Graph Neural Network Application on Clinical Data for Diabetes Predictive Modeling

Eirini Ornithopoulou

NOVEMBER 2, 2024
HALMSTAD UNIVERSITY
Deep Learning using Pytorch Course Project HT2024

Table of Contents

Introduction

Diabetes is a chronic disease that affects millions of people worldwide and is a significant contributor to global morbidity and mortality rates. Early detection and precise outcome prediction of diabetes are critical for effective management and intervention, enabling healthcare providers to make informed decisions that could lead to better patient outcomes and reduced healthcare costs.

Traditional machine learning models have been employed extensively to assist diagnosis[1], but they often struggle to capture the complex interdependencies among patient features and medical conditions due to their reliance on flat, tabular representations of data[2].

In recent years, Graph Neural Networks (GNNs) have emerged as a promising tool in medical data analysis, particularly for tasks involving relationships between entities. Unlike conventional machine learning models, GNNs leverage graph structures to model and learn from relational data, offering a more nuanced representation of connections between patients' health indicators[3]. By representing each patient as a node and creating edges based on feature similarities or medical conditions, GNNs can effectively capture the intricate relationships and dependencies within the data, thereby enhancing the predictive power of the model[3].

This project aims to design a GNN-based predictive model for diabetes outcomes using a graph representation of the dataset. The primary objective is to explore how GNNs can capture latent relationships in patient data to improve predictive accuracy over traditional models. Key steps include structuring the dataset into a graph, training a GNN to classify diabetes outcomes, and evaluating its performance based on accuracy and loss metrics. By creating a visualization of training and validation loss over epochs, the project will also offer insights into the model's convergence behavior.

Methodology

The methodology is divided into several steps, including data preprocessing, graph construction, model definition, training, and evaluation. This section describes the various stages.

Data Preprocessing

Conceptually the dataset to be used was MIMIC III[4] or IV[5] which are large clinical datasets, but license to use required training and credentialing, which would land outside the deadline for this project. Therefore another dataset was chosen. The new dataset is smaller and consists of patient features (8) and a target variable indicating the diabetes outcome (1 or 0)[6]. Initial preprocessing involves handling categorical features (gender and smoking history into numerical) and numerical features used directly without treatment of missing data. The dataset is then split into three subsets: training (80%), validation (10%), and test (10%). Also, a feature was engineered for improved performance: combining bmi with age (age*bmi), this improved accuracy by 1-2%. This feature captures the idea that older people with higher BMIs might be at a higher health risk. The data is then scaled using the scikit-learn's StandardScaler ($z = (x - u) / s$). Finally the data, in principle, was transformed into Pytorch tensors before use.

Graph Construction

To leverage the relational structure of the data, a graph representation is built where each patient corresponds to a node, and edges represent relationships based on feature similarities. Conceptually, a fully connected graph would be used, meaning that every node is connected to every other node, allowing the GNN to learn from the entire set of feature relationships (torch.combinations). Node features are composed of the patient's attributes, while the edges allow for message-passing operations during training, helping the model capture latent relationships[3]. However this led consistently to kernel dying due to the memory load. Instead, a kNN graph [7] was levereged to relieve the memory load, where the edges were forcefully sparse. After creating the graph, we extract the edges. An edge represents a connection between two data points (nodes) if one is a nearest neighbor of the other.

Alternatively, instead of using a fixed graph structure like k-NN, a DGCNN model is implemented that learns the optimal graph structure based on feature similarities during training.

