arXiv:1906.04716v1 [cs.LG] 11 Jun 2019

Graph Convolutional Transformer: Learning the Graphical Structure of Electronic Health Records

Edward Choi[1], **Zhen Xu**[1], **Yujia Li**[2], **Michael W. Dusenberry**[1],
Gerardo Flores[1], **Yuan Xue**[1], **Andrew M. Dai**[1]
[1] Google, USA [2] DeepMind, UK

Abstract

Effective modeling of electronic health records (EHR) is rapidly becoming an important topic in both academia and industry. A recent study showed that utilizing the graphical structure underlying EHR data (*e.g.* relationship between diagnoses and treatments) improves the performance of prediction tasks such as heart failure diagnosis prediction. However, EHR data do not always contain complete structure information. Moreover, when it comes to claims data, structure information is completely unavailable to begin with. Under such circumstances, can we still do better than just treating EHR data as a flat-structured *bag-of-features*? In this paper, we study the possibility of utilizing the implicit structure of EHR by using the Transformer for prediction tasks on EHR data. Specifically, we argue that the Transformer is a suitable model to learn the hidden EHR structure, and propose the Graph Convolutional Transformer, which uses data statistics to guide the structure learning process. Our model empirically demonstrated superior prediction performance to previous approaches on both synthetic data and publicly available EHR data on encounter-based prediction tasks such as graph reconstruction and readmission prediction, indicating that it can serve as an effective general-purpose representation learning algorithm for EHR data.

1 Introduction

Large medical records collected by electronic healthcare records (EHR) systems in healthcare organizations enabled deep learning methods to show impressive performance in diverse tasks such as predicting diagnosis [1–3], learning medical concept representations [4–7], and making interpretable predictions [8, 9]. As diverse as they are, one thing shared by all tasks is the fact that, under the hood, some form of neural network is processing EHR data to learn useful patterns from them. To successfully perform any EHR-related task, it is essential to learn effective representations of various EHR features: diagnosis codes, lab values, encounters, and even patients themselves. EHR data are typically stored in a relational database that can be represented as a hierarchical graph depicted in Figure 1. The common approach for processing EHR data with neural networks has been to treat each encounter as an unordered set of features, or in other words, a *bag of features*. However, the *bag of features* approach completely disregards the graphical structure that reflects the physician's decision process. For example, if we treat the encounter in Figure 1 as a *bag of features*, we will lose the information that *Benzonatate* was ordered because of *Cough*, not because of *Abdominal pain*.

Recently, motivated by this EHR structure, Choi et al. [10] proposed MiME, a model architecture that reflects EHR's encounter structure, specifically the relationships between the diagnosis and its treatment. MiME outperformed various *bag of features* approaches in prediction tasks such as heart failure diagnosis prediction. Their study, however, naturally raises the question: when the EHR data do *not* contain structure information (the red edges in Figure 1), can we still do better than *bag of features* in learning the representation of the data for various prediction tasks? This question

Preprint. Under review.

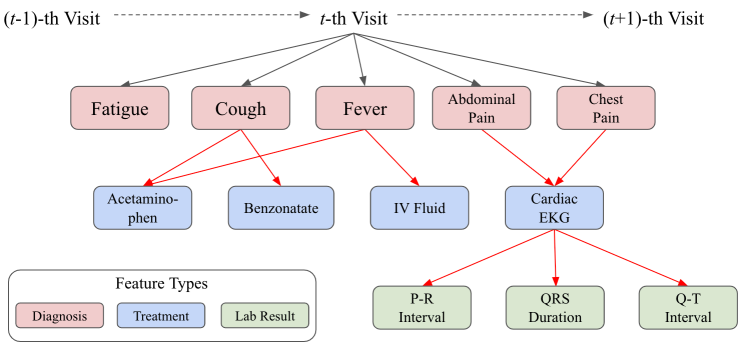


Figure 1: The graphical structure of electronic health records. A single visit consists of multiple types of features, and their connections (red edges) reflect the physician's decision process.

emerges in many occasions, since EHR data do not always contain the entire structure information. For example, some dataset might describe which treatment lead to measuring certain lab values, but might not describe the reason diagnosis for ordering that treatment. Moreover, when it comes to claims data, such structure information is completely unavailable to begin with.

To address this question, we study the possibility of using the Transformer [11] to utilize the unknown encounter structure for various prediction tasks when the structure information is unavailable. Specifically, we describe the graphical nature of encounter records, and argue that the Transformer is a reasonable model to discover implicit encounter structure. Then we propose the Graph Convolutional Transformer (GCT) to more effectively utilize the characteristics of EHR data while performing diverse prediction tasks. We test the Transformer and GCT on both synthetic data and real-world EHR records for encounter-based prediction tasks such as graph reconstruction and readmission prediction. In all tasks, GCT consistently outperformed baseline models, showing its potential to serve as an effective general-purpose representation learning algorithm for EHR data.

2 Related Work

Although there are recent works on medical concept embedding, focusing on patients [4, 7, 12, 13], visits [6], or codes [14, 15], the graphical nature of EHR has not been fully explored yet. Choi et al. [10] proposed MiME, which derives the visit representation in a bottom-up fashion according to the encounter structure. For example in Figure 1, MiME first combines the embedding vectors of lab results with the *Cardiac EKG* embedding, which in turn is combined with both the *Abdominal Pain* embedding and the *Chest Pain* embedding. Then all diagnosis embeddings are pooled together to derive the final visit embedding. By outperforming various bag-of-features models in heart failure prediction and general disease prediction, MiME demonstrated the usefulness of the structure information of encounter records.

The *Transformer* [11] was proposed for natural language processing, specifically machine translation. It uses a novel method to process sequence data using only attention [16], and is recently showing impressive performance in other tasks such as word representation learning [17]. *Graph (convolutional) networks* encompass various neural network methods to handle graphs such as molecular structures, social networks, or physical experiments. [18–21]. In essence, many graph networks can be described as different ways to aggregate a given node's neighbor information, combine it with the given node, and derive the node's latent representation [21].

Some recent works focused on the connection between the Transformer's self-attention and graph networks [20]. Graph Attention Networks [22] applied self-attention on top of the adjacency matrix to learn non-static edge weights, and [23] used self-attention to capture non-local dependencies in images. Although our work also relies on self-attention, our interest lies in whether the Transformer can be an effective tool to capture the underlying graphical structure of EHR data even when the structure information is missing, thus improving encounter-based prediction tasks. In the next section, we first describe the graphical nature of EHR encounter data, then show that the Transformer is a reasonable algorithm for learning the hidden graphical structure of encounter records.

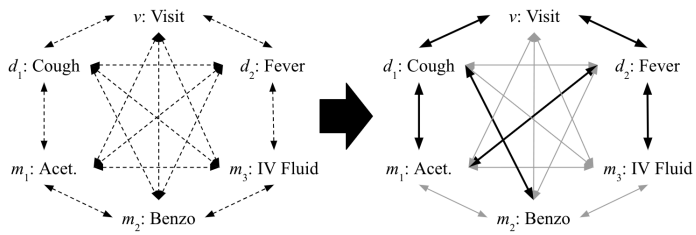


Figure 2: Learning the underlying structure of an encounter. We use Transformer to start from the left, where all nodes are implicitly fully-connected, and arrive at the right, where meaningful connections are described with thicker edges.

