Diagnosis and Medication Anomaly Detection in EHRs

EHR Data

- Electronic Health Records (EHRs)
- Contains a record of patient visits to the doctor/hospital
    - Each patient has a unique ID, and each visit is called an encounter
- Each encounter contains a list of lab test results, diagnoses, medications, etc.
    - Lab tests: blood work, microbiology, chemical, etc.
    - Diagnoses: ICD-9 codes
    - Medications: NDC codes
- There is also metadata about each patient such as demographics, age range, etc.

Survey of anomaly detection (1)

- Problem settings of unsupervised AD
    - 1) Out of distribution detection
        - Given a collection of samples, the goal is to identify the source distribution of each sample
        - E.g., a mixture of CIFAR10 and CIFAR100
    - Related work
        - IPMI2017-Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery
        - ICML2018-Deep One-Class Classification
        - ICLR2021-SSD: A Unified Framework for Self-Supervised Outlier Detection
            - Clustering-based method. Partition the data into several clusters and use the Mahalanobis distance as the criterion

Survey of anomaly detection (2)

- Problem settings of unsupervised AD
  - 2) Anomaly detection in time series
    - Detect the anomaly data point from a series of time point
    - Each data point has a feature
  - Related work
    - IJCAI2019-Beatgan: Anomalous rhythm detection using adversarially generated time series
    - KDD2019-Robust anomaly detection for multivariate time series through stochastic recurrent neural network
    - KDD2021-Multivariate time series anomaly detection and interpretation using hierarchical inter-metric and temporal embedding
    - ICLR2022-Anomaly Transformer- Time Series Anomaly Detection with Association Discrepancy
      - Assume the attention distribution of the anomaly data and use reconstruction loss as criterion

Survey of anomaly detection (3)

- Problem settings of unsupervised AD
  - 3) Text anomaly
    - Detect the anomaly text from a corpus
    - Shuffle the words in a sentence
  - Related work
    - NAACL2021-DATE: Detecting Anomalies in Text via Self-Supervision of Transformers
    - Arxiv2022-Self-Supervised Losses for One-Class Textual Anomaly Detection
      - Apply several ssl methods and use the loss as criterion

Survey of anomaly detection (4)

- Criterion of AD
  - 1) Clustering based
    - Partition the data in multiple clusters
    - The anomaly score is formalized as the distance between the data point and the cluster center. E.g.,

$$s_x = \min_m (z_x - \mu_m)^T \Sigma_m^{-1} (z_x - \mu_m) \quad m\text{: m-th cluster center}$$

  - 2) Reconstruction based
    - Assumption: the anomaly data is hard to reconstruct from the normal data
    - Apply different generative models (VAE, AutoEncoder, GAN) to capture the data distribution

Survey of anomaly detection (5)

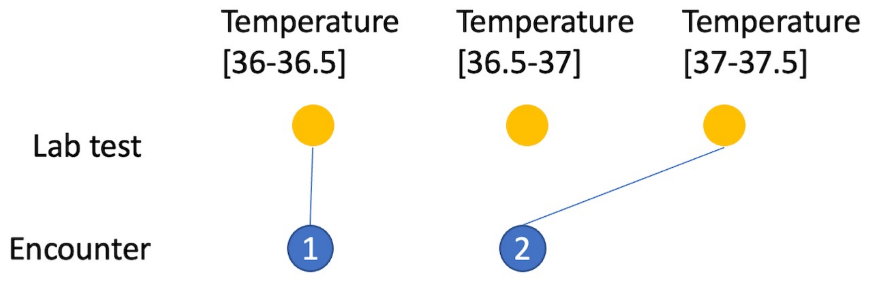
- Evaluation
  - Rank the candidates based on the error/probability
  - Predefine an anomaly/contamination ratio (0.1%,0.5%,1%,…)
  - The candidates with the error/probability higher the threshold will be considered as anomaly data
  - Metrics
    - Precision and recall
    - AUC
    - …

Problem Definition

- Detect the error diagnoses and medications within the dataset
- Graphs can help connect different encounters via similar lab test results/diagnoses/medications
  - E.g. some diagnoses might often occur together
- Motivates a graph with encounters, diagnoses, medications, and lab test results as nodes
  - Essentially several undirected bipartite graphs on encounter nodes and non-encounter nodes merged together
  - An edge is present only between encounter and non-encounter nodes and if and only if that non-encounter node is associated with that encounter