Model Definition

1. For the first model, a **Graph Convolutional Network (GCN)**[8]**,** was defined with two graph convolutional layers. The first layer transformed the input features into a 16-dimensional representation. The second layer reduced these to 2 output classes, corresponding to diabetic and non-diabetic statuses. A ReLU activation function was applied to the output of the first layer, and the final output was normalized using a log softmax function for probabilistic interpretation.
2. For the second model, a **Graph Attention Network (GAT)** was used[9]. In this case, similarly, a kNN approach is used to connect nodes and determine the adjecency matrix/information. Then, a GAT model was designed with two graph attention layers to capture complex, attention-weighted relationships among data points. The first GAT layer included 4 attention heads, each generating an 8-dimensional feature representation. The outputs were concatenated, resulting in a 32-dimensional representation. The second GAT layer combined the features from the first layer and produced a 2-dimensional output, corresponding to the two classes (diabetes and no diabetes). The first GAT layer used an attention mechanism to focus on relevant neighbors, dynamically learning which node connections were most informative.
3. The **Dynamic Graph Convolutional Neural Network (DGCNN)** architecture was tested[10]. First an adjacency matrix that is learnable was defined. This matrix is randomly initialized and then optimized during training. The matrix represents the connections between nodes (graph edges), and the model will learn these relationships based on the node features during training. We apply softmax to ensure the learned adjacency matrix is probabilistic, meaning the learned weights represent edge strengths. We use two GCNConv layers to process the node features. The learned adjacency matrix defines the relationships between nodes, allowing the GCN layers to aggregate information from dynamically learned neighbors. However, the data could not be fed as a whole due to memory requirements, therefore batch processing was used. Finally, an attempt was made to compensate for the lost global information while processing data in batches, which was to aggregate the batch adjacency matrices.

Training and Validation

The training is conducted over 100 epochs using binary cross-entropy as the loss function, as this task involves binary classification. The Adam optimizer is used with a learning rate of

0.003-0.005 depending on the model, to update the model's weights. During training, both training and validation losses are tracked, and the model's performance is assessed on the validation set to monitor convergence and detect overfitting. After each epoch, the model's performance on the validation set is recorded to ensure it generalizes well beyond the training data.

Evaluation
Once training is complete, the model's performance is evaluated on the test set, which the model has not encountered during training. Accuracy is calculated to measure the model's classification performance. Additionally, a plot of training and validation losses over the epochs is generated, providing insight into the model's learning dynamics and stability.

Implementation Details
The model is implemented using Python with the PyTorch and PyTorch Geometric libraries, which provide tools optimized for GNNs. The experiments are run on two different systems with CUDA or MPS support, enabling faster computation for matrix operations required in GNN layers. For visualization of the graph model and loss over epochs, libraries Networkx and Matplotlib were used.

Results and Discussion

The methodology explored provides a structured approach to utilizing GNNs for clinical prediction tasks, with specific adaptations to work directly with the dataset as provided. By leveraging graph-based relationships and rigorous validation, this approach aims to yield a robust predictive model for diabetes outcomes.

First of all, the correlation matrix of the dataset (Fig.1), gives us an idea of the dataset, whether there are strong correlations between the features and how heavily each feature contributes to the outcome (diabetes 1 or 0). For example, we can see in Fig. 1 that HbA1c and blood glucose are the most important features. In a fully connected graph model, such relationships are learned because of the representation of the data as it is fed into the model. So each node (patient) carries some global information about all other nodes and each edge can embeds the relationships between nodes.

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes	age_bmi_risk
gender	1.000000	-0.030656	0.014203	0.077696	-0.077919	-0.022994	0.019957	0.017199	0.037411	-0.023664
age	-0.030656	1.000000	0.251171	0.233354	0.228608	0.337396	0.101354	0.110672	0.258008	0.921490
hypertension	0.014203	0.251171	1.000000	0.121262	0.093177	0.147666	0.080939	0.084429	0.197823	0.271067
heart_disease	0.077696	0.233354	0.121262	1.000000	0.027598	0.061198	0.067589	0.070066	0.171727	0.221444
smoking_history	-0.077919	0.228608	0.093177	0.027598	1.000000	0.179361	0.037369	0.040219	0.094290	0.232631
bmi	-0.022994	0.337396	0.147666	0.061198	0.179361	1.000000	0.082997	0.091261	0.214357	0.628232
HbA1c_level	0.019957	0.101354	0.080939	0.067589	0.037369	0.082997	1.000000	0.166733	0.400660	0.124068
blood_glucose_level	0.017199	0.110672	0.084429	0.070066	0.040219	0.091261	0.166733	1.000000	0.419558	0.135513
diabetes	0.037411	0.258008	0.197823	0.171727	0.094290	0.214357	0.400660	0.419558	1.000000	0.316214
age_bmi_risk	-0.023664	0.921490	0.271067	0.221444	0.232631	0.628232	0.124068	0.135513	0.316214	1.000000

Figure 1 Correlation matrix for diabetes dataset

One similar way to visualize the graph data is, as it was used in this venture, is visualizing the closest (”nearest”) data points, as they are fed into the NN – since we haven't been able to feed a fully connected network into the model. In our approach we used kNN (variating k between 5-8) to force sparsity into the data before feeding it. We connect each data point to its 5 nearest

neighbors. You can think of it as drawing lines from each patient to the 5 most similar patients. In Fig. 2, we can see such a representation for part of the data (50 nodes). Just to make the visualization more interesting, the BMI value is added, and the presence of diabetes is color coded red while the absence is green.

