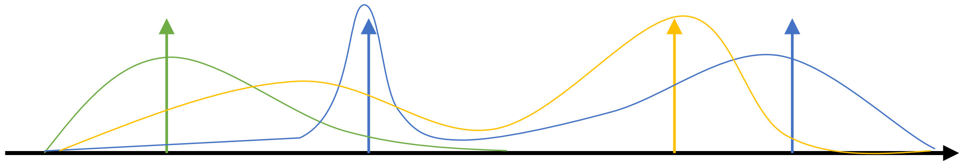
Deep Representation Learning for Prediction of Temporal Event sets in the Continuous Time Domain

P. Dutta, Kawin M, Pratyaksha S. and A. Dukkipati

Indian Institute of Science (IISc), Bangalore

Kawin is currently associated with Observe.ai and Pratyaksha with Goldman Sachs
Work done during their graduate studies at IISc, Bangalore

What are TPPs?

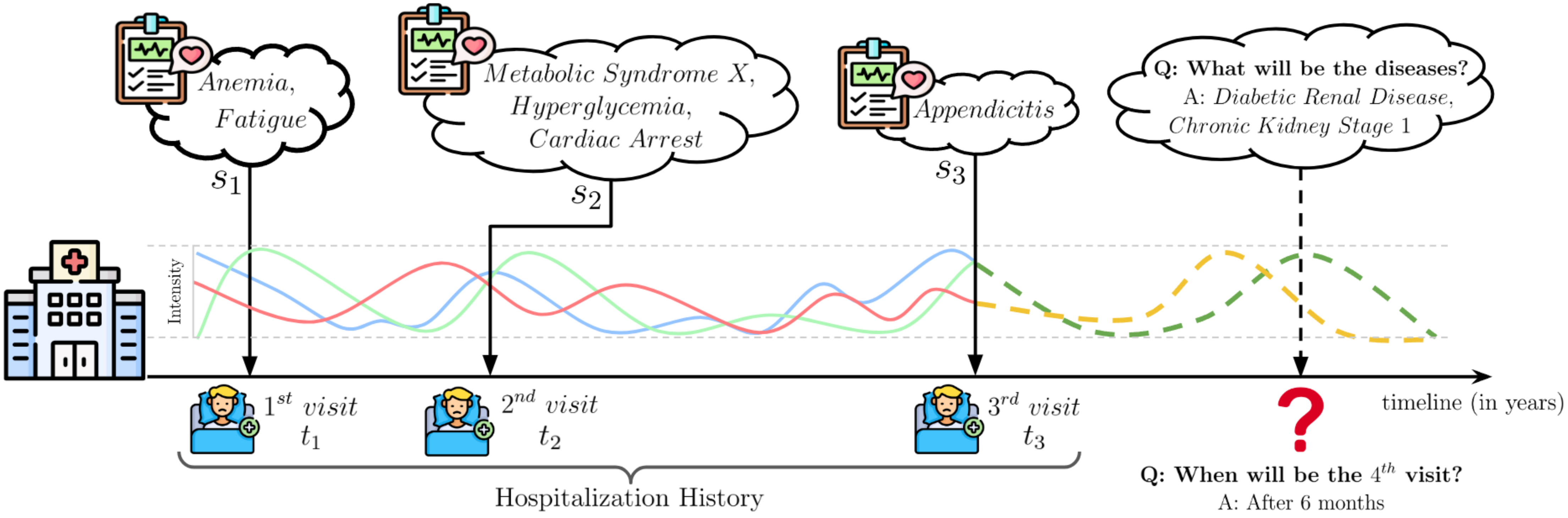


- Temporal Point Processes (TPPs) are probabilistic generative models for continuous-time event sequences.
- TPPs can be learned from data using traditional methods [1, 2, 3] and using deep learning [4, 5, 6].
- Some of the example use cases include modeling and predicting hospital visits, stock portfolio selection, and shopping basket checkouts.

1. M. Winkel – Poisson processes, generalizations and applications (stats.ox.ac.uk/~winkel/bs3a07l1-3.pdf)
2. T. Beckers – An introduction to Gaussian Process models (arxiv.org/pdf/2102.05497.pdf)
3. P. J. Laub and others – Hawkes Processes (arxiv.org/pdf/1507.02822.pdf)
4. Du and others – Recurrent Marked Temporal Point Processes (arxiv.org/pdf/1705.05690.pdf)
5. Mei and Eisner – The Neural Hawkes Process (arxiv.org/pdf/1612.09328.pdf)
6. Zuo and others – Transformer Hawkes Process (arxiv.org/pdf/2002.09291.pdf)

Temporal Event Sets

The notion of sequence of events (in TPPs) is extended to a sequence of sets of events.



