# Variationally Regularized Graph-based Representation Learning for Electronic Health Records

Camilo Velasquez & Aleksandr Rogachev
BDH Reproducibility Challenge: Final Report
GATECH - CSE 6250: Big Data for Health Informatics
Presentation: https://www.youtube.com/watch?v=Db0nZbbvb3k
Github: https://github.gatech.edu/arogachev3/vgnn-reproduce

## 1. Introduction

Electronic Health Records (EHR) are rich sources of information useful for predictive tasks such as mortality prediction, outcome prediction and phenotyping. The tasks that the paper used to test their method were readmission prediction using the eICU dataset, mortality prediction at 24 hours after admission using the MIMIC III dataset, and Alzheimer Disease prediction using an NYU Langone dataset. The main contributions of the paper were:

- The proposed model is robust enough to obtain state-of-the-art performance on multiple tasks.
- Introduction of variational regularization [3] added into a Graph Encoder Decoder Architecture for node representation. It enhances the learning of attention weights by preventing mode collapse and retaining graph representation expressiveness.
- Analysis and interpretation on the effect of variational regularization using Singular Value Analysis and clustering representation.

Our intention in this report is to replicate their method on the public datasets (eICU and MIMIC III), validate the variational regularization improvement and how it affects the clustering representation.

## 2. Scope of reproducibility

The scope is limited to reproduce the results of the article using the eICU readmission and the MIMIC III mortality prediction tasks, we only reimplement the model, to make sure we are doing the expected or intended architecture and that the variational regularization was working as expected. We did omit the reimplementation of the data preprocessing step as it was the same code the authors used was originally implemented in the Graph Convolutional Transformer paper [3].

With the reimplemented model we are going to replicate the results using a smaller model for the MIMIC and the same one for the eICU dataset. Furthermore, we are going to explore the attention weight patterns on the first graph layer for both readmission and mortality task for a positive patient, Singular Value Analysis for the first graph layer, and plot the embeddings in lower dimensional space after a PCA dimensionality reduction. Finally, we are going to explore the attention weights between the Encoder-Decoder, VGNN with low KL regularization and VGNN with high KL regularization.

## 3. Methodology

### 3.1 Model descriptions

The proposed model is based on Graph Attention Network [5] architecture with added variational layer [3] between encoder and decoder. We fully connect all patient events and gradually adjust the edges via self-attention mechanism to learn unknown representation of the graph. Variational regularization aims to improve encoded graph expressiveness and prevent attention weights from being close to each other. To validate the benefit of variational regularization, we followed the authors and implemented and compared two models. The first one is a simpler Encoder-Decoder architecture, and the other one is based on the Encoder-Decoder but with variational regularization added between the Encoder and Decoder.

**Encoder-Decoder:** The first step in the model is to embed each medical event to a fixed embedding using a linear layer. The next step is to fully connect all observed embedded events. The result is then passed to the encoder which consists of several multi-headed graph self-attention layers to obtain encoded graph representation of medical events. Each self-attention layer first calculates self-attention weights for each pair of observed events:

$$\alpha_{i,j} = \frac{\exp\left(LeakyReLu\left(a^T[Wh_i||Wh_j]\right)\right)}{\sum \exp\left(LeakyReLu\left(a^T[Wh_i||Wh_k]\right)\right)}, (1)$$

where $h$ is the embedded representation, $W$ and $a$ are learnable parameters, || indicates concatenation, and the sum in the denominator is across all neighbors of node $i$. Once obtained, the coefficients are applied in linear combination of the observed nodes followed by an activation function:

$$H^{(l+1)} = \sigma\left(A^l W^l H^l\right), (2)$$

where $\sigma$ is ReLU activation function. We apply several such self-attention layers and concatenate results together. The encoded nodes along with an additional decision node that is fully connected with the encoder nodes are passed to the decoder. The decoder consists of just a single self-attention layer like the encoder self-attention layer. Finally, the decoder result is processed by a separate fully connected network to get the final prediction. The model optimizes a simple binary cross-entropy loss $L_{bce}$:

$$L_{bce} = -\sum w_c y_c \log\left(\sigma(\overline{y_c})\right) + (1 - y_c)\log\left(1 - \sigma(\overline{y_c})\right), (3)$$

where $y_c$ is the true label, $\overline{y_c}$ is the model output, and $w_c$ is negative-to-positive ratio that penalizes errors on positive samples. This weight is needed because the data is imbalanced.