Graph in More Detail

- Issue with some lab test nodes being connected to a majority of the encounters, causing over-smoothing
- Discretize the lab test values into bins, so that encounters will connect with the lab test node that has the exact same value
  - For example, temperature: [36°C, 40°C] -> [36.0°C - 36.5°C], [36.5°C - 37.0°C], [37.0°C - 37.5°C], ...
  - Each temperature value can be assigned to the bin that it falls into
  - Each encounter will connect with the lab test node corresponding to its lab test value



MIMIC-III Dataset Graph Stats

- 39,282 patients and 50,109 encounters
    - 6,984 distinct diagnoses
    - Encounters have 11.97 diagnoses on average
- 3,468 distinct medications
    - Encounters have 36.29 medications on average
    - 28% of medications occur only once
- 2,880 lab tests
    - Encounters have 315.90 lab tests on average
    - Over 49,000 encounters have at least 45 lab tests

Problem Setting + Basic GNN Model

- Given this graph, the task boils down to link prediction
  - Predict the probability that an edge exists between a given encounter node and a given diagnosis/medication node
- Basic GNN Model
  - Task = unsupervised, predict the diagnoses and medications for a patient encounter given the lab test results of that encounter
  - GNN consisting of GCN layers to learn embeddings
  - Loss = logarithmic on the predictions (calculated via a dot product predictor)
    - We inject our own anomalies into the dataset, verified by the med school

Past Encounters

- When evaluating patients, doctors make decisions based on a patient’s current lab test results and their past visit history
- For example, certain conditions could return, or certain chronic conditions might make other conditions more likely
- Furthermore, certain past conditions might invalidate certain medications from being prescribed

Past encounter: Attention Aggregation

- Given an encounter and its corresponding n past encounters
- Average the diagnosis/medication label embeddings of all the past encounters
  - $\{e_1^{past}, \dots, e_n^{past}\}$
- Apply the attention mechanism to aggregate the past encounters' embedding
  - $e_{enc}^{past} = attn\_agg(\{e_1^{past}, \dots, e_n^{past}\} | e_{enc}^{gnn})$
- Transform them into a single embedding and conduct prediction
  - $e_{enc} = W(e_{enc}^{gnn} || e_{enc}^{past})$

Past encounter: Time Bias Method

- Incorporate time information
  - more recent encounter records should be assigned a larger weight
  - MIMIC-III includes the date of all the encounters so we can calculate the interval between two encounters
- Partition the interval into bins and assign a learnable embedding to each bin
- Add the bias term in the attention calculation to inject time information

$$q^T k \to q^T k + b$$

Time Encoding Method

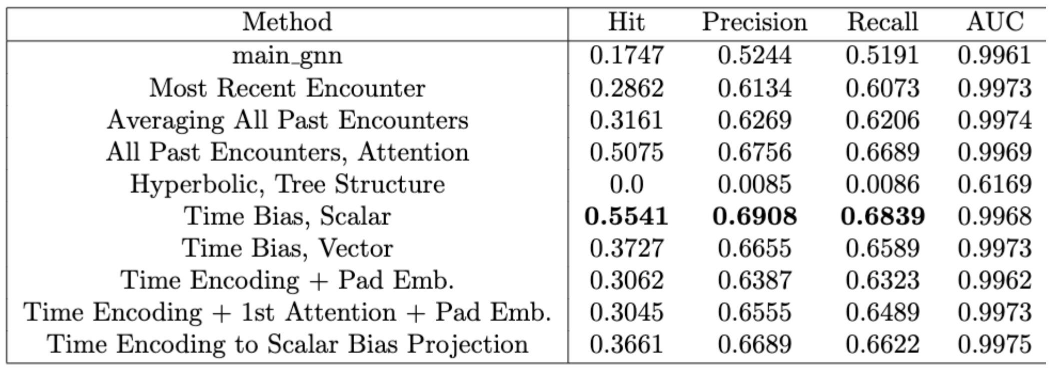
- Instead of partitioning the time intervals into bins, we generate a vector encoding of each time interval
- Similar to transformer positional encodings, we then add these vector encodings to the generated encounter representations before performing attention