3 Method

3.1 Electronic Health Records as a Graph

As depicted in Figure 1, the t-th visit $\mathcal{V}^{(t)}$ starts with the visit node $v^{(t)}$ at the top. Beneath the visit node are diagnosis nodes $d_1^{(t)}, d_2^{(t)}, \ldots, d_{|d^{(t)}|}^{(t)}$, which in turn lead to ordering a set of treatments $m_1^{(t)}, m_2^{(t)}, \ldots, m_{|m^{(t)}|}^{(t)}$, where $|d^{(t)}|, |m^{(t)}|$ respectively denote the number of diagnosis and treatment codes in $\mathcal{V}^{(t)}$. Some treatments produce lab results $r_1^{(t)}, r_2^{(t)}, \ldots, r_{|r^{(t)}|}^{(t)}$, which may be associated with continuous values (*e.g.* blood pressure) or binary values (*e.g.* positive/negative allergic reaction). Since we focus on a single encounter in this study, we omit the time index t throughout the paper.

If we assume all features d_i, m_i, r_i[1] can be represented in the same latent space, then we can view an encounter as a graph consisting of $|d| + |m| + |r|$ nodes with an adjacency matrix $\mathbf{A}$ that describes the connections between the nodes. We use c_i as the collective term to refer to any of d_i, m_i, and r_i for the rest of the paper. Given c_i and $\mathbf{A}$, we can use graph networks or MiME[2] to derive the visit representation $\mathbf{v}$ and use it for downstream tasks such as heart failure prediction. However, if we do not have the structural information $\mathbf{A}$, which is the case in many EHR data and claims data, we typically use feed-forward networks to derive $\mathbf{v}$, which is essentially summing all node representations $\mathbf{c}_i$'s and projecting it to some latent space.

3.2 Transformer and Graph Networks

Even without the structure information $\mathbf{A}$, it is unreasonable to treat $\mathcal{V}$ as a bag of nodes c_i, because obviously physicians must have made some decisions when making diagnosis and ordering treatments. The question is how to utilize the underlying structure without explicit $\mathbf{A}$. One way to view this problem is to assume that all nodes c_i in $\mathcal{V}$ are implicitly fully-connected, and try to figure out which connections are stronger than the other as depicted in Figure 2. In this work, as discussed in section 2, we use Transformer to learn the underlying encounter structure. To elaborate, we draw a comparison between two cases:

- Case A: We know $\mathbf{A}$, hence we can use Graph Convolutional Networks (GCN). In this work, we use multiple hidden layers between each convolution, motivated by Xu et al. [21].

$$\mathbf{C}^{(j)} = \mathrm{MLP}^{(j)}(\tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}\mathbf{C}^{(j-1)}\mathbf{W}^{(j)}), \tag{1}$$

  where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, $\tilde{\mathbf{D}}$ is the diagonal node degree matrix[3] of $\tilde{\mathbf{A}}$, $\mathbf{C}^{(j)}$ and $\mathbf{W}^{(j)}$ are the node embeddings and the trainable parameters of the j-th convolution respectively. $\mathrm{MLP}^{(j)}$ is a multi-layer perceptron of the j-th convolution with its own trainable parameters.

- Case B: We do not know $\mathbf{A}$, hence we use Transformer, specifically the encoder with a single-head attention, which can be formulated as

$$\mathbf{C}^{(j)} = \mathrm{MLP}^{(j)}(\mathrm{softmax}(\frac{\mathbf{Q}^{(j)}\mathbf{K}^{(j)\top}}{\sqrt{d}})\mathbf{V}^{(j)}), \tag{2}$$

[1] If we bucketize the continuous values associated with r_i, we can treat r_i as a discrete feature like d_i, m_i.

[2] MiME is in fact, a special form of graph networks with residual connections.

[3] Xu et al. [21] does not use the normalizer $\tilde{\mathbf{D}}^{-1}$ to improve model expressiveness on multi-set graphs, but we include $\tilde{\mathbf{D}}^{-1}$ to make the comparison with Transformer clearer.

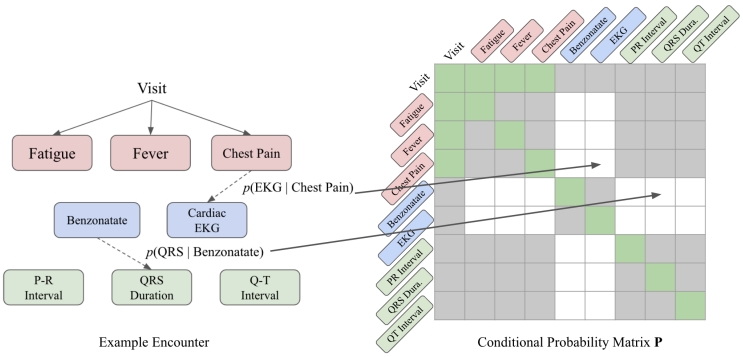


Figure 3: Creating the conditional probability matrix **P** based on an example encounter. The gray cells are masked to zero probability since those connections are not allowed. The green cells are special connections that we know are guaranteed to exist. We assign a pre-defined scalar value (*e.g.* 1) to the green cells. The white cells are assigned the corresponding conditional probabilities.

where $\mathbf{Q}^{(j)} = \mathbf{C}^{(j-1)}\mathbf{W}_Q^{(j)}$, $\mathbf{K}^{(j)} = \mathbf{C}^{(j-1)}\mathbf{W}_K^{(j)}$, $\mathbf{V}^{(j)} = \mathbf{C}^{(j-1)}\mathbf{W}_V^{(j)}$, and d is the column size of $\mathbf{W}_K^{(j)}$. $\mathbf{W}_Q^{(j)}$, $\mathbf{W}_K^{(j)}$, and $\mathbf{W}_V^{(j)}$ are trainable parameters of the j-th Transformer block[4]. Note that positional encoding using sine and cosine functions is not required, since features in an encounter are unordered.

Given Eq. 1 and Eq. 2, we can readily see that there is a correspondence between the normalized adjacency matrix $\tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}$ and the attention map $\text{softmax}(\frac{\mathbf{Q}^{(j)}\mathbf{K}^{(j)T}}{\sqrt{d}})$, and between the node embeddings $\mathbf{C}^{(j-1)}\mathbf{W}^{(j)}$ and the value vectors $\mathbf{C}^{(j-1)}\mathbf{W}_V^{(j)}$. In fact, GCN can be seen as a special case of Transformer, where the attention mechanism is replaced with the known, fixed adjacency matrix. Conversely, Transformer can be seen as a graph embedding algorithm that assumes fully-connected nodes and learns the connection strengths during training. Given this connection, it seems natural to use Transformer as an algorithm to learn the underlying structure of visits.