**Variational Graph Neural Network (VGNN):** This model is the same as the Encoder-Decoder model, but with an additional step between the Encoder and the Decoder. This step is the variational part that creates a Gaussian Distribution $N(\mu, \Sigma)$ from the $h_i^{(L)}$ (encoder's output), and from that distribution we randomly sample a vector $z_i$ which is then passed to the decoder. Additional to this step, the model uses a Kullback–Leibler divergence (KL) loss (Similar to Variational Autoencoders [4]) between the learned distribution and a $Prior\ Distribution \sim N(0, I)$ to regularize the distribution and avoid weight concentration within a single dimension in a high dimensional space. Effectively, it forces the learned distribution by the encoder to be close to normal gaussian. The final loss that the model optimizes is:

$$L = L_{bce}(y, \hat{y}) + \beta \cdot KL[q(Z|X)||p(Z)], (4)$$

where KL is Kullback–Leibler divergence between two distributions, $q(Z|X)$ is the distribution generated by the encoder, $p(Z)$ is the prior distribution (assumed to be normal gaussian), $L_{bce}$ – binary cross-entropy loss, and $\beta$ is KL-divergence weight. Without this weight, KL divergence dominates the bce loss and training does not yield good results. The final loss minimizes prediction error while keeping the encoded distribution close to gaussian to prevent uniformly distributed self-attention weights.

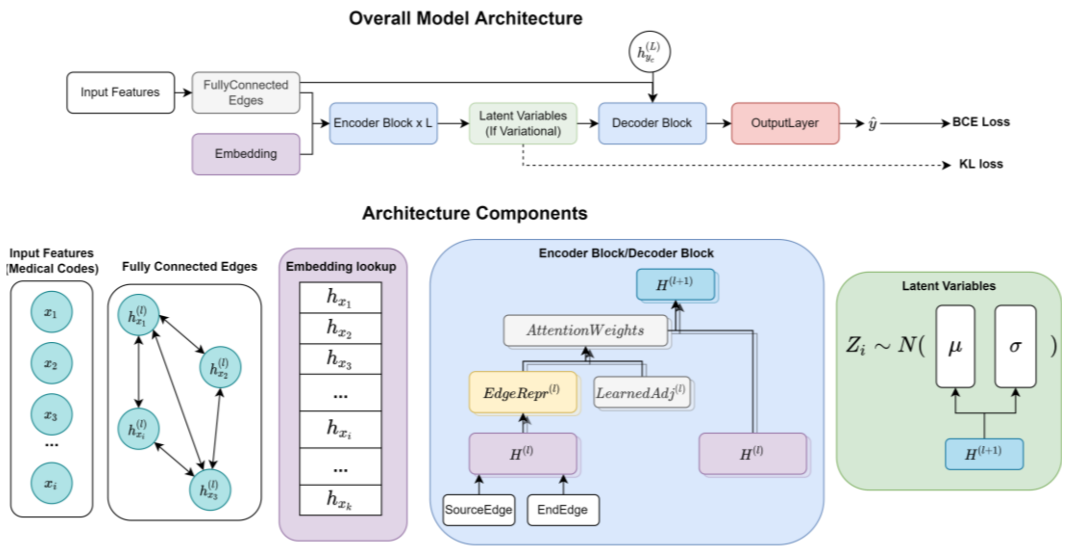

*Figure 1. Model Diagram*

## 3.2 Data description

The authors use three different datasets to test and validate the performance of their method.

**Alzheimer Disease from the NYU Langone health HER (AD-EHR):** This dataset consists of 1.64M distinct patients with medical records from 2016 to 2019 (Long interval window for long term prediction). This dataset includes diagnosis as ICD-10 codes and lab results as LOINC codes. In this replication we didn't use this dataset as it is not publicly available.