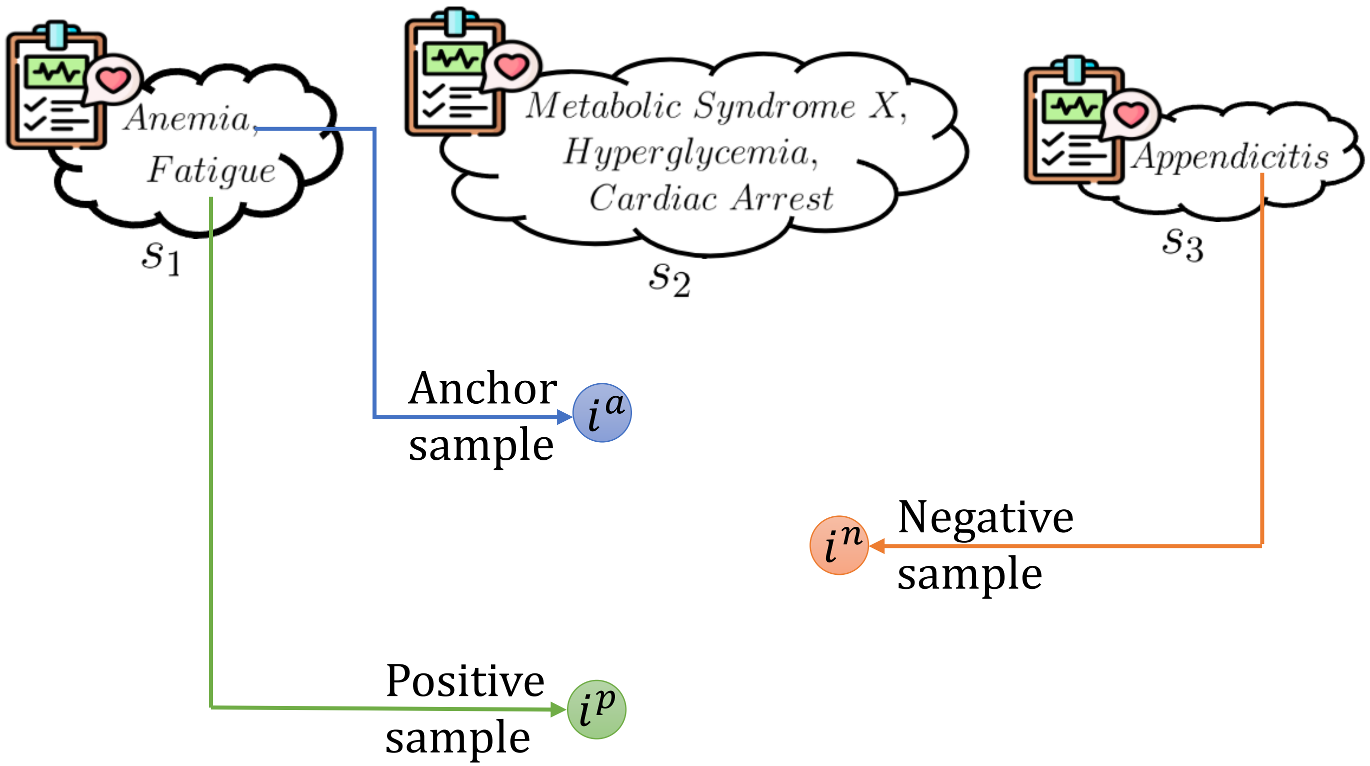
Methodology

How to model Temporal Event sets?

