Reproducibility Project for CS598 DL4H in Spring 2023

Yu-Chung Lee and Hyeonjae Cho
{ycl7, hc53}@illinois.edu

Group ID: 30
Paper ID: 6
Presentation link: https://youtu.be/qMEwlSeE_vk
Code link: https://github.com/lycpaul/dl4h-gp30-gct

1 Introduction

1.1 Citation

The paper (Choi et al., 2020), published by Google DeepMind, demonstrates that the graph convolutional transformer can handle incomplete or zero structure information of Electronic Health Records data.

1.2 General Problem

EHR data have been shown to have an underlying graphical structure that helps improve the performance of the prediction task. However, the previous models treated EHR data in an unordered feature set, disregarding the relationships between diagnoses and treatments. MiME (Choi et al., 2018) proposed a model that learns the structural information of the EHR data but is still insufficient because many publicly accessible datasets do not provide the reasoning to establish graphic connections, such as claim data. To utilize the hidden hierarchical structure, a more generalized method is needed.

1.3 The Approach

The author proposed a GCT network to learn the unknown hierarchical structure of the EHR data by using the self-attention mechanism to obtain a model that allows better classification or prediction performance compared to other baselines that do not consider the hidden graphical structure (Vaswani et al., 2017). In addition, they used prior knowledge as an attention mask. We already know that some treatments are necessary for certain diseases, and others are unavailable. Additionally, we can calculate the conditional probabilities among diagnoses, treatments, and laboratory tests based on examples of encounter records. The prior information, illustrated in Figure 1, replaces the first GCT block attention layer, which is generally uniformly distributed. As the model goes through further

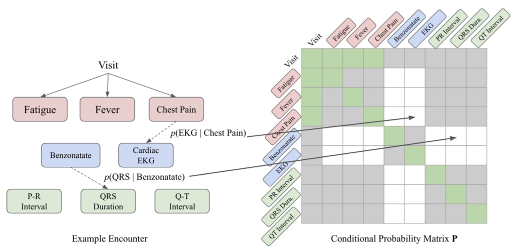

Figure 1: The grey areas indicate not-allowed connections. The green cells show special connections. Conditional probabilities are assigned in corresponding white cells. (Choi et al., 2020)

depth blocks, it penalizes if the attention value of the i-th block is quite different from the one of 1st block. Therefore, attention values can imply both a realistic and a novel relationship between nodes.

2 Scope of reproducibility

The author claims that the GCT model is superior for prediction tasks in a single encounter. Among several experiments conducted in the paper, we plan to reproduce the readmission and mortality prediction task in the eICU dataset and compare it to GCN, which the original paper proposes as benchmarking downstream prediction tasks. We chose the eICU dataset over the synthetic encounter record because the eICU dataset represents real-world EHR data and is considered a close representation of the realistic distribution of diagnostic codes. Therefore, we will verify the claim written in the sub-section below.

We will also evaluate two more baseline models for the prediction tasks: GCN_p, which is the graph convolution network directly trained based on the conditional probability of preprocessed features, and Transformer, which is the original Transformer implementation with conditional probability as a prior guide, guided mask, and regularization.

2.1 Addressed claims from the original paper

In the required downstream prediction task, GCN_p can utilize the conditional probability as a graph connection to assist the prediction, but such interpretation is noisy and inaccurate. While Transformer can learn the hidden connection, it may struggle to converge without prior guidance. Given that GCT is proposed to train the hidden reasoning connection between diagnostic codes, treatment, and laboratory results with prior knowledge as an initial guide, GCT should combine the strength of GCN_p and Transformer to deliver a better result. Therefore, two imperial results should be observed:

- The graph-convolutional transformer will have AUROC/AUCPR higher than a GCN_p and Transformer on readmission prediction.
- The graph-convolutional transformer will have AUROC/AUCPR higher than a GCN_p and Transformer on mortality prediction.

