TF task in the $F_D$ scenario.

<table>
<thead>
<tr>
<th>Diagnosis code</th>
<th>Diagnosis description</th>
<th>MAP@250</th>
<th>MAR@250</th>
</tr>
</thead>
<tbody>
<tr>
<td>677</td>
<td>Late effect of complication of pregnancy, childbirth, and the puerperium</td>
<td>65.34</td>
<td>59.75</td>
</tr>
<tr>
<td>S51812A</td>
<td>Laceration without foreign body of left forearm, initial encounter</td>
<td>65.98</td>
<td>59.60</td>
</tr>
<tr>
<td>S2221XA</td>
<td>Fracture of manubrium, initial encounter for closed fracture</td>
<td>63.95</td>
<td>57.70</td>
</tr>
<tr>
<td>Z8669</td>
<td>Personal history of other diseases of the nervous system and sense organs</td>
<td>62.72</td>
<td>55.72</td>
</tr>
<tr>
<td>V1049</td>
<td>Personal history of malignant neoplasm of other male genital organs</td>
<td>63.46</td>
<td>57.94</td>
</tr>
</tbody>
</table>
Table 4.11: Accuracy of Clinical-GAN on the five most common diseases in MIMIC-IV for the TF task in the $F_D$ scenario.

<table>
<thead>
<tr>
<th>Diagnosis code</th>
<th>Diagnosis description</th>
<th>MAP@250</th>
<th>MAR@250</th>
</tr>
</thead>
<tbody>
<tr>
<td>4019</td>
<td>Unspecified essential hypertension</td>
<td>84.74</td>
<td>73.92</td>
</tr>
<tr>
<td>I10</td>
<td>Essential (primary) hypertension</td>
<td>84.25</td>
<td>73.39</td>
</tr>
<tr>
<td>E785</td>
<td>Hyperlipidemia, unspecified</td>
<td>84.82</td>
<td>72.95</td>
</tr>
<tr>
<td>53081</td>
<td>Esophageal reflux</td>
<td>85.07</td>
<td>74.91</td>
</tr>
<tr>
<td>25000</td>
<td>Diabetes mellitus without mention of complication, type II or unspecified type, not stated as uncontrolled</td>
<td>81.38</td>
<td>72.86</td>
</tr>
</tbody>
</table>

Figure 4.2 demonstrates that our proposed method performed well on long sequences and a higher number of visits. Both plots in Figure 4.2 show no significant degradation with the length of sequences and visits. However, there is a slight degradation in Figure 4.2b when the number of visits is higher than eight, but it stabilizes after a visit length of 10.

Additionally, from Figure 4.2, we infer that our proposed model performs better when more data is available on a patient. The accuracy is lower when the input data has one visit and gradually increases with the number of visits up to six visits. This behavior is due to a lack of historical information about a patient. In addition, there can be multiple trajectories for a certain diagnosis due to multimorbidity and
comorbidity characteristics of diseases. As a result, the model’s ability to forecast an accurate trajectory is lower for the initial visit due to the lack of information.

Figure 4.2: Both plots represent the MAR@250 and MAP@250 scores of the trajectory forecasting in scenario $F_D$ using our method on the test set. The left plot shows the accuracy as a function of input sequence length, where the x-axis corresponds to the input data sorted by their length. The right plot shows the accuracy as a function of the number of visits in the input data, where the x-axis corresponds to the input data sorted by their number of visits to the hospital.

The model’s understanding of the correlation between medical codes will provide better accuracy in forecasting patient trajectories. Further, as the role of the encoder part of the generator network is to learn an effective representation of the medical codes, this learned representation is then used by the decoder network to predict the future medical codes. However, the medical codes are complex as they are multi-modal data, and also, the multi-morbidity and comorbidity characteristics of diseases would hinder the model’s ability to learn the relationships between each medical code. Hence, poorly learned representation of medical codes could negatively impact the model’s accuracy.

Therefore, to understand the model behaviour, we projected the input embedding
of the Generator network into a two-dimensional space using t-SNE \cite{38}. More specifically, we used cosine similarity to find the top 20 associated codes of a particular disease from our final embedding (see Table 4.12) and visualized them in a 2D graph using t-SNE as shown in Figure 4.3.

Finding the associated top 20 codes for a specific diagnosis using cosine similarity would provide an understanding of how effectively our model has learned the correlation between medical codes. In addition, with the t-SNE representation of the medical codes into a 2D map, we could qualitatively validate whether the associated diagnosis, procedure, and medication codes are clustered together, indicating that our model has captured the correlation between the medical codes.