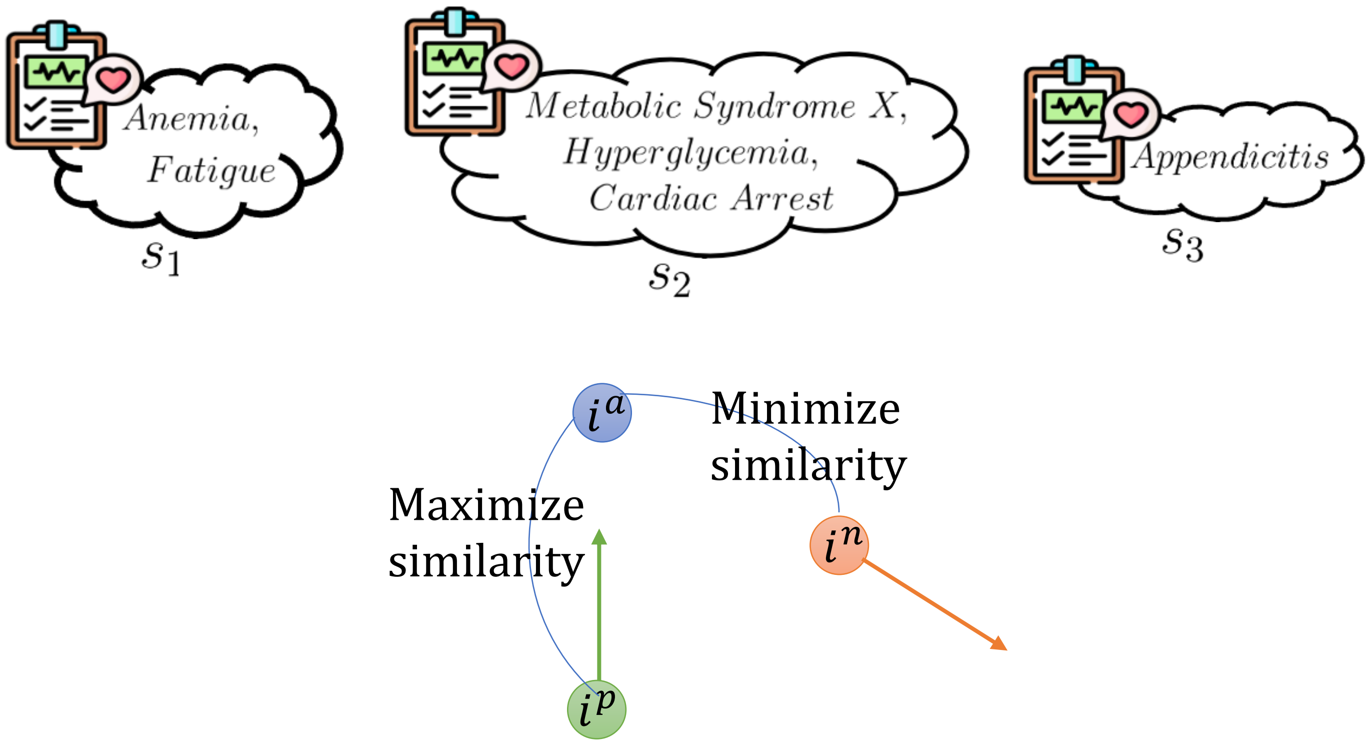
Step 1

Learning Contextual Representations for items in event sets

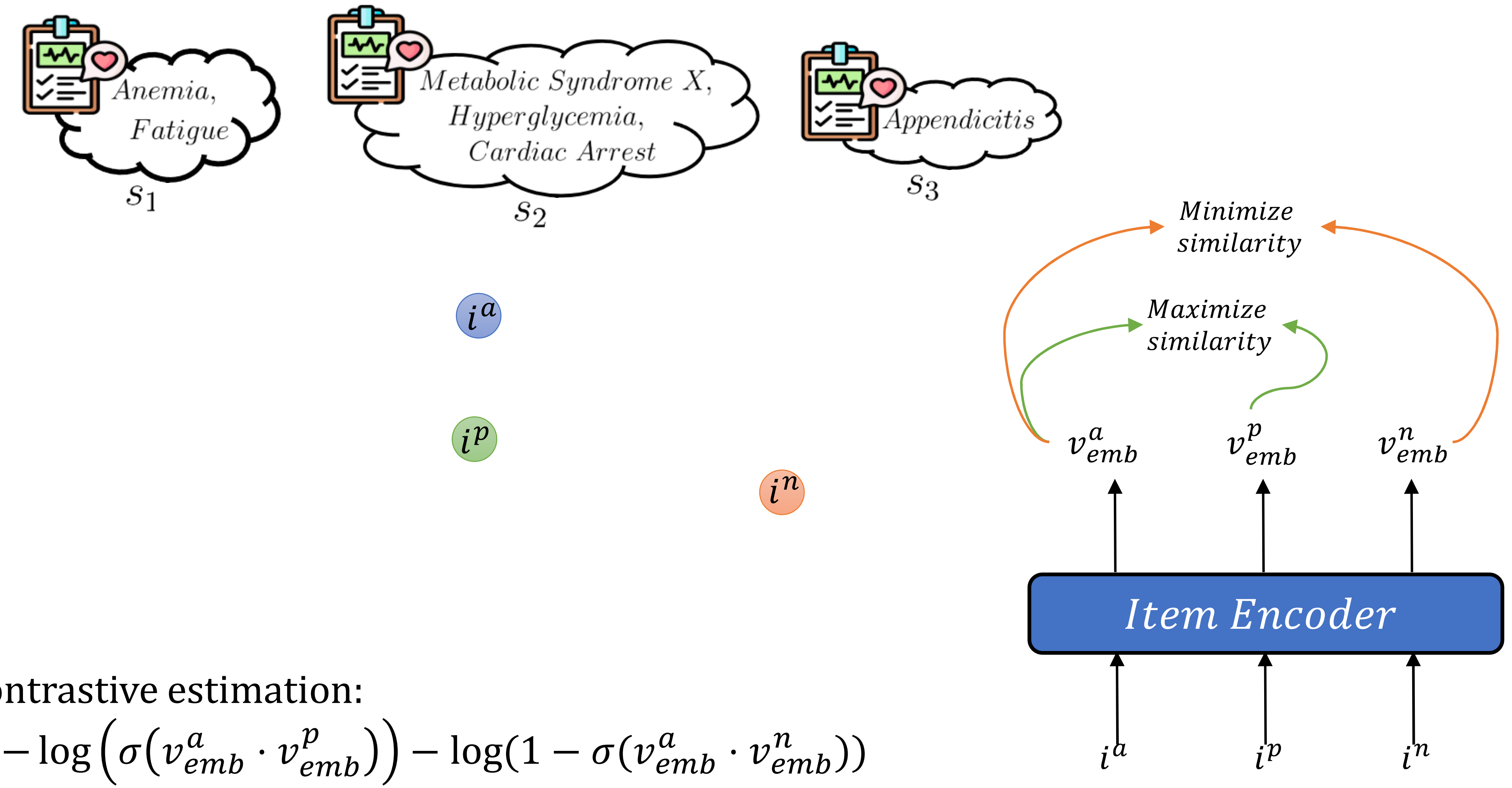
Step 1: Contextual Item Embeddings



Step 1: Contextual Item Embeddings



Step 1: Contextual Item Embeddings



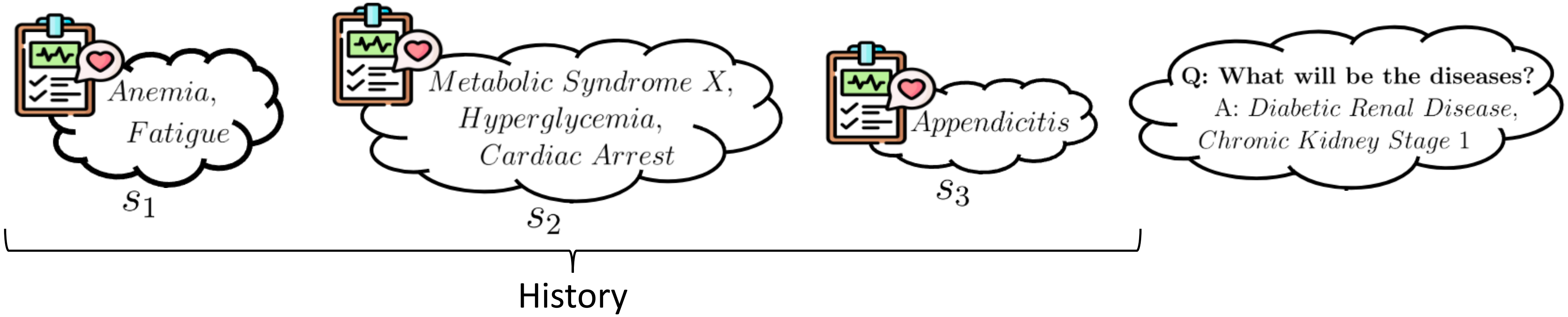
Noise contrastive estimation:

$$\mathcal{L}_{aux} = -\log\left(\sigma\left(v^a_{emb} \cdot v^p_{emb}\right)\right) - \log(1 - \sigma(v^a_{emb} \cdot v^n_{emb}))$$