3 Methodology

We first adopted some of the author's code, publicly available on the Google Health GitHub repository[1]. We made some minor modifications to make the code compatible with the latest versions of Python 3 and TensorFlow 2, which offer a more stable environment. We also referenced another PyTorch reimplementation of the GCT model [2] and implemented the preprocessing and training pipeline with the PyHealth framework (Yang et al., 2022). The baseline training can be done quickly with the processing pipeline and PyHealth pre-defined models. Descriptive notebooks examples describe the complete training to the evaluation process.

3.1 Model descriptions

The proposed model, Graph Convolutional Transformer, combines the concept from the Transformer (Vaswani et al., 2017) and Graph Convolutional Networks (Kipf and Welling, 2016), as the name implies. Especially the Transformer referred to here is an encoder with single-head attention. In addition, GCT does not involve positional embeddings since features in a single encounter are not ordered.

[1]https://github.com/Google-Health/records-research/tree/master/graph-convolutional-transformer
[2]https://github.com/dchang56/gct-pytorch

Graph Convolutional Networks (GCN) formulation with known structure information $\mathbf{A}$

$$\mathbf{C}^{(j)} = \text{MLP}^{(j)}\left(\tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}\mathbf{C}^{(j-1)}\mathbf{W}^{(j)}\right)$$

The intuition of the model construction is that: since the actual graph connections are unknown, instead of training the model with the normalized adjacency matrix $\tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}$, we replaced that with the attention map softmax($\frac{\mathbf{Q}^{(j)}\mathbf{K}^{(j)T}}{\sqrt{d}}$) to learn the underlying connections. In addition, the GCT possesses two known graphical characteristics: First, it uses a mask in the attention generation procedure, which gives negative infinities if the graph connection is not allowed. Second, the regularization formula's loss function restricts the model from inventing unrealistic graph connections. It helps the model discover an underlying graph structure of EHR data in a realistic search space. A simplified model architecture is given in Figure 2. Moreover, the learning objective is given below:

$$\begin{aligned}\text{Define } \hat{\mathbf{A}}^{(j)} &:= \text{softmax}(\frac{\mathbf{Q}^{(j)}\mathbf{K}^{(j)T}}{\sqrt{d}} + \mathbf{M})\\ \text{where } \mathbf{Q}^{(j)} &= \mathbf{C}^{(j-1)}\mathbf{W}_Q^{(j)},\\ \mathbf{K}^{(j)} &= \mathbf{C}^{(j-1)}\mathbf{W}_K^{(j)},\\ \text{d = column size of } \mathbf{W}_K^{(j)}\end{aligned}$$

Self-attention:

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

Regularization:

$$L_{\text{reg}}^{(j)} = D_{KL}(\mathbf{P}||\hat{\mathbf{A}}^{(j)}) \text{ when } j = 1,$$
$$L_{\text{reg}}^{(j)} = D_{KL}(\hat{\mathbf{A}}^{(j-1)}||\hat{\mathbf{A}}^{(j)}) \text{ when } j > 1$$
$$L = L_{pred} + \lambda \sum_j \text{L}_{reg}^{(j)}$$

The above formula contains the observation of the j-th convolution layer in a model. $\mathbf{A}$ is an adjacency matrix for EHR data, which is an unknown or incomplete structure of an encounter. $\mathbf{Q}$ and $\mathbf{K}$ represent Query and Value, respectively, which are common vector notations by the attention mechanism. In the self-attention formula, $\mathbf{C}$ represents a node embedding, and $\mathbf{V}$ stands for a visit. MLP is a multi-layer perceptron located between convolution layers and has its own trainable parameters.

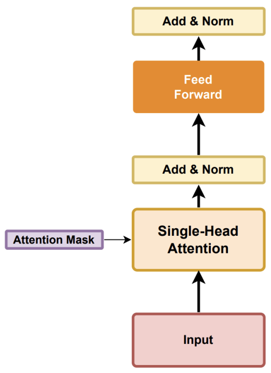

Figure 2: Architecture of Graph Convolutional Transformer.