3.3 Graph Convolutional Transformer

Although Transformer can potentially learn the hidden encounter structure, without a single piece of hint, it must search the entire attention space to discover meaningful connections between encounter features. Therefore we propose Graph Convolutional Transformer (GCT), which, based on data statistics, restricts the search to the space where it is likely to contain meaningful attention distribution.

Specifically, we use 1) the characteristic of EHR data and 2) the conditional probabilities between features. First, we use the fact that some connections are not allowed in the encounter record. For example, we know that treatment codes can only be connected to diagnosis codes, but not to other treatment codes. Based on this observation, we can create a mask **M**, which will be used during the attention generation step. **M** has negative infinities where connections are not allowed, and zeros where connections are allowed.

Conditional probabilities can be useful for determining potential connections between features. For example, given *chest pain*, *fever* and *EKG*, without any structure information, we do not know which diagnosis is the reason for ordering *EKG*. However, we can calculate from EHR data that $p(\text{EKG}|\text{chest pain})$ is typically larger than $p(\text{EKG}|\text{fever})$, indicating that the connection between the former pair is more likely than the latter pair. Therefore we propose to use the conditional probabilities calculated from the encounter records as the guidance for deriving the attention. After calculating $p(m|d)$, $p(d|m)$, $p(r|m)$, and $p(m|r)$ from all encounter records for all diagnosis codes d, treatment codes m, and lab codes r, we can create a guiding matrix when given an encounter record, as depicted by Figure 3. We use $\mathbf{P} \in [0.0, 1.0]^{|c|\times|c|}$ to denote the matrix of conditional probabilities of all features, normalized such that each row sums to 1. Note that GCT's attention $\text{softmax}(\frac{\mathbf{Q}^{(j)}\mathbf{K}^{(j)\top}}{\sqrt{d}})$, the mask **M**, and the conditional probabilities **P** are of the same size.

[4]Since we use MLP in both GCN and Transformer, the terms $\mathbf{W}^{(j)}$ and $\mathbf{W}_V^{(j)}$ are unnecessary, but we put them to follow the original formulations.

Given $\mathbf{M}$ and $\mathbf{P}$, we want to guide GCT to recover the true graph structure as much as possible. But we also want to allow some room for GCT to learn novel connections that are helpful for solving given prediction tasks. Therefore GCT uses the following formulation:

$$\text{Define } \hat{\mathbf{A}}^{(j)} := \text{softmax}(\frac{\mathbf{Q}^{(j)}\mathbf{K}^{(j)\top}}{\sqrt{d}} + \mathbf{M}) \tag{3}$$

Self-attention:

$$\mathbf{C}^{(j)} = \text{MLP}^{(j)}\left(\mathbf{P}\mathbf{C}^{(j-1)}\mathbf{W}_V^{(j-1)}\right) \text{ when } j = 1, \quad \mathbf{C}^{(j)} = \text{MLP}^{(j)}\left(\hat{\mathbf{A}}^{(j)}\mathbf{C}^{(j-1)}\mathbf{W}_V^{(j)}\right) \text{ when } j > 1$$

Regularization:

$$L_{reg}^{(j)} = D_{KL}(\mathbf{P}\|\hat{\mathbf{A}}^{(j)}) \text{ when } j = 1, \quad L_{reg}^{(j)} = D_{KL}(\hat{\mathbf{A}}^{(j-1)}\|\hat{\mathbf{A}}^{(j)}) \text{ when } j > 1$$

$$L = L_{pred} + \lambda \sum_j L_{reg}^{(j)} \tag{4}$$

In preliminary experiments, we noticed that attentions were often uniformly distributed in the first block of Transformer. This seemed due to Transformer not knowing which connections were worth attending. Therefore we replace the attention mechanism in the first GCT block with the conditional probabilities $\mathbf{P}$. The following blocks use the masked self-attention mechanism. However, we do not want GCT to drastically deviate from the informative $\mathbf{P}$, but rather gradually improve upon $\mathbf{P}$. Therefore, based on the fact that attention is itself a probability distribution, and inspired by Trust Region Policy Optimization [24], we sequentially penalize attention of j-th block if it deviates too much from the attention of $j-1$-th block, using KL divergence. As shown by Eq. (4), the regularization terms are summed to the prediction loss term (*e.g.* negative log-likelihood), and the trade-off is controlled by the coefficient λ. GCT's code will be made publicly available in the future.

4 Experiments

4.1 Synthetic Encounter Record

Choi et al. [10] evaluated their model on proprietary EHR data that contained structure information. Unfortunately, to the best of our knowledge, there are no publicly available EHR data that contain structure information (which is the main motivation of this work). In order to evaluate GCT's ability to learn EHR structure, we instead generated synthetic data that has a similar structure as EHR data.

The synthetic data has the same *visit-diagnosis-treatment-lab results* hierarchy as EHR data, and was generated in a top-down fashion. Each level was generated conditioned on the previous level, where the probabilities were modeled with the Pareto distribution. Pareto distribution follows the power law which best captures the long-tailed nature of medical codes. Using 1000 diagnosis codes, 1000 treatment codes, and 1000 lab codes, we initialized $p(D)$, $p(D|D)$, $p(M|D)$, $p(R|M,D)$ to follow the Pareto distribution, where D, M, and R respectively denote diagnosis, treatment, and lab random variables. $p(D)$ is used to draw independent diagnosis codes d_i, and $p(D|D)$ is used to draw d_j that are likely to co-occur with the previously sampled d_i. $P(M|D)$ is used to draw a treatment code m_j, given some d_i. $P(R|M,D)$ is used to draw a lab code r_k, given some m_j and d_i. Detailed description of generating the synthetic records and the link to download them are provided in Appendix A, and Appendix E, respectively. Code for generating the synthetic records will be open-sourced in the future. Table 1 summarizes the data statistics.

4.2 eICU Collaborative Research Dataset

To test GCT on real-world EHR records, we use Philips eICU Collaborative Research Dataset[5] [25]. eICU consists of Intensive Care Unit (ICU) records filtered for remote caregivers, collected from multiple sites in the United States between 2014 and 2015. From the encounter records, medication orders and procedure orders, we extracted diagnosis codes and treatment codes (*i.e.* medication, procedure codes). Since the data were collected from an ICU, a single encounter can last several days, where the encounter structure evolves over time, rather than being fixed as Figure 1. Therefore we used encounters where the patient was admitted for less than 24 hours, and removed duplicate codes (*i.e.* medications administered multiple times). Additionally, we did not use lab results as their values change over time in the ICU setting (*i.e.* blood pH level). We leave as future work how to handle ICU records that evolve over a longer period of time. Note that eICU does not contain

[5] https://eicu-crd.mit.edu/about/eicu/

Table 1: Statistics of the synthetic dataset and eICU

	Synthetic	eICU
# of encounters	50,000	41,026
# of diagnosis codes	1,000	3,093
# of treatment codes	1,000	2,132
# of lab codes	1,000	N/A
Avg. # of diagnosis per visit	7.93	7.70
Avg. # of treatment per visit	14.59	5.03
Avg. # of lab per visit	21.31	N/A

structure information. For example, we know that *cough* and *acetaminophen* in Figure 1 occur in the same visit, but do not know if *acetaminophen* was prescribed due to *cough*. Table 1 summarizes the data statistics.