$$Attention = softmax\left(\frac{(q + enc)(k + enc)^T}{d_k}\right)(v + enc),$$

where *enc* is the vector encoding of the time interval

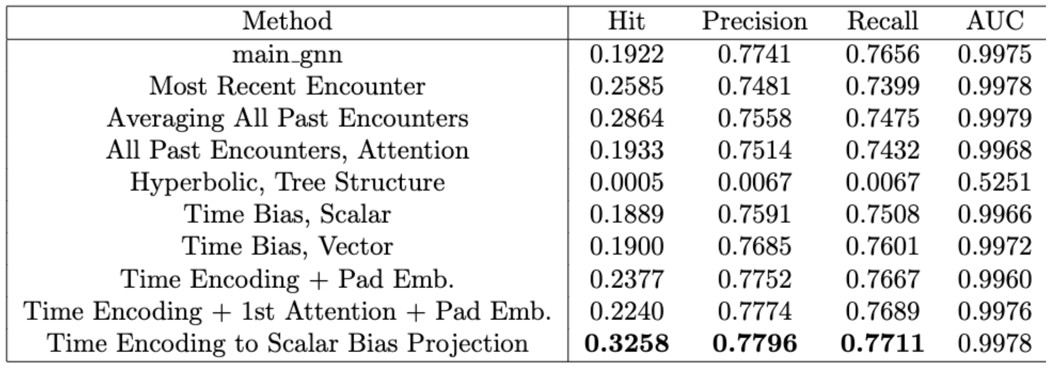
Time Bias + Encoding Results

- Time Bias = learnable additive bias based on bins
  - Bins: [0, 30, 100, 250, 500, 1000, inf]
- Time encoding = similar to transformer positional encoding but for time intervals
- 1st Attention = past encounters attending to themselves
- Pad Emb. = learnable padding embedding for encounters without past encounters



Method	Hit	Precision	Recall	AUC
main_gnn	0.1747	0.5244	0.5191	0.9961
Most Recent Encounter	0.2862	0.6134	0.6073	0.9973
Averaging All Past Encounters	0.3161	0.6269	0.6206	0.9974
All Past Encounters, Attention	0.5075	0.6756	0.6689	0.9969
Hyperbolic, Tree Structure	0.0	0.0085	0.0086	0.6169
Time Bias, Scalar	**0.5541**	**0.6908**	**0.6839**	0.9968
Time Bias, Vector	0.3727	0.6655	0.6589	0.9973
Time Encoding + Pad Emb.	0.3062	0.6387	0.6323	0.9962
Time Encoding + 1st Attention + Pad Emb.	0.3045	0.6555	0.6489	0.9973
Time Encoding to Scalar Bias Projection	0.3661	0.6689	0.6622	0.9975

Table 1: Diagnosis Results, rounded to the nearest ten-thousandth



Method	Hit	Precision	Recall	AUC
main_gnn	0.1922	0.7741	0.7656	0.9975
Most Recent Encounter	0.2585	0.7481	0.7399	0.9978
Averaging All Past Encounters	0.2864	0.7558	0.7475	0.9979
All Past Encounters, Attention	0.1933	0.7514	0.7432	0.9968
Hyperbolic, Tree Structure	0.0005	0.0067	0.0067	0.5251
Time Bias, Scalar	0.1889	0.7591	0.7508	0.9966
Time Bias, Vector	0.1900	0.7685	0.7601	0.9972
Time Encoding + Pad Emb.	0.2377	0.7752	0.7667	0.9960
Time Encoding + 1st Attention + Pad Emb.	0.2240	0.7774	0.7689	0.9976
Time Encoding to Scalar Bias Projection	**0.3258**	**0.7796**	**0.7711**	0.9978