W is the trainable parameter of the *j*-th convolution layer. Additionally, GCT proposes two unique techniques, a mask (**M**) and conditional probabilities (**P**).

3.2 Data descriptions

The eICU dataset includes publicly available[3] de-identified EHR data of over 200,000 ICU admissions from more than 200 hospitals across the United States (Johnson et al., 2017),. The dataset contains clinical data such as laboratory results, treatments, diagnoses, and demographic information about the patient, such as age, sex, and ethnicity. To download the dataset, the CITI training program is required.

The eICU dataset contains single encounters that may span over several days. However, the GCT model does not consider time variance, so we only considered a single encounter per patient. The preprocess.py script includes only encounters of less than 24 hours and eliminates duplicated medical codes, such as medications administered multiple times. Moreover, it excludes lab results since they are subject to change over time in the ICU.

The basic statistics of the processed data that were actually used in the model training are as follows:

3.3 Hyperparameters

We compiled all information regarding hyperparameter settings into a Table 2. In addition to the hyperparameters setting provided by the original paper, we also tried different model configurations. We adjusted the parameters to experiment with particular settings' potential effects and improvements,

[3]https://eicu-crd.mit.edu/gettingstarted/access/

	eICU
# of encounters	41,026
# of diagnosis codes	3,093
# of treatment codes	2,132
Avg. # of diagnosis per visit	7.70
Avg. # of treatment per visit	5.03

Table 1: Statistics of eICU Dataset

which will be further discussed in the results section.

	Readmission	Mortality
Batch size	32	32
MLP dropout rate	0.08	0.72
Post-MLP dropout rate	0.024	0.005
Learning rate	0.00022	0.00011
Regularization coef.	0.1	1.5
Embedding dim.	128	128
# of Transformer stacks	3	3
# of attention heads	1	1

Table 2: Training Hyperparameters for Readmission and Mortality Prediction on eICU as recommended in the original paper.

3.4 Implementation

The modified Tensorflow 2.0 version of the original source code is available in [4] and the modified PyTorch implementation referenced from another open source implementation is available here [5]. Additionally, we also integrate the PyHealth package, creating a new PyTorch training code in [6]. Furthermore, we have included auxiliary functions, such as attention visualization codes. However, this section will primarily focus on the detailed process of re-implementing and integrating the PyTorch version code into PyHealth. Please refer to Section 4.2 for visualization results.

The PyHealth modules provide a five-stage pipeline modality for all healthcare tasks: loading dataset, prediction tasks labeling, building model, training model, and model inference. By utilizing the datasets parser, and the pipeline design, we have minic a similar training procedure to train our GCT model. We have submitted the changes required

[4]https://github.com/lycpaul/dl4h-gp30-gct/tree/main/gct-tf2

[5]https://github.com/lycpaul/dl4h-gp30-gct/tree/main/gct-pytorch

[6]https://github.com/lycpaul/dl4h-gp30-gct/tree/main/gct-pyhealth

to parse the additional tables in eICU datasets to PyHealth (refer to section 5.5 for details) such that it also allows us to utilize the provided PyHealth AI models as baseline training. All models were trained with Adammax Kingma and Ba (2015) as the optimizer with linear scheduler instead of the Adam as proposed in the original paper to improve the performance.

3.5 Computational requirements

According to the original paper (Choi et al., 2020), both the GCT and the baseline used a system with an Nvidia Tesla P100 with a maximum of 16GB of VRAM. Alternatively, our group reproduced the result with an Nvidia RTX 3090Ti GPU 24GB VRAM system capable of performing the training under the same hyperparameters and settings. In our experiments, GCT implemented with TensorFlow2 will occupy 24GB of VRAM and takes around 8 hours to finish training 1M iterations. Meanwhile, another reimplementation of the GCT model using PyTorch requires less than 3GB VRAM with about 3 hours of training for 1,000,000 iterations.