Step 2

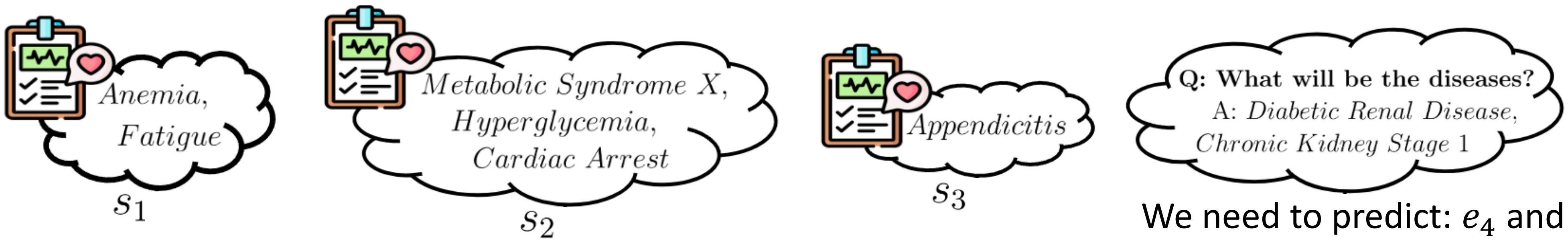
Modeling Temporal Event sets using the embeddings learnt in Step 1

Step 2: Temporal Event set Modeling



Contains: event sets, corresponding timestamps, and associated domain specific features

Step 2: Temporal Event set Modeling



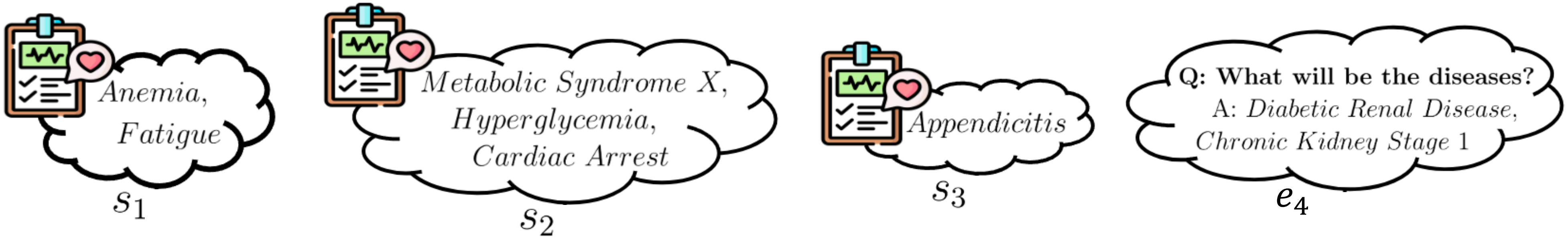
$\mathcal{H}_4 = \{ \ \langle s_1, t_1, f_1 \rangle, \quad \langle s_2, t_2, f_2 \rangle, \quad \langle s_3, t_3, f_3 \rangle \ \}$