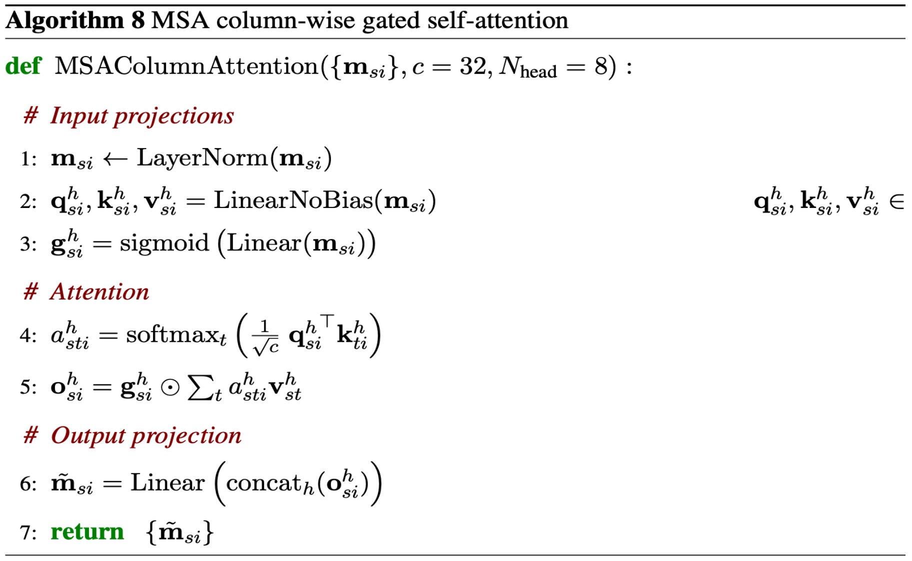
Table 2: Medication Results, rounded to the nearest ten-thousandth

Medication Performance Issue

- Studied encounters with anomalies found by the basic gnn model and by the scalar time bias model
- Overlap in medication data with past encounters was generally around 27%
    - For diagnosis data, around 20% or lower
- The proportion of encounters with past encounters was significantly higher for the encounters that the scalar time bias model correctly identified as having anomalies relative to the analogous encounters for the basic gnn model
- Conclusion: the model seems biased by the past information we've added to it

Gated Attention

- Allow for learning how much of the original embedding to preserve
- Could potentially address the past information bias issue, intuitively speaking
- Have a learnable gate in addition to standard attention to modulate how much of the original embedding to preserve



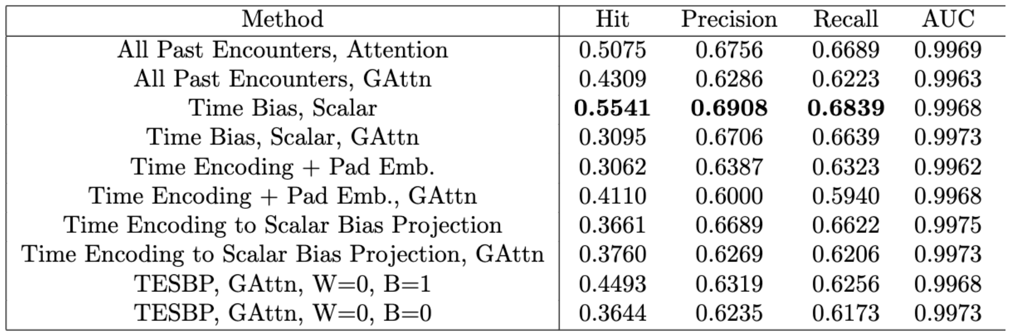
Algorithm 8 MSA column-wise gated self-attention

```
def MSAColumnAttention($\{\mathbf{m}_{si}\}, c = 32, N_{\text{head}} = 8$) :
  # Input projections
  1: $\mathbf{m}_{si} \leftarrow \text{LayerNorm}(\mathbf{m}_{si})$
  2: $\mathbf{q}^h_{si}, \mathbf{k}^h_{si}, \mathbf{v}^h_{si} = \text{LinearNoBias}(\mathbf{m}_{si})$          $\mathbf{q}^h_{si}, \mathbf{k}^h_{si}, \mathbf{v}^h_{si} \in$
  3: $\mathbf{g}^h_{si} = \text{sigmoid}\left(\text{Linear}(\mathbf{m}_{si})\right)$
  # Attention
  4: $a^h_{sti} = \text{softmax}_t \left( \frac{1}{\sqrt{c}} \, {\mathbf{q}^h_{si}}^\top \mathbf{k}^h_{ti} \right)$
  5: $\mathbf{o}^h_{si} = \mathbf{g}^h_{si} \odot \sum_t a^h_{sti} \mathbf{v}^h_{st}$
  # Output projection
  6: $\tilde{\mathbf{m}}_{si} = \text{Linear}\left(\text{concat}_h(\mathbf{o}^h_{si})\right)$
  7: return $\{\tilde{\mathbf{m}}_{si}\}$
```