4 Results

We have trained the GCT with eICU data on readmission and mortality prediction using the hyperparameters provided by the original paper and several other learning rate and dropout rate setting to see the prediction performance. We have also evaluated Transformer and GCN as the baseline model for mortality prediction.

4.1 Results

According to our preliminary result, we noticed that the AUCPR/AUCROC of both predictions of our trained model performed worse than the ones in the paper, even with the hyperparameters stated in the paper. Specifically, the prediction performance is shown in Table 8.

We have already trained the Transformer and GCN against the mortality tasks, and the performance is worse than that of GCT. We also noticed that changing the hyperparameters, such as learning rate, dropout rate, and batch size, would increase the performance.

Table 3: Tensorboard plot of our training experiment over the GCT model. The original files and results of other hyperparameters can be viewed in our public repository.

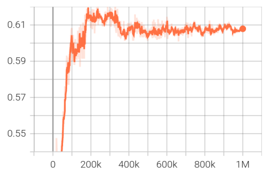

Table 4: AUCPR of mortality prediction.

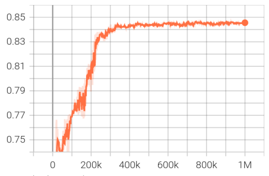

Table 5: AUROC of mortality prediction.

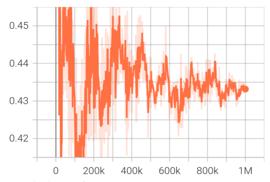

Table 6: AUCPR of readmission prediction.

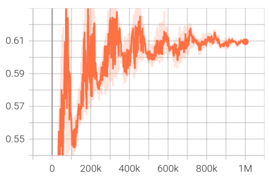

Table 7: AUROC of readmission prediction.

4.2 Additional results not present in the original paper

We first evaluate the choice of optimizer and scheduler for the training. We tested with Adam as proposed by the original paper and Adammax, which is a variant of Adam based on the infinity norm. Table 9 shows the improvements in choosing the Adammax with the linear scheduler. Therefore all of our remaining training experiments are conducted with it.

We investigated different post-MLP dropout rates in the output feedforward layer. In this experiment, we freeze other hyperparameters and choose only to evaluate the mortality prediction task. In table 9, we observed the higher dropout improved the prediction slightly. We also evaluate the effect of stacking self-attention blocks and the usage of multi-head attention. Table 9 indicates that 3 self-attention blocks did improve the performance, which also aligned with the original paper observations that the graphical connections improved and converged better in the third attention block. The multi-head attention also helps improve prediction accuracy.

We have implemented a visualization function that facilitates the comprehension of the attention behavior of the transformer block. According to the paper, GCT utilizes prior information power to predict the patient's outcome better. Therefore, we need to investigate attention maps further to confirm the accuracy of the claim. We have produced a heatmap and a network graph displaying the visit label, actual diagnoses, and treatments. During the processing procedure, the diagnoses and treatment data are concatenated with one visit tensor, resulting in a tensor with a shape of (`batch_size`,

Tasks	Validation AUCPR	Test AUCPR	Validation AUROC	Test AUROC
Readmission (Ours-GCT)	0.4329	0.4081	0.6094	0.6088
Readmission (Paper-GCT)	0.5313	0.5244	0.7525	0.7502
Mortality (Ours-GCT)	0.6079	0.5931	0.8455	0.8118
Mortality (Paper-GCT)	0.6196	0.5992	0.9089	0.9120
Mortality (Ours-Transformer)	0.6105	0.5585	0.8360	0.7847
Mortality (Ours-GCN)	0.4487	0.4523	0.6735	0.6612

Table 8: Readmission and mortality prediction tasks performance on eICU dataset.