We need to predict: e_4 and t_4 given $\mathcal{H}_4$

e_4 is a subset of the event set

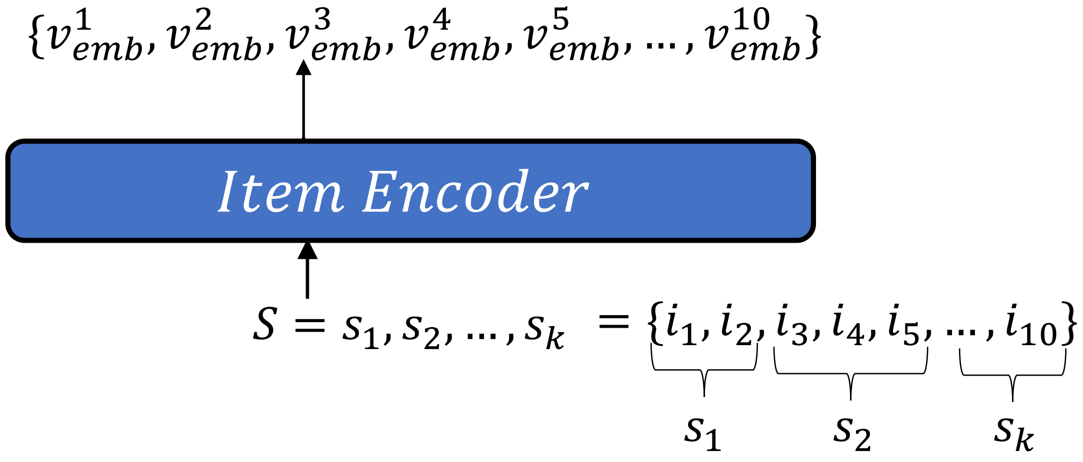
t_4 is the time at which e_4 occurred

Step 2: Temporal Event set Modeling



$$\mathcal{H}_4 = \{ \langle s_1, t_1, f_1 \rangle, \quad \langle s_2, t_2, f_2 \rangle, \quad \langle s_3, t_3, f_3 \rangle \} \rightarrow \langle e_4, t_4 \rangle$$

Step 2: Temporal Event set Modeling



Step 2: Temporal Event set Modeling

$$\{\underbrace{v_{emb}^{1}, v_{emb}^{2}}_{s_1}, \underbrace{v_{emb}^{3}, v_{emb}^{4}, v_{emb}^{5}}_{s_2}, \dots, \underbrace{v_{emb}^{10}}_{s_k}\}$$

Need to preserve the set relations (permutation invariance and equivariance)

But also differentiate among different different sets with how far they are from each other in the timeline

Step 2: Temporal Event set Modeling

Hence, we introduce SpatioTemporal Encodings:

$$\mathbf{v}_{enc}^{pos}(j,d) = \begin{cases} \sin\left(j/10000^{\frac{2d}{\mathbf{d}_{emb}}}\right) & ;\texttt{if } j \texttt{ is even} \\ \cos\left(j/10000^{\frac{2d}{\mathbf{d}_{emb}}}\right) & ;\texttt{otherwise} \end{cases}$$

$$\mathbf{v}_{enc}^{temp}(\mathbf{t}_j,d) = \begin{cases} \sin\left(\mathbf{t}_j/10000^{\frac{2d}{\mathbf{d}_{emb}}}\right) & ;\texttt{if } \mathbf{t}_j \texttt{ is even} \\ \cos\left(\mathbf{t}_j/10000^{\frac{2d}{\mathbf{d}_{emb}}}\right) & ;\texttt{otherwise} \end{cases}$$

$$\mathbf{v}_{enc}(j,\mathbf{t}_j,d) = \mathbf{v}_{enc}^{pos}(j,d) + \mathbf{v}_{enc}^{temp}(\mathbf{t}_j,d)$$

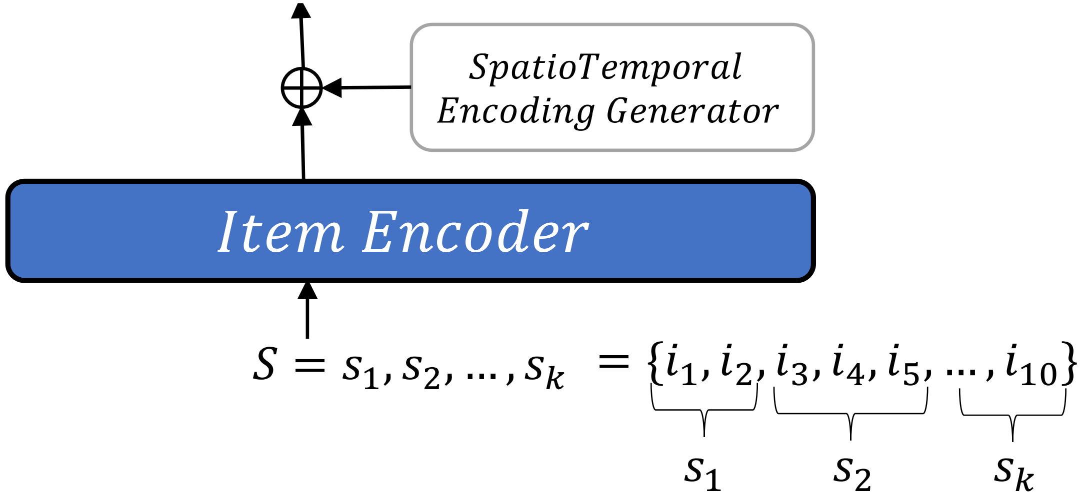
where t_j is the timestamp corresponding to the j^{th} event set s_j for $1 \le j \le k$