From AlphaFold

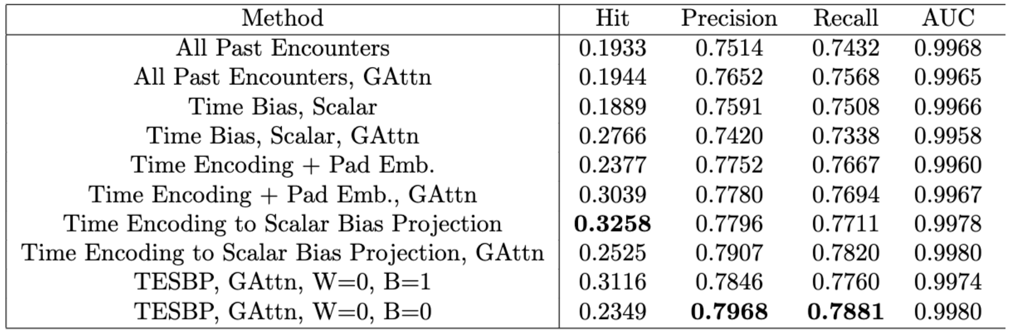
Gated Attention Results

- Compared non-gated attention models to the equivalent with gated attention
  - Also tried different gate initializations
- Conclusions: generally harmful for diagnoses, produced some improvement in some cases for medications



Method	Hit	Precision	Recall	AUC
All Past Encounters, Attention	0.5075	0.6756	0.6689	0.9969
All Past Encounters, GAttn	0.4309	0.6286	0.6223	0.9963
Time Bias, Scalar	**0.5541**	**0.6908**	**0.6839**	0.9968
Time Bias, Scalar, GAttn	0.3095	0.6706	0.6639	0.9973
Time Encoding + Pad Emb.	0.3062	0.6387	0.6323	0.9962
Time Encoding + Pad Emb., GAttn	0.4110	0.6000	0.5940	0.9968
Time Encoding to Scalar Bias Projection	0.3661	0.6689	0.6622	0.9975
Time Encoding to Scalar Bias Projection, GAttn	0.3760	0.6269	0.6206	0.9973
TESBP, GAttn, W=0, B=1	0.4493	0.6319	0.6256	0.9968
TESBP, GAttn, W=0, B=0	0.3644	0.6235	0.6173	0.9973

Table 3: Diagnosis Results with Gated Attention (GAttn), rounded to the nearest ten-thousandth



Method	Hit	Precision	Recall	AUC
All Past Encounters	0.1933	0.7514	0.7432	0.9968
All Past Encounters, GAttn	0.1944	0.7652	0.7568	0.9965
Time Bias, Scalar	0.1889	0.7591	0.7508	0.9966
Time Bias, Scalar, GAttn	0.2766	0.7420	0.7338	0.9958
Time Encoding + Pad Emb.	0.2377	0.7752	0.7667	0.9960
Time Encoding + Pad Emb., GAttn	0.3039	0.7780	0.7694	0.9967
Time Encoding to Scalar Bias Projection	**0.3258**	0.7796	0.7711	0.9978
Time Encoding to Scalar Bias Projection, GAttn	0.2525	0.7907	0.7820	0.9980
TESBP, GAttn, W=0, B=1	0.3116	0.7846	0.7760	0.9974
TESBP, GAttn, W=0, B=0	0.2349	**0.7968**	**0.7881**	0.9980

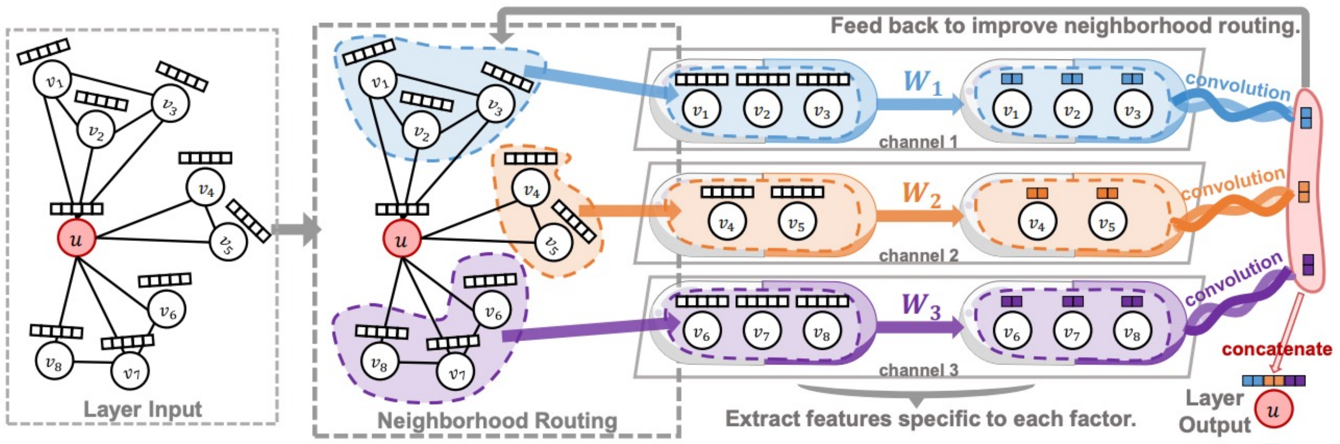
Table 4: Medication Results with Gated Attention (GAttn), rounded to the nearest ten-thousandth