4.3 Baseline Models

- **GCN**: Given the true adjacency matrix $\mathbf{A}$, we follow Eq. (1) to learn the feature representations $\mathbf{c}_i$ of each feature c_i in a visit $\mathcal{V}$. The visit embedding $\mathbf{v}$ (*i.e.* graph-level representation) is obtained from the placeholder visit node v. This model will serve as the optimal model during the experiments.
- **GCN**$_P$: Instead of the true adjacency matrix $\mathbf{A}$, we use the conditional probability matrix $\mathbf{P}$, and follow Eq. (1).
- **GCN**$_{random}$: Instead of the true adjacency matrix $\mathbf{A}$, we use a randomly generated normalized adjacency matrix where each element is indepdently sampled from a uniform distribution between 0 and 1. This model will let us evaluate whether true encounter structure is useful at all.
- **Shallow**: Each c_i is converted to a latent representation $\mathbf{c}_i$ using multi-layer feedforward networks with ReLU activations. The visit representation $\mathbf{v}$ is obtained by simply summing all $\mathbf{c}_i$'s. We use layer normalization [26], drop-out [27] and residual connections [28] between layers.
- **Deep**: We use multiple feedforward layers with ReLU activations (including layer normalization, drop-out and residual connections) on top of **shallow** to increase the expressivity. Note that Zaheer et al. [29] theoretically describes that this model is sufficient to obtain the optimal representation of a set of items (*i.e.,* a visit consisting of multiple features).

4.4 Prediction Tasks

In order to evaluate the model's capacity to leverage the implicit encounter structure, we use prediction tasks based on a single encounter, rather than a sequence of encounters, which was the experiment setup in Choi et al. [10]. Specifically, we test the models on the following tasks. Parentheses indicate which dataset is used for each task.

- **Graph reconstruction (Synthetic)**: Given an encounter with N features, we train models to learn N feature embeddings $\mathbf{C}$, and predict whether there is an edge between every pair of features, by performing an inner-product between each feature embedding pairs $\mathbf{c}_i$ and $\mathbf{c}_j$ (*i.e.* N^2 binary predictions). We do not use **Deep** baseline for this task, as we need individual embeddings for all features c_i's.
- **Diagnosis-Treatment classification (Synthetic)**: We assign labels to an encounter if there are specific diagnosis (d_1 and d_2) and treatment code (m_1) connections. Specifically, we assign label "1" if the encounter contains d_1-m_1 connection, and label "2" if the encounter contains d_2-m_1 connection. We intentionally made the task difficult so that the models cannot achieve a perfect score by just basing their prediction on whether d_1, d_2 and m_1 exist in an encounter. The prevalence for both labels are approximately 3.3%. Further details on the labels are provided in Appendix B. This is a multi-label prediction task using the visit representation $\mathbf{v}$.
- **Masked diagnosis code prediction (Synthetic, eICU)**: Given an encounter record, we mask a random diagnosis code d_i. We train models to learn the embedding of the masked code to predict its identity, *i.e.* a multi-class prediction. For **Shallow** and **Deep**, we use the visit embedding $\mathbf{v}$ as a proxy for the masked code representation. The row and the column of the conditional probability matrix $\mathbf{P}$ that correspond to the masked diagnosis were also masked to zeroes.

Table 2: Graph reconstruction and diagnosis-treatment classification performance.

Model	Graph reconstruction				Diagnosis-Treatment classification			
	Validation AUCPR	Validation AUROC	Test AUCPR	Test AUROC	Validation AUCPR	Validation AUROC	Test AUCPR	Test AUROC
GCN	0.9999	1.0	0.9999	1.0	1.0	1.0	1.0	1.0
GCN_P	0.5863	0.8894	0.5826	0.8887	0.9164	0.9934	0.9175	0.9931
GCN_{random}	0.5657	0.8807	0.5614	0.8803	0.8973	0.9916	0.8900	0.9903
Shallow	0.5482	0.8589	0.5444	0.8582	0.9235	0.9935	0.9103	0.9926
Deep	-	-	-	-	0.9173	0.9936	0.9070	0.9923
Transformer	0.5944	0.8925	0.5885	0.8910	**0.9293**	**0.9942**	0.9146	0.9930
GCT	**0.6063**	**0.8966**	**0.6003**	**0.8956**	0.9291	**0.9942**	**0.9219**	**0.9933**

Table 3: Masked diagnosis code prediction performance on the two datasets.

Model	Synthetic		eICU	
	Validation Accuracy	Test Accuracy	Validation Accuracy	Test Accuracy
GCN	0.2828	0.3024	-	-
GCN_P	0.1908	0.2096	0.7510	0.7389
GCN_{random}	0.1780	0.1980	0.7184	0.7137
Shallow	0.2006	0.2156	0.7387	0.7241
Deep	0.1900	0.2048	0.7406	0.7343
Transformer	0.1962	0.2026	0.7295	0.7158
GCT	**0.2182**	**0.2328**	**0.7796**	**0.7644**

- **Readmission prediction (eICU)**: Given an encounter record, we train models to learn the visit embedding $\mathbf{v}$ to predict whether the patient will be admitted to the ICU again during the same hospital stay, *i.e.,* a binary prediction. The prevalence is approximately 17.2%.
- **Mortality prediction (eICU)**: Given an encounter record, we train models to learn the visit embedding $\mathbf{v}$ to predict patient death during the ICU admission, *i.e.,* a binary prediction. The prevalence is approximately 7.3%.

Note that the conditional probability matrix $\mathbf{P}$ was calculated only with the training set. Further training details and hyperparameter settings are described in Appendix C.

4.5 Prediction Performance

Table 2 shows the graph reconstruction performance and the diagnosis-treatment classification performance of all models. Naturally, GCN shows the best performance since it uses the true adajcency matrix $\mathbf{A}$. Given that GCN_P's performance is only marginally inferior to Transformer, we can infer that the conditional probability is indeed indicative of the true structure. GCT, which combines the strength of both GCN_P and Transformer shows the best performance, besides GCN. It is noteworthy that GCN_{random} outperforms Shallow. This seems to indicate that for graph reconstruction, attending to other features, regardless of how accurately the process follows the true structure, is better than individually embedding each feature. Diagnosis-treatment classification, on the other hand, clearly penalizes randomly attending to the features, since GCN_{random} shows the worst performance. GCT again shows the best performance.

Table 3 shows the model performance for masked diagnosis prediction for both datasets. GCN could not be evaluated on eICU, since eICU does not have the true structure. However, it naturally shows the best performance on the synthetic dataset. GCN_{random} again demonstrates the worst performance for both datasets, indicating that inferring a diagnosis based on other features requires accurate structure knowledge. Interestingly, the task performance is significantly higher for eICU than for the synthetic dataset. This is mainly due to eICU having a very skewed diagnosis code distribution. In eICU, more than 80% of encounters have diagnosis codes related to whether the patient has been in an operating room prior to the ICU admission. Therefore randomly masking one of them does not make the prediction task as difficult as for the synthetic dataset.