$$\{\underbrace{v_{emb}^1, v_{emb}^2}_{s_1}, \underbrace{v_{emb}^3, v_{emb}^4, v_{emb}^5}_{s_2}, \dots, \underbrace{v_{emb}^{10}}_{s_k}\}$$

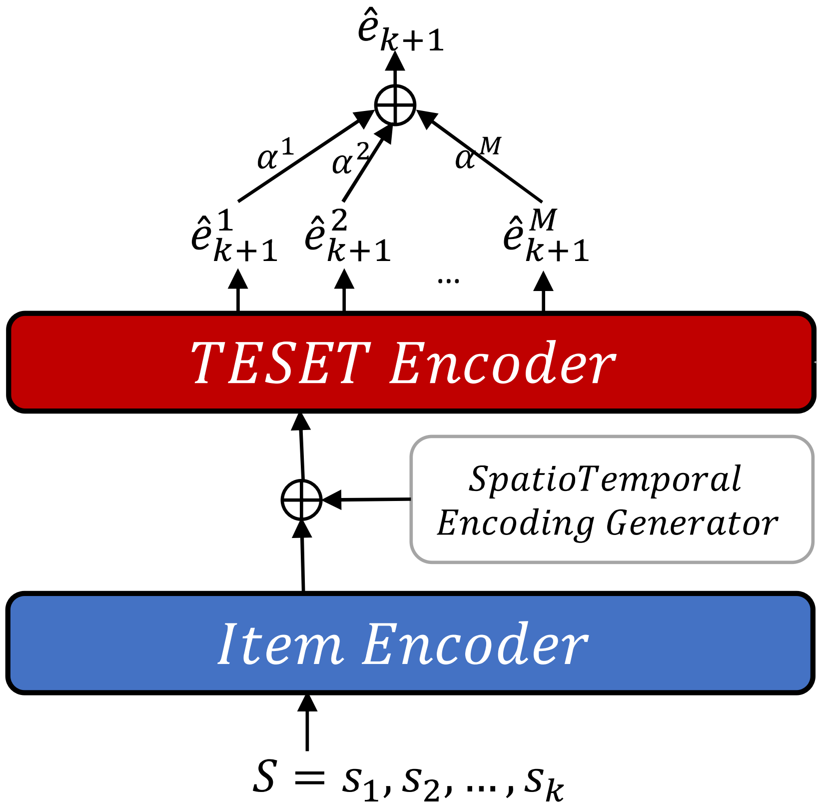
Need to preserve the set relations (permutation invariance and equivariance)

But also differentiate among different different sets with how far they are from each other in the timeline

Step 2: Temporal Event set Modeling



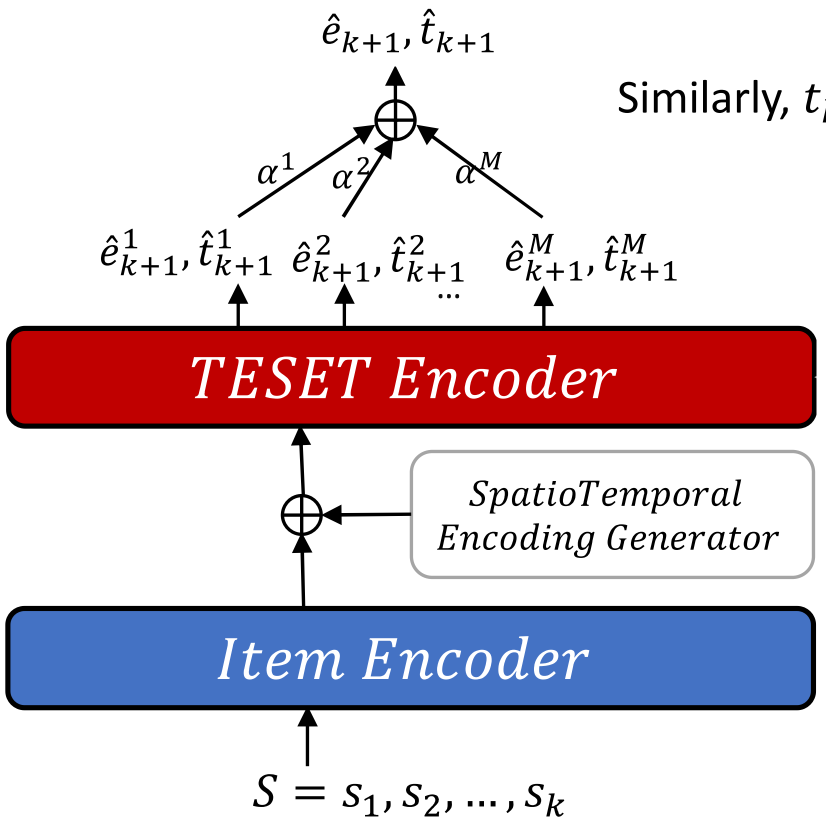
Step 2: Temporal Event set Modeling



Each of the heads predicts a gaussian distribution corresponding to the intensity