Problematic Trend

- Noticed a trade off between diagnosis and medication performance in the time encoding models
- To address this, we plan to use disentangled learning to learn the representations of diagnoses and medications more independently of each other
  - Consider the two tasks as relatively separate but still allow some communication between the models

Disentangled GNNs

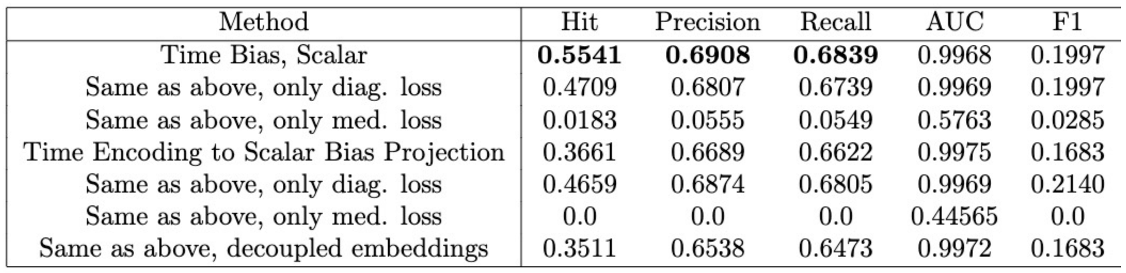
- Motivation: Several different reasons (channels) why an edge could be present in a graph
- Essentially, we'd like to run a mini-GNN on each embedding channel and then stitch those channels together while keeping track of the fact that each edge in each GNN is one edge total across all the GNNs
- Disentangled learning involves separating the embeddings into different chunks (called channels), each corresponding to a different aspect of the entity being embedded, and then learning each channel for each embedding via propagation through a learned channel-specific adjacency matrix



From "Disentangled Graph Convolutional Networks" by Ma et al.

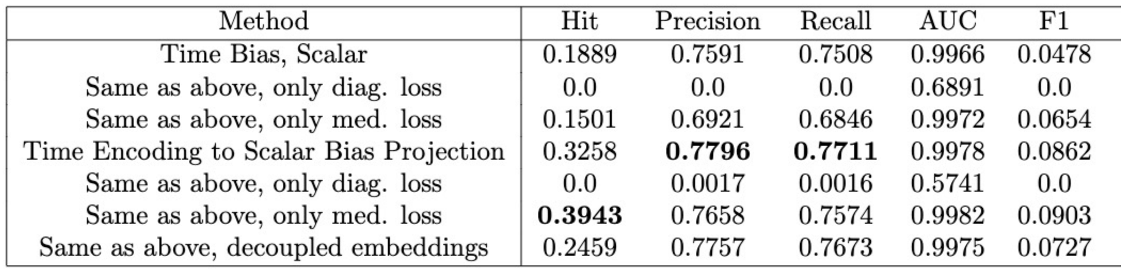
Preliminary Disentangled Experiments

- Testing for tension between the two tasks by using only one of the tasks' loss functions as the whole model's loss function
- Conclusions: having both losses together is generally beneficial, so we don't want complete separation between the two tasks



Method	Hit	Precision	Recall	AUC	F1
Time Bias, Scalar	**0.5541**	**0.6908**	**0.6839**	0.9968	0.1997
Same as above, only diag. loss	0.4709	0.6807	0.6739	0.9969	0.1997
Same as above, only med. loss	0.0183	0.0555	0.0549	0.5763	0.0285
Time Encoding to Scalar Bias Projection	0.3661	0.6689	0.6622	0.9975	0.1683
Same as above, only diag. loss	0.4659	0.6874	0.6805	0.9969	0.2140
Same as above, only med. loss	0.0	0.0	0.0	0.44565	0.0
Same as above, decoupled embeddings	0.3511	0.6538	0.6473	0.9972	0.1683