Table 4: Readmission prediction and mortality prediction performance on eICU.

Model	Readmission prediction				Mortality prediction			
	Validation AUCPR	Validation AUROC	Test AUCPR	Test AUROC	Validation AUCPR	Validation AUROC	Test AUCPR	Test AUROC
GCN_P	0.4952	0.7327	0.4706	**0.7422**	0.6664	0.9153	0.6082	0.9066
GCN_{random}	0.4740	0.6948	0.4421	0.6953	0.6592	0.9196	0.5996	0.9098
Shallow	0.3962	0.6852	0.3517	0.6740	0.6687	0.9206	0.6120	0.9120
Deep	0.4527	0.7170	0.4429	0.7403	0.6724	0.9213	0.6023	0.9110
Transformer	0.5057	0.7341	0.4571	0.7161	0.6594	0.9120	0.6039	0.9102
GCT	**0.5279**	**0.7345**	**0.4837**	0.7421	**0.6757**	**0.9209**	**0.6178**	**0.9122**

Table 5: KL divergence between the normalized true adjacency matrix and the attention map. We also show the entropy of the attention map to indicate the sparseness of the attention distribution.

Model	Graph Reconstruction		Diagnosis-Treatment Classification		Masked Diagnosis Code Prediction	
	KL Divergence	Entropy	KL Divergence	Entropy	KL Divergence	Entropy
GCN_P	8.2250	1.8761	8.2250	1.8761	8.2250	1.8761
Transformer	28.8123	1.6120	13.994	2.0953	15.1882	2.0230
GCT	7.6487	1.8788	8.5952	2.3143	9.0101	1.2618

Table 4 shows the readmission prediction and mortality prediction performance of all models on eICU. As shown by GCN_P and GCT's superior performance, it is evident that readmission prediction heavily benefits from using the latent encounter structure. Mortality prediction, on the other hand, seems to rely little on the encounter structure, as can be seen from the marginally superior performance of GCT. Rather, given the comparable performances of Shallow, it seems that mortality prediction can be successfully performed by simply adding all features. Even when the conditional probability is unnecessary, GCT still slightly outperforms other models, demonstrating its potential to be used as a general-purpose EHR modeling algorithm. These two experiments indicate that not all prediction tasks require the true encounter structure, and it is our future work to apply GCT to various prediction tasks to evaluate its effectiveness.

4.6 Evaluating the Learned Encounter Structure

In this section, we analyze the learned structure of both Transformer and GCT. As we know the true structure $\mathbf{A}$ of synthetic records, we can evaluate how well both models learned $\mathbf{A}$ via self-attention $\hat{\mathbf{A}}$. Since we can view the normalized true adjacency matrix $\tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}$ as a probability distribution, we can measure how well the attention map $\hat{\mathbf{A}}$ in Eq. (3) approximates $\tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}$ using KL divergence $D_{KL}(\tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}||\hat{\mathbf{A}})$. Table 5 shows the KL divergence between the normalized true adjacency and the learned attention on the test set of the synthetic data while performing three different tasks. For GCN_P, the adjacency matrix is fixed to the conditional probability matrix $\mathbf{P}$, so KL divergence can be readily calculated. For Transformer and GCT, we calculated KL divergence between $\tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}$ and the attention maps in each self-attention block, and averaged the results. Note that KL divergence can be lowered by evenly distributing the attention across all features, which is the opposite of learning the encounter structure. Therefore we also show the entropy of $\hat{\mathbf{A}}$ alongside the KL divergence.

As shown by Table 5, the conditional probabilities are closer to the true structure than what Transformer has learned, in all three tasks. GCT shows similar performance to GCN_P in all tasks, and was even able to improve upon $\mathbf{P}$ in the graph reconstruction task. Transformer showing strong performance in graph reconstruction, even with attentions significantly different from the true structure, again indicates the importance of just attending to other features in graph reconstruction, which was discussed in Section 4.5 regarding the performance of GCN_{random}. For the other two tasks, regularizing the models to stay close to $\mathbf{P}$ seems to help the models generalize better to the test set, especially in masked diagnosis code prediction. We show visual examples of attention behavior of both Transformer and GCT in Appendix D.

5 Conclusion

Learning effective patterns from raw EHR data is an essential step for improving the performance of many downstream prediction tasks. In this paper, we addressed the issue where the previous state-of-the-art method required the complete encounter structure information, and proposed GCT to

capture the underlying encounter structure when the structure information is unknown. Experiments demonstrated that GCT outperformed various baseline models on encounter-based tasks on both synthetic data and a publicly available EHR dataset, demonstrating its potential to serve as a general-purpose EHR modeling algorithm. In the future, we plan to apply GCT on patient-level tasks such as heart failure diagnosis prediction or unplanned emergency admission prediction, while working on improving the attention mechanism to learn more medically meaningful patterns.

References

[1] Zachary C Lipton, David C Kale, Charles Elkan, and Randall Wetzel. Learning to diagnose with lstm recurrent neural networks. *arXiv preprint arXiv:1511.03677*, 2015.

[2] Edward Choi, Mohammad Taha Bahadori, Andy Schuetz, Walter F Stewart, and Jimeng Sun. Doctor ai: Predicting clinical events via recurrent neural networks. In *Machine Learning for Healthcare Conference*, pages 301–318, 2016.

[3] Alvin Rajkomar, Eyal Oren, Kai Chen, Andrew M Dai, Nissan Hajaj, Michaela Hardt, Peter J Liu, Xiaobing Liu, Jake Marcus, Mimi Sun, et al. Scalable and accurate deep learning with electronic health records. *NPJ Digital Medicine*, 1(1):18, 2018.

[4] Zhengping Che, David Kale, Wenzhe Li, Mohammad Taha Bahadori, and Yan Liu. Deep computational phenotyping. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 507–516. ACM, 2015.

[5] Youngduck Choi, Chill Yi-I Chiu, and David Sontag. Learning low-dimensional representations of medical concepts. *AMIA Summits on Translational Science Proceedings*, 2016:41, 2016.

[6] Edward Choi, Mohammad Taha Bahadori, Elizabeth Searles, Catherine Coffey, Michael Thompson, James Bost, Javier Tejedor-Sojo, and Jimeng Sun. Multi-layer representation learning for medical concepts. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1495–1504. ACM, 2016.

[7] Riccardo Miotto, Li Li, Brian A Kidd, and Joel T Dudley. Deep patient: an unsupervised representation to predict the future of patients from the electronic health records. *Scientific reports*, 6:26094, 2016.

[8] Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter Stewart. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In *Advances in Neural Information Processing Systems*, pages 3504–3512, 2016.

[9] Fenglong Ma, Radha Chitta, Jing Zhou, Quanzeng You, Tong Sun, and Jing Gao. Dipole: Diagnosis prediction in healthcare via attention-based bidirectional recurrent neural networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1903–1911. ACM, 2017.