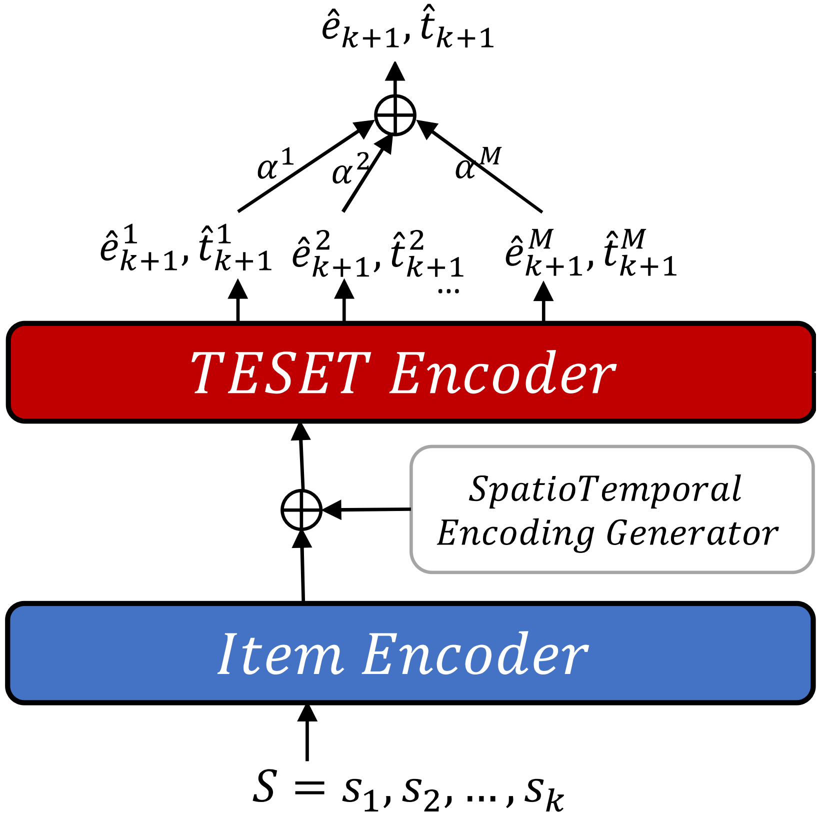
We use a mixture of gaussians to predict arbitrarily complex intensities

Step 2: Temporal Event set Modeling



Similarly, t_{k+1} is also obtained

Step 2: Temporal Event set Modeling



Losses ($\mathcal{T}$ is the target set):

- *Binary Cross Entropy* (to model and predict event sets)

$$\mathcal{L}^{BCE}_{Event} = \frac{1}{|\mathcal{T}|} \sum_{d \in [|\mathcal{T}|]} \mathbb{1}_{\{\mathcal{T}^{(d)} \in \mathbf{e}_{k+1}\}} \hat{\mathbf{e}}^{(d)}_{k+1} + \mathbb{1}_{\{\mathcal{T}^{(d)} \notin \mathbf{e}_{k+1}\}} (1 - \hat{\mathbf{e}}^{(d)}_{k+1})$$

- *Dice Loss* (to handle class imbalance problem)

$$\mathcal{L}^{Dice}_{Event} = 1 - \frac{1}{|\mathcal{T}|} \sum_{d \in [|\mathcal{T}|]} \frac{2\, \hat{\mathbf{e}}^{(d)}_{k+1}\, \mathbf{e}^{(d)}_{k+1} + \epsilon}{\sum_{d' \in [|\mathcal{T}|]} \hat{\mathbf{e}}^{(d')}_{k+1} + \mathbf{e}^{(d')}_{k+1} + \epsilon}$$

- *Huber Loss* (to learn temporal relations)

$$\mathcal{L}^{Huber}_{Temporal} = \begin{cases} \Delta^2/2 & ;\text{if } \Delta < \delta \\ \delta(\Delta - \delta/2) & ;\text{otherwise} \end{cases}$$

where Δ is the absolute value of ($\hat{t}_{k+1} - t_{k+1}$).