Table 4.12: Top 20 associated medical codes for Asthma calculated using cosine similarity from our final embedding. The valid medical codes are underlined. The code descriptions shown in the table are CCS and CCSR descriptions. Some of the diagnoses and procedures can be repeated because we are considering both ICD-9 and ICD-10 codes.

<table>
<thead>
<tr>
<th>Diagnosis: COPD, Chronic obstructive pulmonary disease and bronchiectasis, Ot compl bir, OB-related perin trauma, Other pregnancy and delivery including normal, Asthma, External cause codes: motor vehicle traffic (MVT); initial encounter, Postprocedural or postoperative musculoskeletal system complication, Abdominal pain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asthma</td>
</tr>
<tr>
<td>Procedures: Thoracentesis (diagnostic)</td>
</tr>
<tr>
<td>Medications: Fluticasone Propionate 110mcg, Albuterol Inhaler, Fluticasone-Salmeterol Diskus (250/50), Fluticasone-Salmeterol Diskus (100/50), Ipratropium-Albuterol Inhalation Spray, Ferrous Sulfate, Methadone, NF* Zafirlukast, Omeprazole, Fluticasone-Salmeterol Diskus (500/50)</td>
</tr>
</tbody>
</table>
From Table 4.12 and Figure 4.3, it is evident that our proposed method captures the relationship between diagnosis, procedure, and medication codes. For instance, the associated codes mentioned in Table 4.12 for Asthma, such as Chronic obstructive pulmonary disease and bronchiectasis ($D_x$), Abdominal pain ($D_x$), Thoracentesis (diagnostic) ($P_x$), Fluticasone-Salmeterol Diskus (250/50) ($R_x$), and Albuterol Inhaler ($R_x$), are clustered together when projected to t-SNE as shown in Figure 4.3. Similarly, the clusters of relevant codes associated with Breast cancer, Essential hypertension, and Heart failure are shown in Table 4.13 and in Figures 4.4, 4.5, and 4.6, respectively.

Additionally, we projected all medical codes into a 2D graph using t-SNE as shown
We found that every diagnosis code is surrounded by at least one or more procedure or medication codes. This further suggests that our model learns the correlations between medical codes effectively.

Table 4.13: Top 20 associated medical codes for Breast cancer, Essential hypertension, and Heart failure. The valid codes are underlined.