**MIMIC-III:** This dataset is publicly available, so it was used within the replication. The task tested using MIMIC was mortality prediction at early days after ICU admission (Short term window interval) which is one of the most widely studied and clinically useful tasks based on this dataset. For this task only the first 24 hours were considered to avoid data leakage from preventative events preceding the mortality event.

The dataset consists of diagnosis codes, procedure codes, and lab values. Lab values are binned into -10, -3, -1, -0.5, 0.5, 1, 3, 10 standard deviations from the mean for each lab type. Therefore, the final dataset consists only of zeros and ones, indicating whether a patient had the corresponding event. The dataset has 50391 records with 5377 positive cases. The total number of features is 11816.

**eICU:** This dataset is publicly available; therefore, it was considered in the replication. The task on this dataset is readmission prediction for ICU patients. The dataset preprocessing steps were the same from Choi et al. [3] in order to compare directly with Choi et al. knowledge graph proposal.

Only diagnosis and procedure codes were used. Lab values change over time and therefore were omitted for simplicity. The dataset has 41026 records with 7051 positive cases. Like MIMIC, the final dataset consists of zeros and ones. The total number of features is 5462. Data was split between train/validation/test sets at 80/10/10 ratio for both datasets.

Since both explored datasets are highly imbalanced, like the paper, we utilize the area under the precision-recall curve (AUPRC) as the primary performance metric, a training up-sampling for the positive classes and class weights on the binary cross entropy loss to level the unbalance training. Furthermore, in both datasets, each patient has only a few events (not more than a few dozen) out of the full set of features.

Therefore, it is crucial to utilize effective sparse representation in the model implementation to make computations faster.

| Total Samples (N° Positive) | MIMIC | eICU |
|---|---|---|
| **Train** | 44449 (8440) | 38441 (11242) |
| **Valid** | 5042 (543) | 4103 (676) |
| **Test** | 5043 (552) | 4103 (754) |

*Table 1 – Train, validation and test dataset samples*

## 3.3 Computational Implementation

First, we explored the code provided by the authors in detail [2]. While studying the original code and implementing our own model, we found a few issues listed below (sorted from most to least critical).

- After obtaining mean and standard deviation from the encoder, KL-divergence was computed only on the first two nodes in the graph instead of the actual input (lines 197-198 in model.py). The selection should use "data == 1" (Boolean indexing or mask slicing) instead of just "data" that consists of 0s and 1s (Integer indexing or slicing), which results in capturing first and second elements only. Therefore, the model applies variational approach only on two nodes in the graph multiple times.
- The additional decoder node is used in the encoder processing logic as well (its embedding from line 188 is passed to encoder self-attention on line 190), where there are operations that are applied to all nodes. However, the paper explains that this node is used in the decoder only.
- The node that was ignored in the eICU dataset because it was a 'readmission event' was actually the following diagnosis *'admission diagnosis|non-operative organ systems|organ system|metabolic/endocrine'* according to the map obtained from the data processing. Nonetheless, we saw 2 nodes that may refer to the readmission: '*admission diagnosis|was the patient admitted from the o.r. or went to the o.r. within 4 hours of admission?|{yes/no}'*. The none graph feature was ignored without affecting performance or any other analysis.

The first issue is the most critical. It makes the model mostly non-variational – only two nodes will have encoded distribution close to standard gaussian. We are not sure if this exact code was used in the paper, or if these issues were introduced as a part of refactoring.

Our code re-implements the model and fixes the above-mentioned issues. The code effectively utilizes sparse representation of the graph in self-attention calculations to significantly speed up computations. This optimization is crucial because each patient has no more than a few dozen events. The code is split into several sub-modules: LayerNorm (Layer normalization), GraphAttentionLayer (Single head of graph self-attention layer), MultiHeadedGraphAttentionLayer (Multi-head graph self-attention layer).