[10] Edward Choi, Cao Xiao, Walter Stewart, and Jimeng Sun. Mime: Multilevel medical embedding of electronic health records for predictive healthcare. In *NeurIPS*, pages 4552–4562, 2018.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.

[12] Harini Suresh, Nathan Hunt, Alistair Johnson, Leo Anthony Celi, Peter Szolovits, and Marzyeh Ghassemi. Clinical intervention prediction and understanding using deep networks. In *MLHC*, 2017.

[13] Phuoc Nguyen, Truyen Tran, and Svetha Venkatesh. Resset: A recurrent model for sequence of sets with applications to electronic medical records. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE, 2018.

[14] Truyen Tran, Tu Dinh Nguyen, Dinh Phung, and Svetha Venkatesh. Learning vector representation of medical objects via emr-driven nonnegative restricted boltzmann machines (enrbm). *Journal of Biomedical Informatics*, 2015.

[15] Edward Choi, Mohammad Taha Bahadori, Le Song, Walter F Stewart, and Jimeng Sun. Gram: Graph-based attention model for healthcare representation learning. In *SIGKDD*, 2017.

[16] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[18] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[19] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

[20] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv:1806.01261*, 2018.

[21] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.

[22] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

[23] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, pages 7794–7803, 2018.

[24] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.

[25] Tom J Pollard, Alistair EW Johnson, Jesse D Raffa, Leo A Celi, Roger G Mark, and Omar Badawi. The eicu collaborative research database, a freely available multi-center database for critical care research. *Scientific data*, 5, 2018.

[26] Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[27] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[29] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in neural information processing systems*, pages 3391–3401, 2017.

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

[31] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

Algorithm 1 Synthetic Encounter Records Generation

$|D|$ = 1000 // Dx vocab size
$|M|$ = 1000 // Treatment vocab size
$|R|$ = 1000 // Lab vocab size

// Independent diagnosis occurrence probability
$p(D)$ = permute(normalize(numpy.random.pareto(α = 2.0, size=|D|)))

// Conditional probability of a diagnosis co-occurring with another diagnosis
$p(D|d_1)$ = permute(normalize(numpy.random.pareto(α = 1.5, size=|D|)))
...
$p(D|d_{|D|})$ = permute(normalize(numpy.random.pareto(α = 1.5, size=|D|)))

// Conditional probability of a treatment being ordered for a specific diagnosis
$p(M|d_1)$ = permute(normalize(numpy.random.pareto(α = 1.5, size=|M|)))
...
$p(M|d_{|D|})$ = permute(normalize(numpy.random.pareto(α = 1.5, size=|M|)))

// Conditional probability of a lab being ordered for a specific treatment and diagnosis
$p(R|m_1, d_1)$ = permute(normalize(numpy.random.pareto(α = 1.5, size=|R|)))
...
$p(R|m_{|M|}, d_{|D|})$ = permute(normalize(numpy.random.pareto(α = 1.5, size=|R|)))

// Bernoulli probability to determine whether to sample a diagnosis code given a previous diagnosis. Values are clipped to [0., 1.).
$a(D)$ = numpy.random.normal(μ = 0.5, σ = 0.1, size=|D|))

// Bernoulli probability to determine whether to sample a treatment code given a diagnosis. Values are clipped to [0., 1.).
$b(D)$ = numpy.random.normal(μ = 0.5, σ = 0.25, size=|M|))

// Bernoulli probability to determine whether to sample a lab code given a treatment and a diagnosis. Values are clipped to [0., 1.).
$c(M, D)$ = numpy.random.normal(μ = 0.5, σ = 0.25, size=(|M|, |D|))

// Start creating synthetic records
repeat
  Sample a diagnosis d_i from $p(D)$
  repeat
    Sample a diagnosis d_j from $p(D|d_j)$
  until $x \sim Uniform(0, 1) < a(d_i)$
until $x \sim Uniform(0, 1) < 0.5$
for d_i in the sampled diagnosis codes **do**
  repeat
    Sample a treatment m_j from $P(M|d_i)$.
    repeat
      Sample a lab r_k from $P(R|m_j, d_i)$.
    until $x \sim Uniform(0, 1) < c(m_j, d_i)$
  until $x \sim Uniform(0, 1) < b(d_i)$
end for

A Generating Synthetic Encounter Records

We describe the synthetic data creation process in this section. As described in Section 4.1, we use the Pareto distribution to capture the long-tailed nature of medical codes. We also define $a(D)$, $b(D)$, and $c(M, D)$ to determine when to stop sampling the codes. The overall generation process starts by sampling a diagnosis code. Then we sample a diagnosis code that is likely to co-occur with the

previous sampled diagnosis code. After the diagnosis codes are sampled, we iterate through the sampled diagnosis code to sample a treatment code that is likely to be ordered for each diagnosis code. At the same time as sampling the treatment code, we sample lab codes that are likely to be produced by each treatment code. The overall algorithm is described in Algorithm 1.

Note that we use $p(M|D)$ to model the treatment being ordered due to a diagnosis code, instead of $P(M|M, D)$, which might be more accurate since a treatment may depend on the already ordered treatments as well. However, we assume that given a diagnosis code, treatments that follow are conditionally independent, therefore each treatment can be factorized by $p(M|D)$. The same assumption went into using $P(R|M, D)$, instead of $P(R|R, M, D)$.

Finally, among the generated synthetic encounters, we removed the ones that had less than 5 diagnosis or treatment codes, in order to make the encounter structure sufficiently complex. Additionally, we removed encounters which contained more than 50 diagnosis or treatment or lab codes in order to make the encounter structure realistic (*i.e.* it is unlikely that a patient receives more than 50 diagnosis codes in one hospital encounter). For the eICU dataset, we also removed the encounters with more than 50 diagnosis or treatment codes. But we did not remove any encounters for having less than 5 diagnosis or treatment codes, as that would leave us only approximately 7,000 encounter records, which are rather small for training neural networks.

B Diagnosis-Treatment Classification Task

This task is used to test the model's ability to derive a visit representation $\mathbf{v}$ (*i.e.* graph-level representation) that correctly reserves the encounter structure. As described in Section 4.4, this is a multi-label classification problem, where an encounter is assigned the label "1" if it contains a connected pair of a diagnosis code d_1 and a treatment code m_1 (*i.e.* m_1 was ordered because of d_1). An encounter is assigned the label "2" if it contains a connected pair of d_2 and m_1. Therefore it is possible that an encounter is assigned both labels "1" and "2", or not assigned any label at all.