<table>
<thead>
<tr>
<th>Diagnosis</th>
<th>Procedures</th>
<th>Medications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breast cancer</td>
<td><strong>Hepatitis, Gastritis, Breast cancer - all other types, Headache/mig, Chr kidney disease, Kidney/rnl ca, Non-Hodg lym, Acquired absence of limb or organ, Pancreas can, Maint chem/r, G1/perit can, Personal/family history of disease</strong></td>
<td><strong>Breast reconstruction, Administration of thrombolitics and platelet inhibitors</strong></td>
</tr>
<tr>
<td>Essential hypertension</td>
<td><strong>HTN, Hypertension with complications and secondary hypertension, HTN in preg, Acute and unspecified renal failure, Osteoarthros, Hyperlipidem, Htn complicn, Uterus canc</strong></td>
<td><strong>-</strong></td>
</tr>
<tr>
<td>Heart failure</td>
<td><strong>Myocarditis and cardiomyopathy, Ac renl fail, Carditis, Unclassified, Essential hypertension, Hepatitis, Non-pressure ulcer of skin, Ot nutrit dx, Other specified diseases of veins and lymphatics, Coagulation and hemorrhagic disorders, Other specified and unspecified liver disease, Oth low resp</strong></td>
<td><strong>Thoracentesi</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Diagnosis</th>
<th>Procedures</th>
<th>Medications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breast cancer</td>
<td><strong>Hepatitis, Gastritis, Breast cancer - all other types, Headache/mig, Chr kidney disease, Kidney/rnl ca, Non-Hodg lym, Acquired absence of limb or organ, Pancreas can, Maint chem/r, G1/perit can, Personal/family history of disease</strong></td>
<td><strong>Breast reconstruction, Administration of thrombolitics and platelet inhibitors</strong></td>
</tr>
<tr>
<td>Essential hypertension</td>
<td><strong>HTN, Hypertension with complications and secondary hypertension, HTN in preg, Acute and unspecified renal failure, Osteoarthros, Hyperlipidem, Htn complicn, Uterus canc</strong></td>
<td><strong>-</strong></td>
</tr>
<tr>
<td>Heart failure</td>
<td><strong>Myocarditis and cardiomyopathy, Ac renl fail, Carditis, Unclassified, Essential hypertension, Hepatitis, Non-pressure ulcer of skin, Ot nutrit dx, Other specified diseases of veins and lymphatics, Coagulation and hemorrhagic disorders, Other specified and unspecified liver disease, Oth low resp</strong></td>
<td><strong>Thoracentesi</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Diagnosis</th>
<th>Procedures</th>
<th>Medications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breast cancer</td>
<td><strong>Hepatitis, Gastritis, Breast cancer - all other types, Headache/mig, Chr kidney disease, Kidney/rnl ca, Non-Hodg lym, Acquired absence of limb or organ, Pancreas can, Maint chem/r, G1/perit can, Personal/family history of disease</strong></td>
<td><strong>Breast reconstruction, Administration of thrombolitics and platelet inhibitors</strong></td>
</tr>
<tr>
<td>Essential hypertension</td>
<td><strong>HTN, Hypertension with complications and secondary hypertension, HTN in preg, Acute and unspecified renal failure, Osteoarthros, Hyperlipidem, Htn complicn, Uterus canc</strong></td>
<td><strong>-</strong></td>
</tr>
<tr>
<td>Heart failure</td>
<td><strong>Myocarditis and cardiomyopathy, Ac renl fail, Carditis, Unclassified, Essential hypertension, Hepatitis, Non-pressure ulcer of skin, Ot nutrit dx, Other specified diseases of veins and lymphatics, Coagulation and hemorrhagic disorders, Other specified and unspecified liver disease, Oth low resp</strong></td>
<td><strong>Thoracentesi</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Diagnosis</th>
<th>Procedures</th>
<th>Medications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breast cancer</td>
<td><strong>Hepatitis, Gastritis, Breast cancer - all other types, Headache/mig, Chr kidney disease, Kidney/rnl ca, Non-Hodg lym, Acquired absence of limb or organ, Pancreas can, Maint chem/r, G1/perit can, Personal/family history of disease</strong></td>
<td><strong>Breast reconstruction, Administration of thrombolitics and platelet inhibitors</strong></td>
</tr>
<tr>
<td>Essential hypertension</td>
<td><strong>HTN, Hypertension with complications and secondary hypertension, HTN in preg, Acute and unspecified renal failure, Osteoarthros, Hyperlipidem, Htn complicn, Uterus canc</strong></td>
<td><strong>-</strong></td>
</tr>
<tr>
<td>Heart failure</td>
<td><strong>Myocarditis and cardiomyopathy, Ac renl fail, Carditis, Unclassified, Essential hypertension, Hepatitis, Non-pressure ulcer of skin, Ot nutrit dx, Other specified diseases of veins and lymphatics, Coagulation and hemorrhagic disorders, Other specified and unspecified liver disease, Oth low resp</strong></td>
<td><strong>Thoracentesi</strong></td>
</tr>
</tbody>
</table>
Figure 4.4: t-SNE representation of the associated medical codes for Breast cancer.
Figure 4.5: t-SNE representation of the associated medical codes for Essential hypertension.
Figure 4.6: t-SNE representation of the associated medical codes for Heart failure.
Figure 4.7: t-SNE representation of all the codes together, where each diagnosis code is associated with at least one or more procedure and medication codes. This demonstrates that our proposed method is effective at learning the relationships between medical codes.

We found that some of the associated medical codes generated by cosine similarity for particular diagnoses can overlap with other diagnoses codes, resulting in codes being sparse in the 2D graph even though they are relevant. For instance, from Figures 4.3 and 4.6 we can see that the thoracentesis procedure is associated with asthma as well as heart failure. The position of the thoracentesis in the graph depends upon
the frequency of the thoracentesis appearing together with a specific diagnosis in the dataset. The greater the frequency, the more closely the corresponding medical code will be plotted in the graph. Hence, some medical codes tend to be closer to a specific diagnosis code than others, even though they are relevant. Nevertheless, this approach can be used to understand the comorbidities of certain diseases. More examples of disease-associated medical codes and their t-SNE representations are provided in Appendix C.