VGNN is the final module that combines the layers above. High-level description of the processing steps:

1. Embed sample into n-dimensional space using a linear transformation
2. Pass embedding through a series of multi-head graph self-attention encoder layers
3. Treat the result as the mean and standard deviation of a normal distribution. Randomly sample from the distribution using reparameterization trick so that the model is differentiable [4].
4. Pass the encoded representation to a single multi-head graph self-attention decoder layer, which consists of the encoded nodes plus an additional decoder node connected to all encoded nodes.
5. Pass the decoded representation to two fully connected layers to get the final prediction
6. Finally, minimize the loss that consists of binary cross-entropy loss and KL-divergence between the encoded and standard normal distributions.

In our model, we used two single-headed encoder layers and one single-headed decoded layer for both MIMIC and eICU datasets. For eICU, we used embedding size of 128 to match the paper. However, for MIMIC, we used embedding size of 256 as opposed to the 756 mentioned in the paper because we did not have enough hardware resources to train such an expensive model. We also found that embedding size of 256 already overfits the data, so the model should be expressive enough. For the output, we used a single fully connected layer with either 128 or 256 nodes followed by ReLU and dropout. The final layer is a single neuron followed by sigmoid activation function.

We used Adam optimizer to minimize the loss function. The optimizer dynamically adjusts the learning rate based on the history of gradient changes. We used an initial learning rate of 0.0001 for all models. However, to improve training even further, we utilized additional learning rate scheduling that decreases the learning rate two times every five epochs. The model was trained in Google Colab on a single Nvidia P100 GPU.

## 4. Code

The code is written with PyTorch library. Code is available in the public Gatech GitHub repository: https://github.gatech.edu/arogachev3/vgnn-reproduce. The repository contains README that explains how to preprocess datasets, train, and evaluate the model.

We re-implement the model while keeping data preprocessing the same since the authors adopted this code from Graph Convolutional Transformer [3]. With this approach, the dataset used is the same, and the results can be reliably compared to the paper. Moreover, the original repository does not contain code to generate analysis results such as attentions and embeddings interpretation and visualization. Therefore, we implemented this part from scratch.

## 5. Results

### 5.1. Performance Comparison

We implemented both Encoder-Decoder and Variational Graph Neural Network (VGNN) models. First, we trained Encoder-Decoder model on both MIMIC and eICU datasets and successfully confirmed that the AUPRC scores match (Table 2). Then, we focused on VGNN training. Our first finding was that the loss function mentioned in the paper does not yield good results. We found that the KL loss term is a few magnitudes larger than the bce term, and it dominates during training. Therefore, we modified the loss function to include an additional KL weight $\beta$ that has a value between 0 and 1. We performed various experiments and found optimal values of $\beta$ for both MIMIC and eICU, 0.02 and 0.005 accordingly. We used embedding size of 256 for MIMIC and 128 for eICU. Using embeddings larger than 256 requires more expensive hardware that was not available to us. The final VGNN model for MIMIC performs better than Encoder-Decoder, as expected. However, for eICU, VGNN performed worse than Encoder-Decoder. One possible explanation is that eICU has much smaller graphs for each patient - as small as five nodes. Therefore, adding variational regularization does not help but even hurts performance.

| **Method** | **AUPRC val (Ours)** | **AUPRC test (Ours)** | **AUPRC test (Original)** |
|---|---|---|---|
| MIMIC, Encoder-Decoder, embedding 256 | 0.6834 | 0.6975 | 0.6962 |
| MIMIC, VGNN, embedding 256: | | | 0.7102 |
| KL weight 0.1 | 0.6842 | 0.6978 | |
| **KL weight 0.02** | **0.6960** | **0.7029** | |
| KL weight 0.005 | 0.6843 | 0.6861 | |
| eICU, Encoder-Decoder, embedding 128 | 0.3730 | 0.4240 | 0.3881 |

| | | | |
|---|---|---|---|
| eICU, VGNN, embedding 128:<br>KL weight 0.02<br>KL weight 0.005<br>KL weight 0.0001 | <br>0.3112<br>0.3284<br>0.3377 | <br>0.3629<br>0.3870<br>0.3911 | 0.3986 |

*Table 2 – Performance comparison between our and original models on eICU and MIMIC datasets*

## 5.2. Visualization of Attention Weights

To visualize and compare the attention weights the models were learning and were using according to the sample, we got them for fully connected edges and when it was passed to the first encoder layer inside the first attention that was applied over it. This attention weights were normalized using the row sum of the edge-to-edge matrix obtaining the following images. Additionally, to identify the nodes, we used the map of the datasets to correctly identify diagnosis, treatments, and lab results.