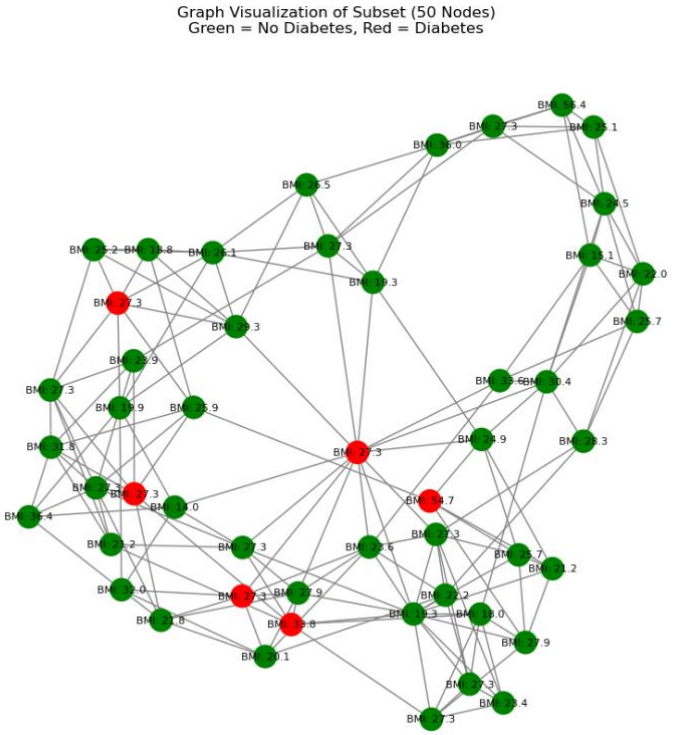

Figure 2 Visualization of 50 nodes (patients) of graph data, connected through kNN.

Three architectures were explored, **GCN, GAT, and DGCNN**. In one of the first attempts, the GCN model achieved a test accuracy of **0.9141** and a test loss of **0.3249.** The training and validation loss curves (see Fig. 3) indicated that the model converged, with both losses decreasing over the course of training. However, the validation remained relatively high, suggesting **overfitting**, which could potentially be mitigated through regularization techniques, data augmentation, hyperparameter tuning, and other ways. Through model engineering, for example by adding combinations of L2 regularization, edge dropout, data augmentation and batch normalization – also for the first time I tried learning rate scheduling – overfitting persisted and the performance did not improve.

In the second architecture, the GAT model achieved a test accuracy **0.9574** and a test loss of **0.1197.** The training and validation loss curves (Fig. 4) indicated consistent convergence, with validation loss stable across epochs, suggesting that the model successfully generalized to unseen data.

The GAT's attention mechanism provided a key advantage by dynamically learning which neighboring nodes (patients) to focus on during prediction. This enabled the model to prioritize relationships with nodes that shared similar health attributes or risk factors, leading to more accurate predictions. In comparison to other models (e.g., GCN or traditional ML models), the GAT's use of multi-head attention allowed it to capture nuanced patterns in the data that would likely be missed by models without an attention mechanism. That said, several optimization

tricks were tried on top of this architecture, such as data augmentation (added noise to features for example, combined edges, removed edges to create perturbation), added dropout layer, and dropped edges, increased the number of GAT layers and that increased or decreased the accuracy by 0.01. So, engineering the model produced insignificant results on this model application for this dataset.