There are limitations to the proposed method. Firstly, the data representation is based on the MIMIC-IV database schema, where the diagnosis, procedure, and medication codes are stored in a sequential manner. Therefore, our method is only applicable to other EHR datasets, if they align with the MIMIC-IV schema. Secondly, ICD-10 codes have been commissioned to replace ICD-9 codes from 2015 onwards. Despite its discontinuation, the historical data uses ICD-9 codes representation, and there is no mapping from ICD-9 to ICD-10 codes. To handle this, we took both ICD-9 and ICD-10 into consideration. Unfortunately, this increases the hypothesis space, which hinders the accuracy of the model. Thirdly, the GAN architecture is computationally expensive. It requires a larger number of parameters than Transformer architecture, as demonstrated in Table 4.14, and thus is more computationally expensive. Finally, our approach towards interpretation of our method is limited to local interpretability of an individual instance.
Table 4.14: Total number of model parameters and training time of algorithms in all three scenarios.

<table>
<thead>
<tr>
<th>Model parameters</th>
<th>Training time (hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$F_D$</td>
</tr>
<tr>
<td>GRU Encoder-Decoder</td>
<td>9,047,229</td>
</tr>
<tr>
<td>Transformer</td>
<td>5,672,637</td>
</tr>
<tr>
<td>Clinical-GAN</td>
<td>7,559,358</td>
</tr>
</tbody>
</table>

4.6 Summary

In this chapter, we briefly discussed the MIMIC-IV dataset schema and presented its overall statistics related to patients’ demographics, diagnosis, procedure, and medication codes. We also discussed how to tackle the challenges of high granularity in ICD-9 and ICD-10 codes using CCS and CCSR mapping mechanisms. Furthermore, we covered the data preparation and preprocessing steps.

We empirically showed the superior performance of the Clinical-GAN over other baseline methods on TF and SDP tasks, using Mean Average Recall (MAR) and Mean Average Precision (MAP) as our evaluation metrics. Moreover, we demonstrated the performance of our model on the top five most and least common diseases based on MAP@250 and MAR@250 metrics. In addition to that, we showed that our model could perform efficiently on long sequences and with a large number of visits in the input data. We qualitatively analyzed the final embedding of our model using cosine similarity and t-SNE representations of disease-associated medical codes. Our findings suggest that our proposed method learns the correlation between medical codes effectively. Finally, we discussed the limitations of our framework.
Chapter 5

Conclusion and Future Work

In this thesis, we addressed the research problem stated below:

"Given the past medical codes of a patient’s visits, how should the patient’s trajectory be forecasted?"

We solved the research problem by answering the following three research questions:

1. How should the data be modelled such that the deep learning model is able to forecast the patient’s medical codes for the subsequent visits?

2. How should a deep learning architecture be built that will be well suited for the research problem?

3. How should the impact of input features on the model’s predictions be computed?

Our main contribution to answering the above research questions is as follows:

1. We introduced a novel data representation method for input and output features that helps to forecast the patient’s medical codes for the subsequent visits. We use the sequential order of diagnosis, procedure, and medication codes in our data representation and consider each code as a unique token. Then, we group them together and use three additional tokens to distinguish between each visit of a patient.
2. We proposed a novel architecture called Clinical-GAN, a Transformer-based GAN model, that tackles the problem of misdiagnosis error by forecasting patient trajectory. Our method takes the Transformer Encoder-Decoder framework as the Generator network and the Transformer Encoder as the Discriminator network. Furthermore, both the networks are trained adversarially to update the model parameters.

3. We demonstrated the local interpretability of individual outcomes generated by our model using multi-head attention.

Our initiative in forecasting patient trajectories can help physicians effectively assess patients’ health conditions and give appropriate care based on their forecasted medical codes for the subsequent visits. Consequently, not only could the patient’s quality of care be enhanced, but healthcare costs could also be reduced.

We conducted our experiments on the real-world, publicly available dataset MIMIC-IV v1.0 and incorporated the CCS and CCSR mapping mechanisms to reduce the high granularity of the ICD-9 and ICD-10 codes.

We tested our proposed model on two tasks: TF and SDP. Firstly, in the TF task, Clinical-GAN achieved statistically significant results compared to the baseline methods. Secondly, in the SDP task, our proposed methodology outperformed the existing works. We further showed that our model performs well on long input sequences comprised of multiple visits by a patient.

We used cosine similarity from the trained model to predict the associated codes of specific diagnoses and projected them into a two-dimensional space using t-SNE. Our qualitative analysis of the associated codes and 2D space showed that our proposed method learns the correlation between the diagnosis, procedure, and medication codes. Therefore, we believe that the proposed method is a promising step towards patient
trajectory forecasting, and the approach to predict the associated codes using cosine similarity can lead us to understand comorbidities.