An interesting result found was that for positive samples the attention was getting higher weights, leading to more concentrated values instead of a uniform value representation. Finally, a comparison between Enc-Dec and the VGNN architecture, we find that the VGNN was getting less noisy weights and more focused columns or features, indicating which features were the receiving the most attention.

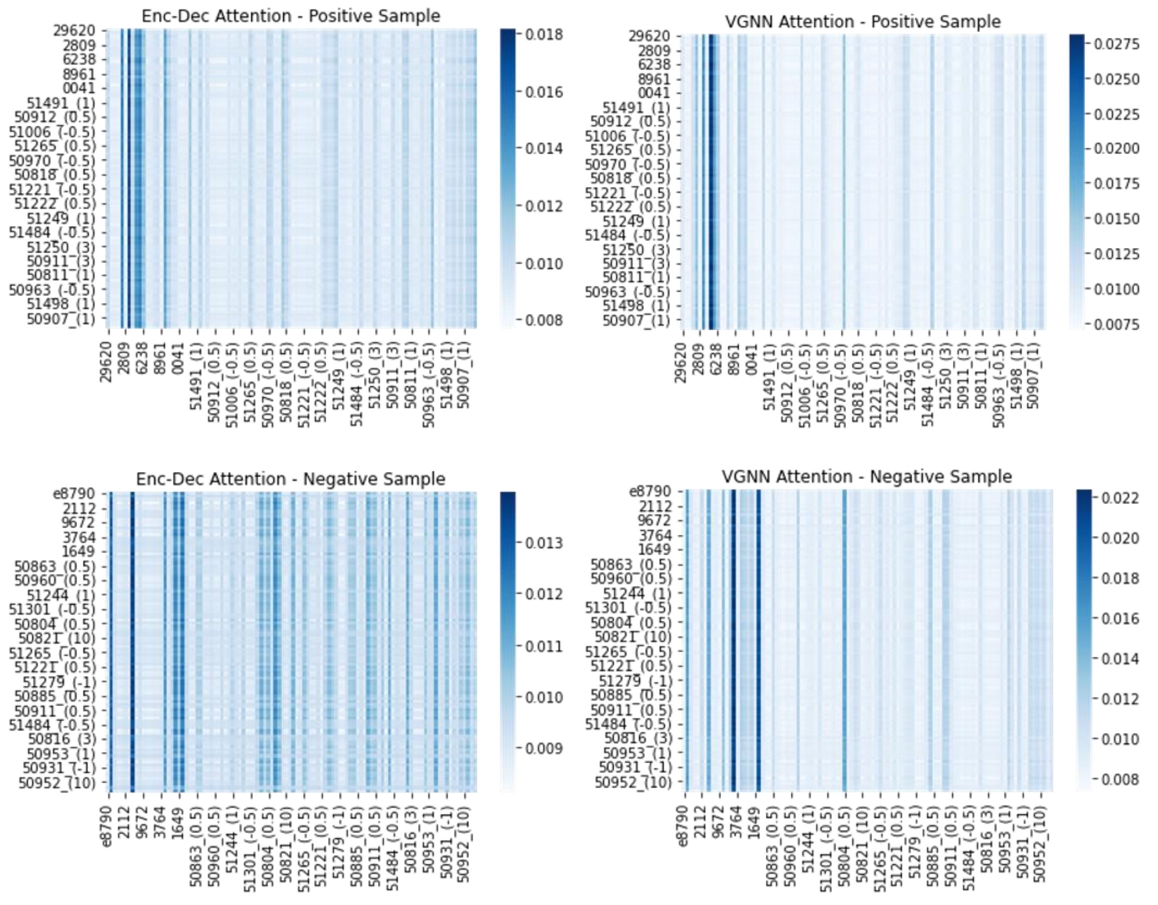

*Figure 2. Attention weights for positive and negative samples comparing both Encoder Decoder architecture and VGNN architecture*