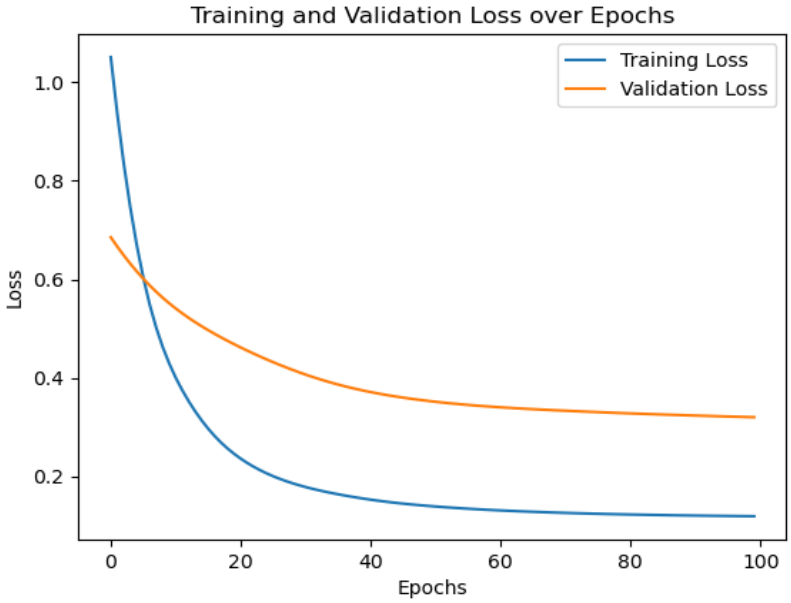

Figure 3 GCN training and validation loss plot.

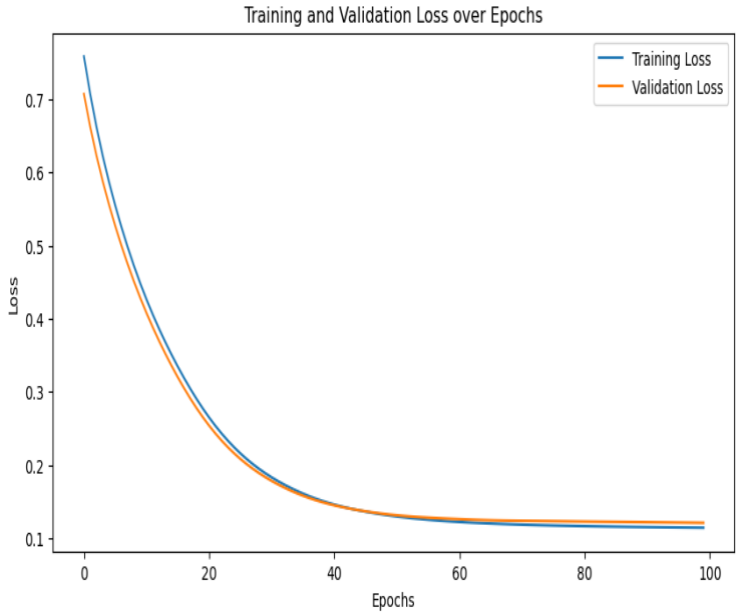

Figure 4 GAT training and validation loss plot.

Finally a **Dynamic Graph Convolutional Neural Network (DGCNN)** was implemented to allow the model to learn relationships between data points without relying on a fixed graph structure (see kNN before). GCNs and GATs use static edge connections, which can be limiting in cases where relationships between nodes (such as patients in a clinical dataset) are not

straightforward. The DGCNN can incorporate a learnable adjacency matrix that dynamically adjusts based on the feature similarity between nodes, making it particularly suited for data with complex or evolving relationships. To manage computational load, batch processing was implemented to enable the model to process subsets of data, with each batch dynamically generating and learning its adjacency matrix during training (the kernel died so many times, I don't feel sorry anymore). The approach did not perform as well as expected (see Fig. 5, and test accuracy of **0.9137**). One idea was that it is probably not working as it should if the learnable adjacency matrix is used on small batches of the data. Then any global relations will not be captured. To compensate for that, one final attempt was made to aggregate the learnable matrices (add and average them). In actuality, this did not improve the performance, leading me to believe the dataset is not nuanced enough to showcase and explore the abilities of these architectures meant to leverage complex relationships within the data.