Since we want to test the model's ability to correctly learn the encounter structure, we do not want the model to achieve a perfect score, for example, by just predicting label "1" based on whether both d_1 and m_1 simply exist in an encounter. Therefore we adjusted the sampling probabilities to make this task difficult. Specifically, we set $p(d_1) = 0.33, a(d_1) = 0.8, p(d_2|d_1) = 0.33, b(d_1) = 0.5, b(d_2) = 0.5, (p(m_1|d_1) = 0.2, p(m_1|d_2) = 0.8$. Therefore the probability of an encounter containing a d_1-m_1 connection is $p(d_1)b(d_1)p(m_1|d_1) = 0.33 \times 0.5 \times 0.2 = 0.033$. The probability of an encounter contaning a d_2-m_1 connection is $p(d_1)a(d_1)p(d_2|d_1)b(d_2)p(m_1|d_2) = 0.33 \times 0.8 \times 0.33 \times 0.5 \times 0.8 \approx 0.033$. Therefore The overall probability of the two connection pairs occurring in an encounter are more or less the same, and the model cannot achieve a perfect score unless the model correctly identifies the encounter structure.

C Training Details

We divided both the synthetic dataset and eICU into a training set, a validation set, and a test set in 8:1:1 ratio. All models were trained with Adam [30] on the training set, and performance was evaluated against the validation set to select the final model. Final performance was evaluated against the test set. We used the minibatch of size 32, and trained all models for 200,000 iterations (*i.e.* minibatch updates), which was sufficient for convergence for all tasks. After an initial round of preliminary experiments, the embedding size of the encounter features was set to 128. For GCN, GCN_P, GCN_{random}, Transformer, and GCT, we used undirected adjacency/attention matrix to enhance the message passing efficiency. All models were implemented in TensorFlow 1.13 [31], and trained with a system equipped Nvidia P100's.

Tunable hyperparameters for models **Shallow**, **Deep**, **GCN**, $\mathbf{GCN}_P$, $\mathbf{GCN}_{random}$, and **Transformer** are as follows:

- Adam learning rate ($0.0001 \sim 0.1$)
- Drop-out rate between layers ($0.0 \sim 0.9$)

Shallow used 15 feedforward layers and **Deep** used 8 feedforward layers before, and 7 feedforward layers after summing the embeddings. The number of layers were chosen to match the number

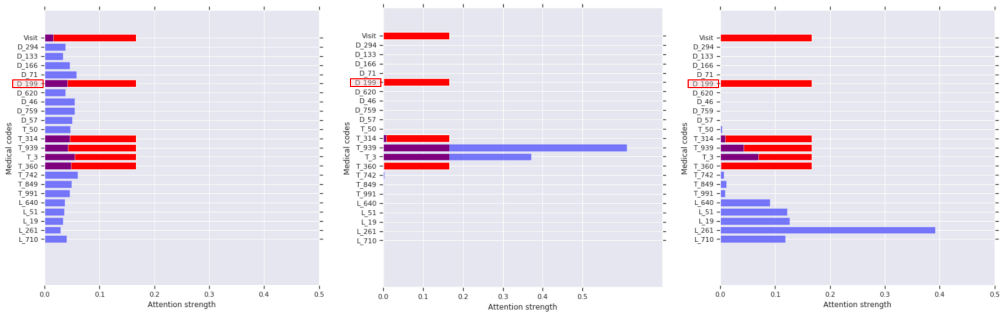
of trainable parameters of Transformer and GCT. GCN, GCN_P, GCN_{random} and Transformer three self-attention blocks, which was sufficient to cover the entire depth of EHR encounters. Transformer used one attention head to match its representative power to GCN, GCN_P, and GCN_{random}, and so that we can accurately evaluate the effect of learning the correct encounter structure.

Tunable hyperparameters for **GCT** are as follows:

- Adam learning rate (0.0001 ~ 0.1)
- Drop-out rate between layers (0.0 ~ 0.9)
- Regularization coefficient (0.01 ~ 100.0)

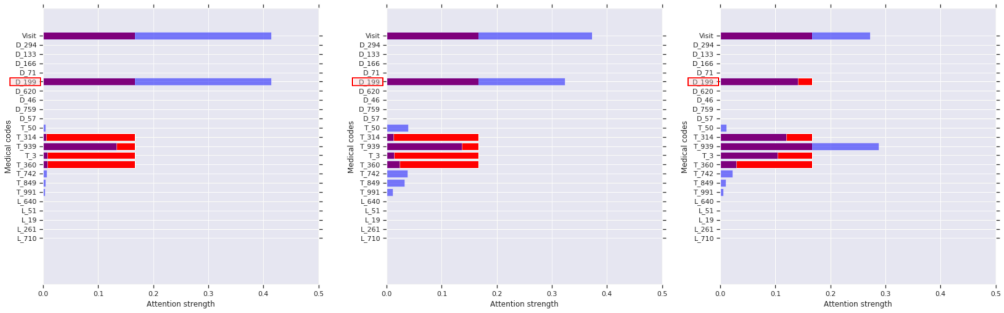
GCT also used three self-attention blocks and one attention head. All Hyperparameters were searched via bayesian optimization with Gaussian Process for 48-hour wall clock time.

D Attention Behavior



Self-attention Block 1 Self-attention Block 2 Self-attention Block 3

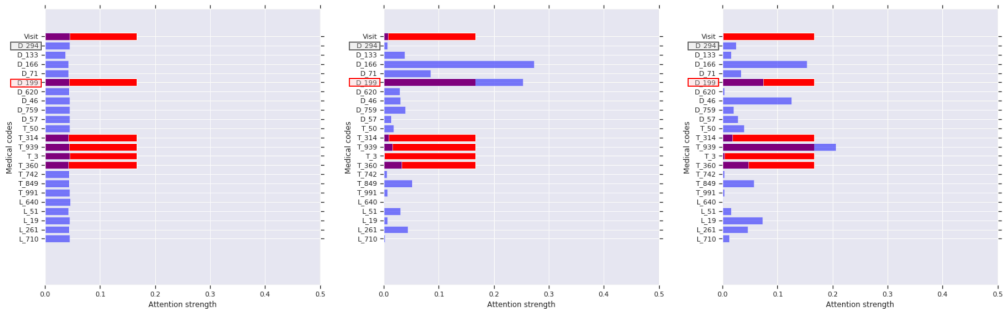
Figure 4: Attentions from each self-attention block of Transformer trained for graph reconstruction. Code starting with 'D' are diagnosis codes, 'T' treatment codes, 'L' lab codes. The diagnosis code with the red background D_199 is attending to the other features. The red bars indicate the codes that are actually connected to D_199, and the blue bars indicate the attention given to all codes.