## 5.3. Singular Value Decomposition of Adjacency Matrix

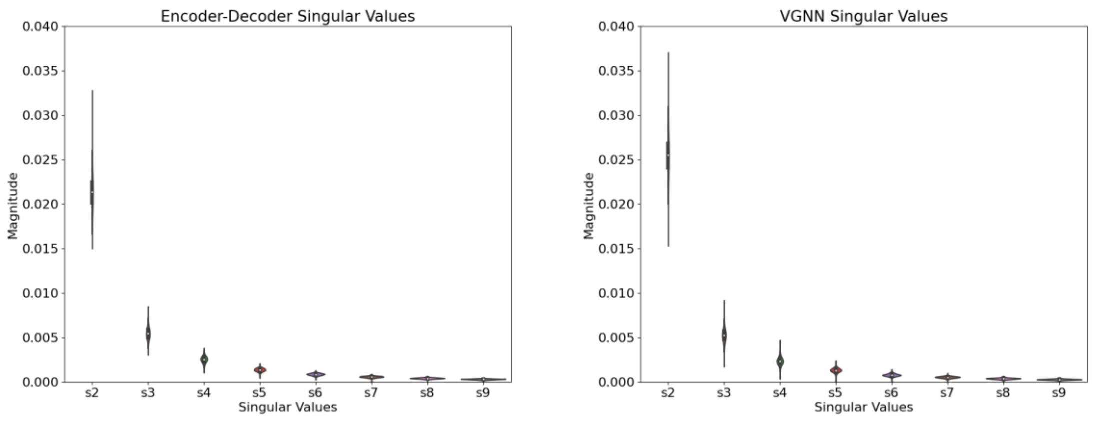

*Figure 3. Singular values of adjacency matrix in the first encoder layer for positive patients in test set*

Learned self-attention weights in the encoder block can be viewed as an adjacency matrix of the graph representation of observed nodes. SVD of the adjacency matrix represents magnitude of messages passed among nodes. Larger values lead to more effective dimensions in the graph layer. It can be shown that the first singular value is always larger or equal to 1 for self-attention weights. Therefore, we analyze the second and greater singular values. We apply SVD to the observed self-attention weights in the first encoder layer for all patients with positive label. The results are shown in Fig. 3. As compared to Encoder-Decoder, VGNN's singular values are slightly larger and with greater range, which means that input nodes are more effectively combined to get the next representation. However, the paper mentioned that the difference between Encoder-Decoder and VGNN should be of at least several magnitudes. We could not reproduce this result.

## 5.4. Visualization of Embeddings

In the paper they do a brief analysis of the Embeddings and how the medical codes where represented. They used a PCA to reduce the dimensionality and visualize in a 2D plot. We did a similar approach to them and compared both Embedding representations Enc Dec and VGNN. We did not find any useful pattern or groups of medical codes that lead to a positive outcome. We got an almost random representation of the codes (See Figure below), instead of the clusters that they showed, this was the case for both MIMIC and eICU. Furthermore, we expected to identify or differentiate diagnosis code, treatments, and lab results on the embedding representation.