(a) Optimizer, Scheduler & learning rate	Validation AUCPR	Test AUCPR
Adam, none, 1.1e-4	0.5340	0.5312
Adammax, linear, 1.1e-4	0.5658	**0.5974**
Adammax, linear, 1e-3	**0.5803**	0.5844
(b) Post-MLP dropout rate	Validation AUCPR	Test AUCPR
0.005	0.5658	0.5974
0.2	0.5624	0.6042
0.5	0.5655	**0.6061**
0.72	**0.5662**	0.5971
(c) # of Attention blocks & attention heads	Validation AUCPR	Test AUCPR
2 blocks 1 heads	0.5624	0.5842
2 blocks 2 heads	0.5601	0.6102
3 blocks 1 heads	0.5604	0.6042
3 blocks 2 heads	**0.5759**	**0.6191**

Table 9: (a) Training results of different optimizers, scheduler, and learning rate combination in mortality prediction. (b) Training results of different post-MLP dropout rates in mortality prediction. (c) Training results of different multi-head and number of Transformer stacks combination in mortality prediction. The learning rate is fixed at 1.1e-4 with post-MLP dropout rate at 0.2.

`2*max_num_codes+1)`. For compact visualization, we have dropped the padded diagnoses and treatments. Specifically, the heatmap demonstrates which diagnoses led the patient to visit and what treatments were administered. The graph network exhibits thicker edges when the attention value is higher. One can easily understand the relationship between components through the graph network at a glance. For a more detailed interpretation of the graphs, please refer to Figure 3 and 4.

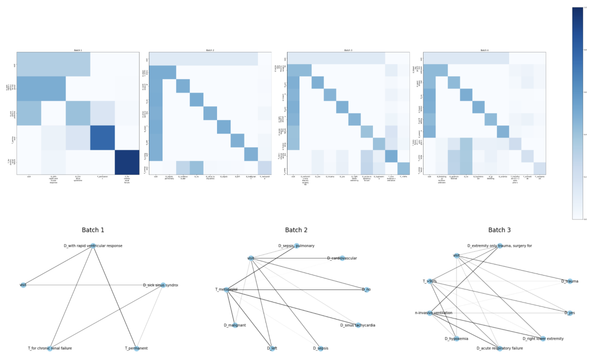

Figure 3: Visualization results in the GCT transformer block. Treatments and diagnoses are denoted with the prefixes 'T' and 'D', respectively. The full name of each component could not be displayed due to the length limit. (Note: This chart was plotted in one of the early training batches)

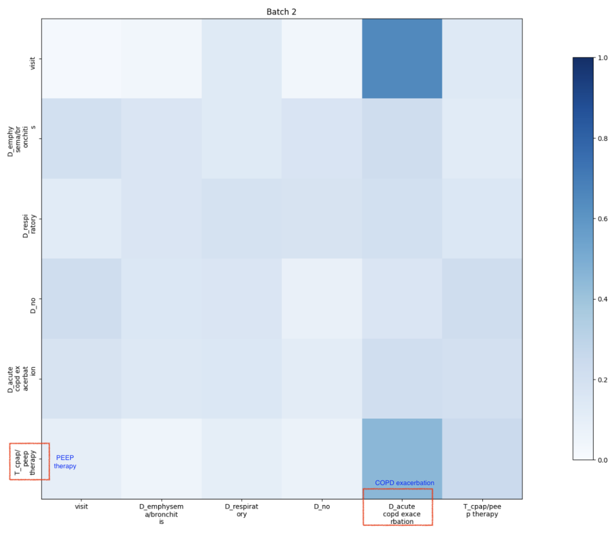

Figure 4: The heatmap demonstrates that the attention mechanism has learned meaningful relationships between the components. According to the chart, during this encounter, the primary reason for the visit was COPD exacerbation, and PEEP therapy was found to be the most closely associated treatment for the patient.