The proposed method uses the vanilla Transformer and WGAN. Hence, future work should explore transfer learning and advanced NLP concepts such as Transformer-based pre-training techniques. Further work should also utilize the additional static features of the patient, such as age and gender. In addition, the mortality of a patient, together with the medical codes, could be utilized to predict on which admission the patient is going to die. Moreover, further studies should explore the global interpretation and counterfactual explanation of the model to enhance the visualization and interpretation.

Finally, without conducting clinical trials, it is challenging to assess the true potential of Clinical-GAN. Physicians would be the most qualified to evaluate the effectiveness of this approach. Long-term research with explicit testing by medical professionals would give us a better insight into what would be the best next step.
References


[5] Hardeep Singh, Traber Davis Giardina, Ashley N.D. Meyer, Samuel N. Forjuoh,


[18] Bryan Lim and Mihaela van der Schaar. Disease-atlas: Navigating disease


International Conference on Knowledge Discovery and Data Mining, pages 1095–1103, 2012.


[84] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional


Appendix A

Hyperparameter Details

We used the Adam optimizer [96] for all the models and in the Generator network of Clinical-GAN. In addition, we applied a Stochastic Gradient Descent (SGD) optimizer [97] to the Discriminator network in Clinical-GAN. We used the Noam optimizer from the original Transformer architecture [24] to vary the learning rate over the training, with an initial learning rate of 4e-4 in both the Transformer and Clinical-GAN models. We ran all models of the TF task for 100 epochs and used early stopping based on validation loss to select the best model during the training process.
Table A.1: Hyperparameter details

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Clinical-GAN</th>
<th>Transformer</th>
<th>GRU Encoder-Decoder</th>
<th>Doctor-AI</th>
<th>LIG-Doctor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning rate</td>
<td>4e-4</td>
<td>4e-4</td>
<td>0.0005</td>
<td>0.00005</td>
<td>0.0005</td>
</tr>
<tr>
<td>G Encoder/Decoder layer</td>
<td>3</td>
<td>3</td>
<td>0.0005</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dropout</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>Clip</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alpha</td>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Heads</td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Embedding size</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
</tr>
<tr>
<td>Hidden size</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>2000</td>
<td>272</td>
</tr>
<tr>
<td>Batch size</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>100</td>
</tr>
<tr>
<td>D$_{\text{Disc}}$ layer</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D$_{\text{Disc}}$ heads</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D$_{\text{Disc}}$ gradient clipping</td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Epochs</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>20</td>
<td>200</td>
</tr>
<tr>
<td>Warm-up steps</td>
<td>30</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of layers</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Generator, *Discriminator.
Table A.2: Hyperparameter search space

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Clinical-GAN</th>
<th>Transformer</th>
<th>GRU Encoder-Decoder</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Learning rate</strong></td>
<td>1e-5, 2e-4, 3e-4,</td>
<td>1e-5, 2e-4, 3e-4,</td>
<td>1e-5, 2e-5, 3e-5,</td>
</tr>
<tr>
<td></td>
<td>4e-4, 5e-5, 6e-5,</td>
<td>4e-4, 5e-5, 6e-5,</td>
<td>4e-5, 5e-5, 6e-5,</td>
</tr>
<tr>
<td></td>
<td>2e-7, 1e-7, 3e-7,</td>
<td>2e-7, 1e-7, 3e-7,</td>
<td>2e-7, 1e-7, 3e-7,</td>
</tr>
<tr>
<td></td>
<td>2e-8</td>
<td>2e-8</td>
<td>2e-8</td>
</tr>
<tr>
<td><strong>Encoder/Decoder layer</strong></td>
<td>1-4</td>
<td>1-4</td>
<td>2-8</td>
</tr>
<tr>
<td><strong>Dropout</strong></td>
<td>0.1-0.5</td>
<td>0.1-0.5</td>
<td>0.1-0.5</td>
</tr>
<tr>
<td><strong>Clip</strong></td>
<td>0.1-1.0</td>
<td>0.1-1.0</td>
<td>0.1-1.0</td>
</tr>
<tr>
<td><strong>Alpha</strong></td>
<td>0.1-1.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Heads</strong></td>
<td>2, 4, 8, 16</td>
<td>2, 4, 8, 16</td>
<td></td>
</tr>
<tr>
<td><strong>Embedding size</strong></td>
<td>64, 128, 256, 512</td>
<td>64, 128, 256, 512</td>
<td>64, 128, 256, 512</td>
</tr>
<tr>
<td><strong>Hidden size</strong></td>
<td>128, 259, 512,</td>
<td>128, 259, 512,</td>
<td>128, 259, 512,</td>
</tr>
<tr>
<td></td>
<td>1024</td>
<td>1024</td>
<td>1024</td>
</tr>
<tr>
<td><strong>Batch size</strong></td>
<td>4, 8, 16</td>
<td>4, 8, 16</td>
<td>4, 8, 16</td>
</tr>
<tr>
<td><em><em>D</em> layer</em>*</td>
<td>1, 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td><em><em>D</em> heads</em>*</td>
<td>1-4</td>
<td></td>
<td></td>
</tr>
<tr>
<td><em><em>D</em> gradient clipping</em>*</td>
<td>0.1, 0.01, 0.9,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.09, 0.001</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Warm-up steps</strong></td>
<td>10, 20, 30</td>
<td>10, 20, 30</td>
<td></td>
</tr>
<tr>
<td><strong>Epochs</strong></td>
<td>100, 200</td>
<td>100, 200</td>
<td>100, 200</td>
</tr>
</tbody>
</table>