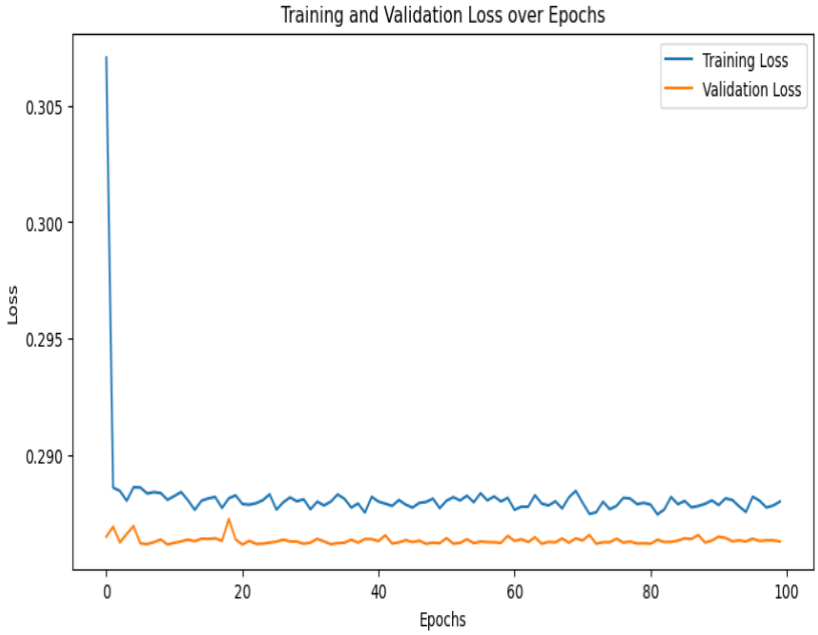

Figure 5 DGCNN training and validation loss plot.

Conclusion

GNNs were successfully implemented, and engineered to improve the accuracy and performance of the model. Out of the 3 main architectures, GAT performed best, without any special engineering required. Was it all worth it? Well, in this particular case, maybe not. Implementing the project was definitely a rich learning experience, however due to the heavy requirements of running such a model fully connected with the relatively simple relationships of the present dataset, the same problem could have been equally solved with logistic regression or a random forest classifier (see code). However, as can be seen in the review by Oss Boll et al.[3], this would not have been the case had the MIMIC III dataset been used for example. The idea behind this project was that the underlying correlations between the various features, or clinical data, for each patient can in principle contribute to better predictions for various applications such as decision support systems for diagnoses or risk predictions. Though in this project it has not been shown, the logic is still sound and it can be applied and explored in larger and more nuanced datasets. Thanks for reading!

1. Zheng, T., et al., *A machine learning-based framework to identify type 2 diabetes through electronic health records.* Int J Med Inform, 2017. **97**: p. 120-127.
2. Goldstein, B.A., et al., *Opportunities and challenges in developing risk prediction models with electronic health records data: a systematic review.* J Am Med Inform Assoc, 2017. **24**(1): p. 198-208.
3. Oss Boll, H., et al., *Graph neural networks for clinical risk prediction based on electronic health records: A survey.* J Biomed Inform, 2024. **151**: p. 104616.
4. Johnson, A.E., et al., *MIMIC-III, a freely accessible critical care database.* Sci Data, 2016. **3**: p. 160035.
5. Johnson, A.E.W., et al., *MIMIC-IV, a freely accessible electronic health record dataset.* Sci Data, 2023. **10**(1): p. 1.
6. Mustafa, T.Z., *Diabetes Prediction Dataset [Data set]*. 2023, Kaggle. https://www.kaggle.com/datasets/iammustafatz/diabetes-prediction-dataset.
7. Kang, S., *k-Nearest Neighbor Learning with Graph Neural Networks.* Mathematics, 2021. **9**(8): p. 830.
8. Parisot, S., et al., *Disease prediction using graph convolutional networks: Application to Autism Spectrum Disorder and Alzheimer's disease.* Medical Image Analysis, 2018. **48**: p. 117-130.
9. Ma, M., et al., *Predicting the risk of mortality in ICU patients based on dynamic graph attention network of patient similarity.* Math Biosci Eng, 2023. **20**(8): p. 15326-15344.
10. Zhu, Y., et al., *Interpretable learning based Dynamic Graph Convolutional Networks for Alzheimer's Disease analysis.* Information Fusion, 2022. **77**: p. 53-61.