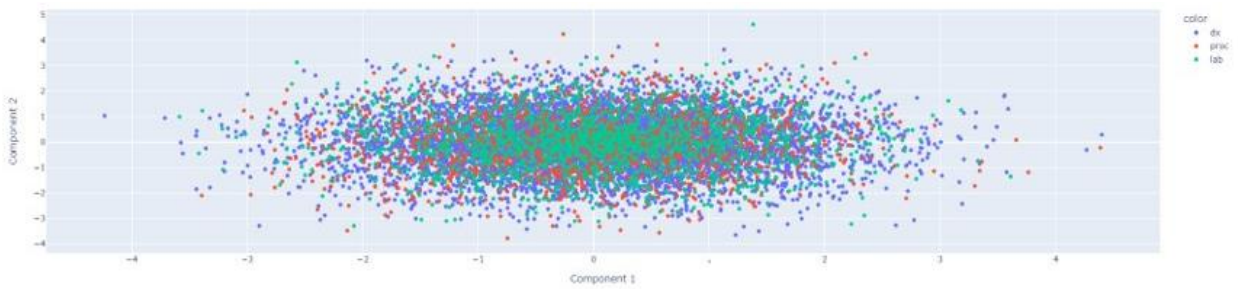

*Figure 4. Embedding representation of medical codes on MIMIC dataset*

## 6. Discussion

The paper introduces a promising variation of graph attention network architecture that should be able to improve the learned graph representation of medical concepts. In this work, we were able to reproduce performance improvement on MIMIC dataset as compared to baseline Encoder-Decoder. However, the model did not perform well as-is. We introduced and found optimal values of an additional KL weight in the loss function to compensate for the dominating KL divergence term. Otherwise, the model prioritized optimization of KL divergence over binary cross entropy, and it led to insufficient results.

By visualizing and analyzing attention weights, we also confirmed that VGNN produces less uniform and more focused attention coefficients, which allows it to learn richer representation. However, unlike the paper, VGNN performed worse than Encoder-Decoder on eICU dataset. We found that for some positive samples, attention weights were the same for all output nodes. For such samples, effectively, all node representations are compressed to a single encoding. It significantly hurts the performance of the model. One possible explanation is that eICU has much smaller graphs for each patient (as small as 5 nodes), so adding variational regularization does not help and only hurts performance.

SVD analysis did not reveal significant differences between Encoder-Decoder and VGNN. Singular values of VGNN are slightly larger and have slightly higher range which indicates that VGNN can propagate messages among nodes more effectively, but they are on the same scale as Encoder-Decoder. On the other hand, the paper mentions that VGNN's singular value should be at least a few magnitudes larger than Encoder-Decoder.

Embeddings 2D PCA visualization also did not reveal patterns in the learned embedding representations. The result is noisy and does not have explicit clusters. We were expecting some small clusters of medical codes that were meaningful for the outcome (readmission or mortality prediction) but we didn0t find that. Also, we could not differentiate between laboratory, treatments, and diagnosis. We think that the internal weights and attentions were the ones that were transforming and selecting the key features that lead to a particular pattern.

Finally, we found a few issues in the code provided by the authors, as described in section 3.3. The most critical issue is that the code applies variational regularization to only two out of thousands of nodes. If this is the exact code that was used in the paper, the results are likely not to be correct.

Our advice to the authors would be to double-check the mentioned in section 3.3 places in the code and re-run all experiments. Moreover, the code that produces the plots in the paper should be included in the repository to help researchers easily reproduce and understand the results. In some cases, it was not clear to us what exactly was done to obtain some plots (such as the PCA plot).

## References

[1] Zhu, W., & Razavian, N. (2021, April). Variationally regularized graph-based representation learning for electronic health records. In Proceedings of the Conference on Health, Inference, and Learning (pp. 1-13).

[2] Github: https://github.com/NYUMedML/GNN_for_EHR

[3] Choi, E., Xu, Z., Li, Y., Dusenberry, M. W., Flores, G., Xue, Y., & Dai, A. M. (2019). Graph convolutional transformer: Learning the graphical structure of electronic health records. *arXiv preprint arXiv:1906.04716*. Choi, E., Xu, Z., Li, Y., Dusenberry, M. W., Flores, G., Xue, Y., & Dai, A. M. (2019).

Graph convolutional transformer: Learning the graphical structure of electronic health records. arXiv preprint arXiv:1906.04716.

[4] Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.

[5] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. arXiv preprint arXiv:1710.10903.

[6] Bruna, J., Zaremba, W., Szlam, A., & LeCun, Y. (2013). Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:1312.6203.