**Generator, *Discriminator.
Appendix B

Input and Output Samples of Clinical-GAN

In this appendix, we provide a few examples of Clinical-GAN’s output given an input with respect to the three scenarios ($F_D$, $F_{DP}$, $F_{DPR}$) in the trajectory forecasting task (see Tables B.1, B.2, B.3, B.4, B.5, B.6, B.7, B.8). Highlighted codes in the Forecasted data section signify that the predicted code is correct compared to Ground truth. All the ICD codes have a prefix of the type of medical code, followed by the version of the ICD it belongs to and an underscore.
Table B.1: Example 1 - Input for the $F_D$ scenario.

Table B.2: Example 1 - Ground truth and Clinical-GAN’s forecasted data for the input mentioned in the Table B.1

Table B.3: Example 2 - Input for the $F_D$ scenario.

<table>
<thead>
<tr>
<th>Input</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>'SOH', 'D10_INJ033', 'D10_CIR030', 'D10_CIR026', 'D10_CIR007', 'D10_INJ029', 'D10_FAC025', 'D10_MBD024', 'D10_MBD021', 'D10_EXT025', 'D10_EXT027', 'P10_CAR008', 'DR_00264958720', 'DR_00409198530', 'DR_63323001302', 'DR_00338004904', 'DR_63323018410', 'DR_00338004904', 'DR_63323026201', 'DR_00641607825', 'DR_51079055803', 'DR_51079021020', 'DR_00338011704', 'DR_68084034701', 'DR_66758016013', 'DR_76439034310', 'DR_00904198261', 'DR_00641607825', 'DR_6739049910', 'DR_00904629461', 'DR_50458057910', 'EOV', 'D10_INJ033', 'D10_INJ033', 'D10_FAC016', 'D10_EXT027', 'D10_CIR011', 'D10_FAC009', 'D10_SYM002', 'D10_CIR012', 'D10_CIR026', 'D10_MBD017', 'D10_MBD021', 'D10_FAC025', 'D10_DIG004', 'P10_CAR014', 'DR_00338004904', 'DR_63739027201', 'DR_00409128331', 'DR_00071015892', 'DR_00904224461', 'DR_51079010520', 'DR_51079021020', 'DR_00904053061', 'DR_00904526161', 'DR_00409180001', 'DR_00338070341', 'DR_00264958720', 'DR_57896045208', 'DR_00904224461', 'DR_00904404073', 'DR_45802011222', 'DR_68084034701', 'DR_00338001702', 'DR_00641614225', 'DR_00409117630', 'DR_00121065721', 'DR_00703450204', 'DR_00409909332', 'DR_00338004903', 'DR_00409665305', 'DR_00056017275', 'DR_00075062280', 'DR_00245004101', 'EOV', 'EOH'</td>
<td></td>
</tr>
</tbody>
</table>
Table B.4: Example 2 - Ground truth and Clinical-GAN’s forecasted data for the input mentioned in the Table B.3

Table B.5: Example 3 - Ground truth and Clinical GAN’s forecasted data for the input in the $F_{DP}$ scenario.