5 Discussion

Based on our training results, we evaluate the prediction results of GCT and the effects of different hyperparameters and model configurations. We also conducted some baseline comparisons with the Transformer, GCN, and reduced the number of Transformer stacks to evaluate whether the GCT model is aligned with the stated claims. In general, with our training settings, GCT performed consistently better than these baselines, and the visualization results also indicate that the Self-Attention blocks are capable of capturing the structure connections, thus providing a better visit representation for the specified prediction tasks. However, we cannot reproduce the exact prediction performance as stated in the original paper. Some other researchers attempting to replicate the model also faced a similar issue, and one potential issue of this is due to some unsolved bug in the open-sourced eICU pre-processing script [7] [8].

5.1 What was easy

We could take advantage of various online resources, which helped us understand the structure of GCT and its implementation more efficiently. Additionally, we also found a PyTorch version of the model on a public GitHub repository. Therefore, it was relatively easy to get started reproducing the original paper.

5.2 What was difficult

During the model experiments, we found that training and testing the model took a minimum of 6 or 7 hours, which made it difficult to change the parameters and observe changes in performance.

Additionally, we encountered some challenges when attempting to integrate GCT into PyHealth. Firstly, we had to modify the training data format to compute the prior probability. This was difficult because the preprocessing pipeline differed from other standard PyHealth models. Secondly, like other PyHealth models, GCT needs to handle the required input parameter, `mode`. However, the model currently produces a probabilities dimension based on the specified number of labels in the argument values of `num_labels`. Lastly, the calculation process of prior information requires the specification of vocabulary sizes to shape the tensor in the same size. Conversely, in PyHealth model modules, data statistics information is not included, and users are not required to know or specify this information as input parameters.

To make the GCT compatible with PyHealth data pipeline frameworks, we have modified the PyHealth library (details refer to 5.5) and include an additional preprocessing pipeline on top of the eICUDataset parser. We also created a trainer that is similar to the one in PyHealth model and reimplemented the GCT forward function such that it is compatible with our designed pipeline.

5.3 Recommendations for reproducibility

To fully utilize the available information in the eICU dataset, we recommend dedicating sufficient time to explore it. The proposed model heavily relies on the given data to extract valuable insights. Specifically, the model leverages masks and conditional probability, constructed by processing the eICU data, to provide prior information to the model. This plays a vital role in the model prediction of graph connections with higher accuracy.

5.4 Descriptive Notebooks

We have also included descriptive notebooks in the GitHub repository[9] to help with understanding the overall code flow. Another set of notebooks provides a step-by-step walkthrough of the data preprocessing, model training, and evaluation process [10]. They also include visualizations of the model's attention maps.

5.5 PyHealth Contribution

Three main modifications have been made in the PyHealth library and merged to the "master" branch recently[11].

- Updated `parse_diagnosis()` for parsing the additional diagnosisString columns required in this paper.
- Added `parse_admissiondx()` to parse admissionDX table in eICUDataset of the PyHealth Datasets modules.
- New prediction tasks are also included to parse the required feature key sets, namely `mortality_prediction_eicu_fn2()` and `readmission_prediction_eicu_fn2()`.

[7]https://github.com/Google-Health/records-research/issues/6

[8]https://github.com/Google-Health/records-research/issues/13

[9]https://github.com/lycpaul/dl4h-gp30-gct/tree/main/gct-pytorch/notebooks

[10]https://github.com/lycpaul/dl4h-gp30-gct/tree/main/gct-pyhealth

[11]https://github.com/sunlabuiuc/PyHealth/pull/148

References

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

Edward Choi, Zhen Xu, Yujia Li, Michael Dusenberry, Gerardo Flores, Emily Xue, and Andrew M. Dai. 2020. Learning the graphical structure of electronic health records with graph convolutional transformer. pages 606–613.

Alistair E. W. Johnson, Tom J. Pollard, Leo A. Celi, and Roger G. Mark. 2017. Analyzing the eicu collaborative research database. page 631.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization.

Thomas N. Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. pages 5998–6008.

Chaoqi Yang, Zhenbang Wu, Patrick Jiang, Zhen Lin, and Jimeng Sun. 2022. PyHealth: A deep learning toolkit for healthcare predictive modeling.