Table 3: Diagnosis Results, rounded to the nearest ten-thousandth

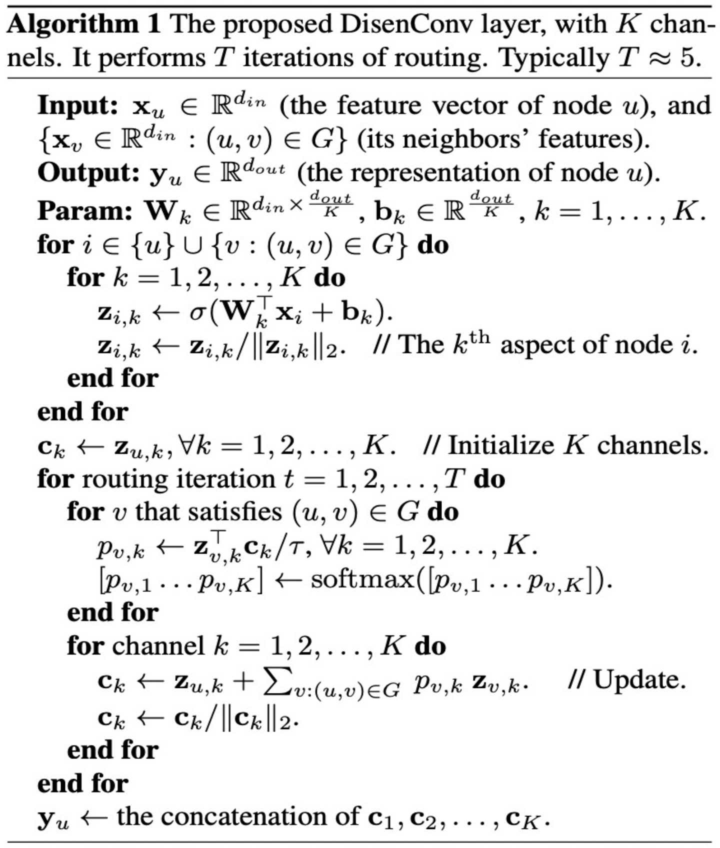


Method	Hit	Precision	Recall	AUC	F1
Time Bias, Scalar	0.1889	0.7591	0.7508	0.9966	0.0478
Same as above, only diag. loss	0.0	0.0	0.0	0.6891	0.0
Same as above, only med. loss	0.1501	0.6921	0.6846	0.9972	0.0654
Time Encoding to Scalar Bias Projection	0.3258	**0.7796**	**0.7711**	0.9978	0.0862
Same as above, only diag. loss	0.0	0.0017	0.0016	0.5741	0.0
Same as above, only med. loss	**0.3943**	0.7658	0.7574	0.9982	0.0903
Same as above, decoupled embeddings	0.2459	0.7757	0.7673	0.9975	0.0727

Table 4: Medication Results, rounded to the nearest ten-thousandth

Issue with Disentangled Modeling

- Algorithm:
  - Project feature vectors to channel space and normalize
  - For each channel embedding, predict the probability of that edge being associated with that channel and then update the channel embedding
  - Concatenate channel embeddings
- Far too memory intensive
  - O(E)
- Does not scale up to our somewhat large graph (>60k nodes)