Table B.6: Example 4 - Ground truth and Clinical GAN’s forecasted data for the input in the $F_{DP}$ scenario. In this example, Clinical-GAN forecasted only one visit instead of two compared to the ground truth. However, the model captured all the unique codes assigned to that patient relative to the ground truth. Surprisingly, it gave all the ground truth’s second visit codes of a patient as the first visit codes in the forecasted data.

| Input: | 'SOH', 'D10_GEN020', 'D10_CIR007', 'D10_GEN008', 'D10_DIG004', 'D10_END010', 'P10_FRS001', 'P10_FRS004', 'P10_MST028', 'P10URN012', 'P10_FRS015', 'P10URN001', 'DR_51293080101', 'DR_00641607825', 'DR_00409379301', 'DR_00409128331', 'DR_00904224461', 'DR_00904198261', 'DR_00338011704', 'DR_00064930100', 'DR_6379035810', 'DR_00406055262', 'DR_00904585361', 'DR_00078035834', 'DR_1679018201', 'DR_00071015892', 'EOV', 'EOH' |
| Ground truth: | 'SOH', 'D10_MUS006', 'D10_CIR007', 'D10_DIG004', 'P10_MST006', 'EOV', 'D10_MUS006', 'D10_FAC009', 'D10_CIR007', 'D10_FAC021', 'D10_END010', 'D10_SYM002', 'D10_EXT025', 'D10_EXT027', 'P10_MST006', 'EOV', 'EOH' |
| Forecasted data: | 'SOH', 'D10_MUS006', 'D10_FAC009', 'D10_CIR007', 'D10_FAC021', 'D10_END010', 'D10_SYM002', 'D10_EXT025', 'D10_EXT027', 'P10_MST006', 'EOV', 'EOH' |
Table B.7: Example 5 - Ground truth and Clinical GAN’s forecasted data for the input in the $F_{DPR}$ scenario.

|--------------|------------------------------------------------------------------------------|
Table B.8: Example 6 - Ground truth and Clinical GAN’s forecasted data for the input in the $F_{DPR}$ scenario.

| Ground truth:              | 'SOH', 'D9_191', 'D9_196', 'D9_193', 'P9_140', 'P9_136', 'P9_137', 'DR_00904224461', 'DR_66553000401', 'DR_00182864389', 'DR_00168004631', 'DR_00121043130', 'DR_00574705050', 'DR_00904198861', 'DR_00054815624', 'DR_00904585461', 'DR_00406051262', 'DR_00802396216', 'DR_63323001201', 'DR_00338011704', 'EOV', 'EOH' |
| Forecasted data:          | 'SOH', 'D9_193', 'D9_196', 'P9_140', 'P9_139', 'DR_00904585461', 'DR_00054815624', 'DR_00182845389', 'DR_00574705050', 'DR_00182864389', 'DR_66553000401', 'DR_00904224461', 'DR_00406051262', 'EOV', 'EOH' |
Appendix C

Disease-Associated Medical Codes and their t-SNE Representation

This section lists a few disease-related medical codes (see Tables C.1, C.2, C.3, C.4, C.5, C.6) and their t-SNE representation calculated using cosine similarity from our proposed model’s final embedding (see Figures C.1, C.2, C.3, C.4, C.5, C.6). The valid medical codes are underlined. The code descriptions shown in the table are CCS and CCSR descriptions. Some of the diagnoses and procedures can be repeated as we are considering both ICD-9 and ICD-10 codes.
Table C.1: Top 20 associated medical codes for Diabetes mellitus, Type 2.

<table>
<thead>
<tr>
<th>Diagnosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diabetes mellitus with complication,</td>
</tr>
<tr>
<td>Diabetes mellitus without complication,</td>
</tr>
<tr>
<td>Hyperlipidem, Other specified and unspecified</td>
</tr>
<tr>
<td>gastrointestinal disorders, Other gastrointestinal disorder, Essential hypertension, Implant, device or graft related encounter, Other acquired deformities, Disorders of lipid metabolism,</td>
</tr>
<tr>
<td>Thyroid disorders</td>
</tr>
<tr>
<td>Procedures: Other non-OR therapeutic procedures on musculoskeletal system</td>
</tr>
<tr>
<td>Medications: Cetirizine, Insulin, Ibuprofen,</td>
</tr>
<tr>
<td>Glucose Gel, MetFORMIN (Glucophage), Senna,</td>
</tr>
<tr>
<td>Albuterol Inhaler, 0.9% Sodium Chloride,</td>
</tr>
<tr>
<td>Potassium Chl 20 mEq / 1000 mL D5 1/2 NS,</td>
</tr>
<tr>
<td>Glucagon</td>
</tr>
</tbody>
</table>
Figure C.1: t-SNE representation of the associated medical codes for Diabetes mellitus, Type 2 mentioned in the Table C.1.
Table C.2: Top 20 associated medical codes for Anxiety and fear-related disorders.