Self-attention Block 1 Self-attention Block 2 Self-attention Block 3

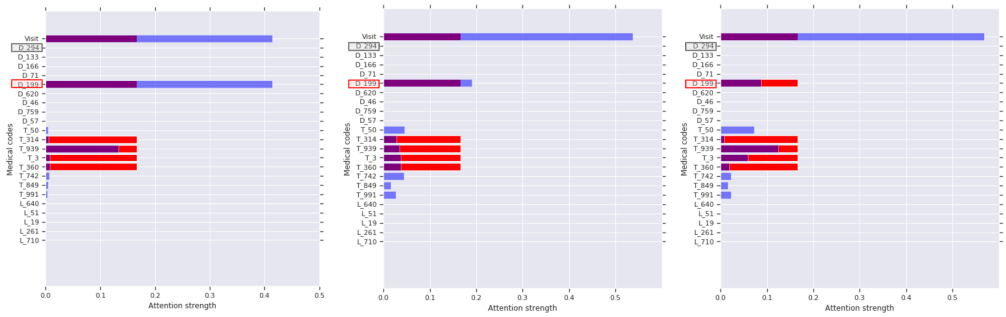
Figure 5: Attentions from each self-attention block of GCT trained for graph reconstruction. Code starting with 'D' are diagnosis codes, 'T' treatment codes, 'L' lab codes. The diagnosis code with the red background D_199 is attending to the other features. The red bars indicate the codes that are actually connected to D_199, and the blue bars indicate the attention given to all codes.

In this section, we compare the attention behavior of Transformer and GCT in two different context; graph reconstruction and masked diagnosis code prediction. We randomly chose an encounter record from the test set of the synthetic dataset, which had less than 30 codes in order to enhance readability. To show the attention distribution of a specific code, we chose the first diagnosis code connected to at least one treatment. Figure 4 shows Transformer's attentions in each self-attention block when performing graph reconstruction. Specifically we show the attention given by the diagnosis code D_199 to other codes. The red bars indicate the true connections, and the blue bars indicate the



Self-attention Block 1 Self-attention Block 2 Self-attention Block 3

Figure 6: Attentions from each self-attention block of Transformer trained for masked diagnosis code prediction. Code starting with 'D' are diagnosis codes, 'T' treatment codes, 'L' lab codes. The diagnosis code with the red background D_199 is attending to the other features. The diagnosis code with the gray background D_294 is the masked diagnosis code. The red bars indicate the codes that are actually connected to D_199, and the blue bars indicate the attention given to all codes.



Self-attention Block 1 Self-attention Block 2 Self-attention Block 3

Figure 7: Attentions from each self-attention block of GCT trained for masked diagnosis code prediction. Code starting with 'D' are diagnosis codes, 'T' treatment codes, 'L' lab codes. The diagnosis code with the red background D_199 is attending to the other features. The diagnosis code with the gray background D_294 is the masked diagnosis code. The red bars indicate the codes that are actually connected to D_199, and the blue bars indicate the attention given to all codes.

attention given to all codes. It can be seen that Transformer evenly attends to all codes in the first block, then develops its own attention. In the second block, it successfully recovers two of the true connections, but attends to incorrect codes in the third block.

Figure 5 shows GCT's attention in each self-attention blcok when performing graph reconstruction. Contrary to Transformer, GCT starts with a very specific attention distribution. The first two attentions given to the placeholder Visit node, and to itself are determined by the scalar value from Figure 3. However, the attentions given to the treatment codes, especially T_939 are derived from the conditional probability matrix **P**. Then in the following self-attention blocks, GCT starts to deviate from **P**, and the attention distribution becomes more similar to the true adjacency matrix. This nicely shows the benefit of using **P** as a guide to learning the encounter structure.

Since the goal of the graph reconstruction task is to predict the edges between nodes, it may be an obvious result that both Transformer and GCT's attentions mimic the true adjacency matrix. Therefore, we show another set of attentions from Transformer and GCT trained for the masked diagnosis code prediction task. Figure 6 shows Transformer's attention while performing the masked diagnosis code prediction. Note that the diagnosis code D_294 is maksed, and therefore the model does not know its identity. Similar to graph reconstruction, Transformer starts with an evenly distributed attentions, and develops its own structure. Interestingly, it learns to attend to the right treatment in the third block, but mostly tries to predict the masked node's identity by attending to other diagnosis codes, while mostly ignoring the lab codes.

Figure 7 shows GCT's attention while performing the masked diagnosis code prediction task. Again, GCT starts with the conditional probability matrix **P**, then develops its own attention. But this time, understandably, the attention maps are not as similar to the true structure as in the graph reconstruction task. An interesting finding is that GCT attends heavily to the placeholder Visit node in this task. This is inevitable, given that we only allow diagnosis codes to attend to treatment codes (see the white cells in Figure 3), and therefore, if GCT wants to look at other diagnosis codes, it can only be done by indirectly receiving information via the Visit node. And as Figure 6 suggests, predicting the identity of the masked code seems to require knowing the co-occurring diagnosis codes as well as the treatment codes. Therefore, unlike in the graph reconstruction task, GCT puts heavy attention to the Visit node in this task, in order to learn the co-occurring diagnosis codes.

E Sharing the Synthetic Records

The synthetic records used for the experiments can be downloaded via this link (https://www.dropbox.com/s/ojx9jr4yyvmfdum/synthetic.tar.gz). It is a compressed file, which you can decompress to obtain the following files.

- **visits_50k.p**: This is a Python cPickle file. It is a List of encounter records, where each record is a List of a diagnosis code and the associated treatment-lab Lists. For example, [[1, []], [2, [[3, []], [4, [5, 6]]]]] describes a single encounter record. The first diagnosis code is "1", and no treatment or lab codes follow. The second diagnosis code is "2", and the treatment "3", and treatment "4" are ordered because of the diagnosis "2". Additionally, treatment "4" is followed by two lab codes, "5" and "6". **visits_50k.p** consists of 500,000 encounter records that follow this format.
- **dx_probs.npy**: This Python Numpy file corresponds to $p(D)$ in Algorithm 1. It is a 1000-dimensional vector, where the i-th element represents $p(d_i)$.
- **dx_dx_cond_probs.npy**: This Python Numpy file corresponds to $p(D|D)$ in Algorithm 1. It is a 1000-by-1000 matrix, where the (i, j)-th element represents $p(d_j|d_i)$.
- **dx_proc_cond_probs.npy**: This Python Numpy file corresponds to $p(M|D)$ in Algorithm 1. It is a 1000-by-1000 matrix, where the (i, j)-th element represents $p(m_j|d_i)$.
- **dx_proc_lab_cond_probs.npy**: This Python Numpy file corresponds to $p(R|M, D)$ in Algorithm 1. It is a 1000-by-1000-by-1000 3D tensor, where the (i, j, k)-th element represents $p(r_k|m_j, d_i)$.
- **dx_dx_probs.npy**: This Python Numpy file corresponds to $a(D)$ in Algorithm 1. It is a 1000-dimensional vector, where the i-th element represents $a(d_i)$.
- **multi_proc_probs.npy**: This Python Numpy file corresponds to $b(D)$ in Algorithm 1. It is a 1000-dimensional vector, where the i-th element represents $b(d_i)$.
- **multi_lab_probs.npy**: This Python Numpy file corresponds to $c(M, D)$ in Algorithm 1. It is a 1000-by-1000 matrix, where the (i, j)-th element represents $c(m_j, d_i)$.

Note that d_1, d_2 and m_1 discussed in Appendix B correspond to the actual codes 199, 133 and 939, respectively. Therefore, for example, the (199, 939)-th element in **dx_proc_cond_probs.npy** equals approximately 0.2, and the (133, 939)-the element equals approximately 0.8.