Algorithm 1 The proposed DisenConv layer, with K channels. It performs T iterations of routing. Typically $T \approx 5$.

```
Input: $\mathbf{x}_u \in \mathbb{R}^{d_{in}}$ (the feature vector of node $u$), and $\{\mathbf{x}_v \in \mathbb{R}^{d_{in}} : (u,v) \in G\}$ (its neighbors' features).
Output: $\mathbf{y}_u \in \mathbb{R}^{d_{out}}$ (the representation of node $u$).
Param: $\mathbf{W}_k \in \mathbb{R}^{d_{in} \times \frac{d_{out}}{K}}$, $\mathbf{b}_k \in \mathbb{R}^{\frac{d_{out}}{K}}$, $k = 1, \ldots, K$.
for $i \in \{u\} \cup \{v : (u,v) \in G\}$ do
    for $k = 1, 2, \ldots, K$ do
        $\mathbf{z}_{i,k} \leftarrow \sigma(\mathbf{W}_k^\top \mathbf{x}_i + \mathbf{b}_k)$.
        $\mathbf{z}_{i,k} \leftarrow \mathbf{z}_{i,k} / \|\mathbf{z}_{i,k}\|_2$.   // The $k^{\text{th}}$ aspect of node $i$.
    end for
end for
$\mathbf{c}_k \leftarrow \mathbf{z}_{u,k}, \forall k = 1, 2, \ldots, K$.   // Initialize $K$ channels.
for routing iteration $t = 1, 2, \ldots, T$ do
    for $v$ that satisfies $(u,v) \in G$ do
        $p_{v,k} \leftarrow \mathbf{z}_{v,k}^\top \mathbf{c}_k / \tau, \forall k = 1, 2, \ldots, K$.
        $[p_{v,1} \ldots p_{v,K}] \leftarrow \mathrm{softmax}([p_{v,1} \ldots p_{v,K}])$.
    end for
    for channel $k = 1, 2, \ldots, K$ do
        $\mathbf{c}_k \leftarrow \mathbf{z}_{u,k} + \sum_{v:(u,v)\in G} p_{v,k}\, \mathbf{z}_{v,k}$.   // Update.
        $\mathbf{c}_k \leftarrow \mathbf{c}_k / \|\mathbf{c}_k\|_2$.
    end for
end for
$\mathbf{y}_u \leftarrow$ the concatenation of $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_K$.
```

From “Disentangled Graph Convolutional Networks” by Ma et al.

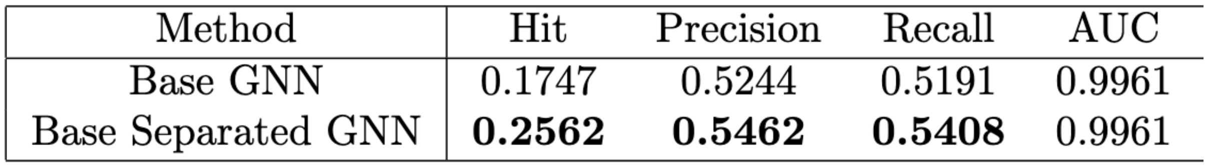
Modified Disentangled GNN

- Separated GNN: take a scalar gate value by which to multiply each edge weight (which would otherwise be the GCN Norm weights)
  - If doing the diagnosis task, multiply by the gate score
  - Otherwise, multiply by 1 – the gate score
  - Simplification of Disentangled learning to learn adjacency matrices through learnable edge weights
  - Since the graph is bipartite, this simplification intuitively approximates disentangled learning

$$score \rightarrow gate(x)\ if\ diag;\ else\ 1 - gate(x)$$
$$edge_w \rightarrow softmax(GCN\_norm(x) * score)$$

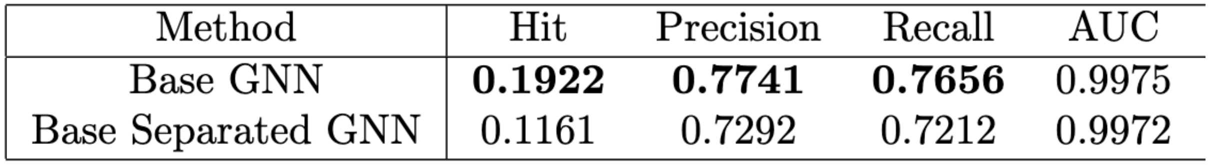
Base Separated GNN Results

- Only tested the basic GNN model vs. the basic Separated GNN model so far
- Significant improvement in diagnosis performance, but we observe the same task tension trend



Method	Hit	Precision	Recall	AUC
Base GNN	0.1747	0.5244	0.5191	0.9961
Base Separated GNN	**0.2562**	**0.5462**	**0.5408**	0.9961

Diagnosis results, rounded to the nearest ten-thousandth



Method	Hit	Precision	Recall	AUC
Base GNN	**0.1922**	**0.7741**	**0.7656**	0.9975
Base Separated GNN	0.1161	0.7292	0.7212	0.9972

Medication results, rounded to the nearest ten-thousandth

Next Steps for Modeling

- Incorporate the Separated GCN convolutions into our different models
- Hierarchical code embeddings
  - Preliminary results are not promising
- Any ideas?