| Diagnosis: | Anxiety disorders, Mood disorders, Ot preg comp, Biliary dx, Trauma- and stressor-related disorders, OB-related perin trauma, Esophgeal dx, Other GI dx, Gasduo ulcer, Headache/mig, Personal/family history of disease, OB-related trauma to perineum and vulva |
| Procedure: | ClonazePAM, ALPRAZolam, Clonazepam, Lorazepam, Lorazepam, Potassium Chloride, BusPIRone, Metclopramide, Diazepam, Albuterol 0.083% Neb Soln, Simvastatin |
Figure C.2: t-SNE representation of the associated medical codes for Anxiety and fear-related disorders mentioned in the Table C.2.
<table>
<thead>
<tr>
<th>Diagnosis:</th>
<th>HIV infection, Other pregnancy and delivery including normal, Fatigue, Headache; including migraine, Ot endo dsor, Hyperplasia of prostate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Procedures:</td>
<td>Peripheral arterial bypass procedures, Ot Dx US</td>
</tr>
<tr>
<td>Medications:</td>
<td>Morphine SR (MS Contin), Calcium Chloride, Efavirenz, Tamsulosin, Omeprazole, Isavuconazonium Sulfate, Venlafaxine XR, Acetaminophen, Minocycline, OLANZapine, Ranitidine, 0.9% Sodium Chloride, Fluconazole, Donepezil, Cepacol (Sore Throat Lozenge)</td>
</tr>
</tbody>
</table>
Figure C.3: \textit{t-SNE} representation of the associated medical codes for HIV infection mentioned in the Table C.3.
Table C.4: Top 20 associated medical codes for Pancreatic disorders (excluding diabetes).

| Pancreatic disorders (excluding diabetes) | Diagnosis: Endocrine system cancers - pancreas, Chronic phlebitis; thrombophlebitis and thromboembolism, Erectile dysfunction, Urinary system cancers - bladder, Thyroid disorders, Pleurisy, Pancreas dx | Procedures: Hepatobiliary and pancreatic drainage, Appendectomy, Pancreatectomy, Ultrasonography, nOR musc骷 | Medications: Creon 12, 5% Dextrose, PredniSONE, Gelclair, Ondansetron, 5% Dextrose (EXCEL BAG), Venlafaxine XR, Bisacodyl, GlipiZIDE XL, Influenza Virus Vaccine, Meperidine |
Figure C.4: t-SNE representation of the associated medical codes for Pancreatic disorders (excluding diabetes) mentioned in the Table C.4.
Table C.5: Top 20 associated medical codes for Dysrhythmia.

<table>
<thead>
<tr>
<th>Diagnosis:</th>
<th>Cardiac dysrhythmias, Urinary tract infections, Fluid and electrolyte disorders, Other specified status, Non-Hodgkin lymphoma</th>
</tr>
</thead>
<tbody>
<tr>
<td>Procedures:</td>
<td>Routine CXR, Cardiac cath, Cholecystec, Spontaneous vaginal delivery</td>
</tr>
<tr>
<td>Medications:</td>
<td>Warfarin, Amiodarone, Amiodarone, Rivaroxaban, Aspirin, Digoxin, Dexamethasone, Diltiazem, Potassium Chloride, HYDROMorphone, Simethicone, OxyCODONE (Immediate Release), Furosemide, Metoprolol Tartrate</td>
</tr>
</tbody>
</table>
Figure C.5: t-SNE representation of the associated medical codes for Dysrhythmia mentioned in the Table C.5.
Table C.6: Top 20 associated medical codes for Mood disorders.

| Diagnosis:  | Depressive disorders, Bipolar and related disorders, Anxiety and fear-related disorders, Esophgeal dx, Sleep wake disorders, |
| Mood disorders | Headache/mig, Ot compl bir, Ot preg comp, |
| Abdom hernia, Nausea/vomit |
| Procedures:  | Ot asst del, Cardiac cath |
| Medications: | Citalopram Hydrobromide, Sertraline, Fluoxetine, Escitalopram Oxalate, Sertraline, Sertraline, Ibuprofen, Albuterol Inhaler, Paroxetine, BuPROPion (Sustained Release), |
| Venlafaxine XR |
Figure C.6: t-SNE representation of the associated medical codes for Mood disorders mentioned in the Table C.6.
Publications