We use a linear combination of the the above:

$$\mathcal{L} = \lambda_1 \mathcal{L}^{BCE}_{Event} + \lambda_2 \mathcal{L}^{Dice}_{Event} + \lambda_3 \mathcal{L}^{Huber}_{Temporal}$$

Results

Temporal Event set Modeling

Training method	Synthea		Instacart	
	Event set Predictions (DSC)	Time Predictions (MAE)	Event Set Predictions (DSC)	Time Predictions (MAE)
Baselines:				
Neural Hawkes Process	0.08	2.50	0.29	0.24
Transformer Hawkes Process	0.18	2.41	0.32	0.24
Hierarchical Model	0.12	2.51	0.30	0.23
Ours:				
Temporal Event Set Modeling	0.20	2.29	0.35	0.21
Temporal Event set Modeling + Contextual Embeddings	**0.30**	**2.17**	**0.42**	**0.18**

We gain improvement when compared to baselines irrespective of whether we use contextual embeddings

Fine-tuning to down-stream tasks

Training method	Synthea		Instacart	
	Event set Prediction given time (DSC)	Time Prediction given event (MAE)	Event Set Prediction given time (DSC)	Time Prediction given event (MAE)
Trained from scratch				
Neural Hawkes Process	0.21	5.70	0.35	2.19
Transformer Hawkes Process	0.20	4.52	0.34	2.15
Hierarchical Model	0.19	5.29	0.34	2.20
Ours	*0.22*	*4.28*	*0.38*	*1.83*
Finetuned				
Neural Hawkes Process	0.13	6.01	0.30	2.29
Transformer Hawkes Process	0.19	4.60	0.33	2.24
Hierarchical Model	0.18	5.87	0.35	2.31
Ours	***0.25***	***3.91***	***0.41***	***1.19***

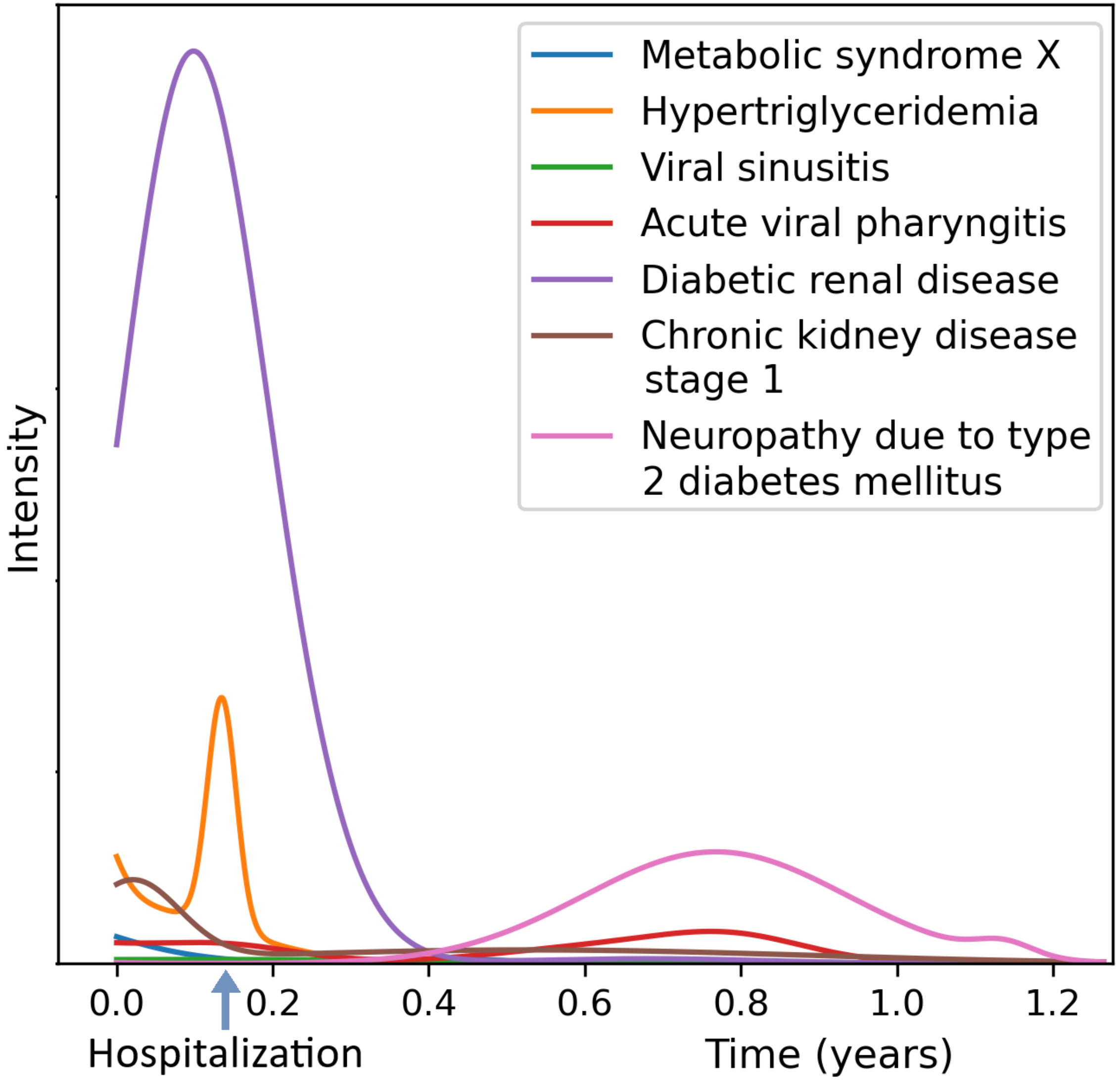
Note than the baselines are not good at being fine-tuned since training from scratch often gives better results.

Transfer Learning (from Synthea to MIMIC-III)

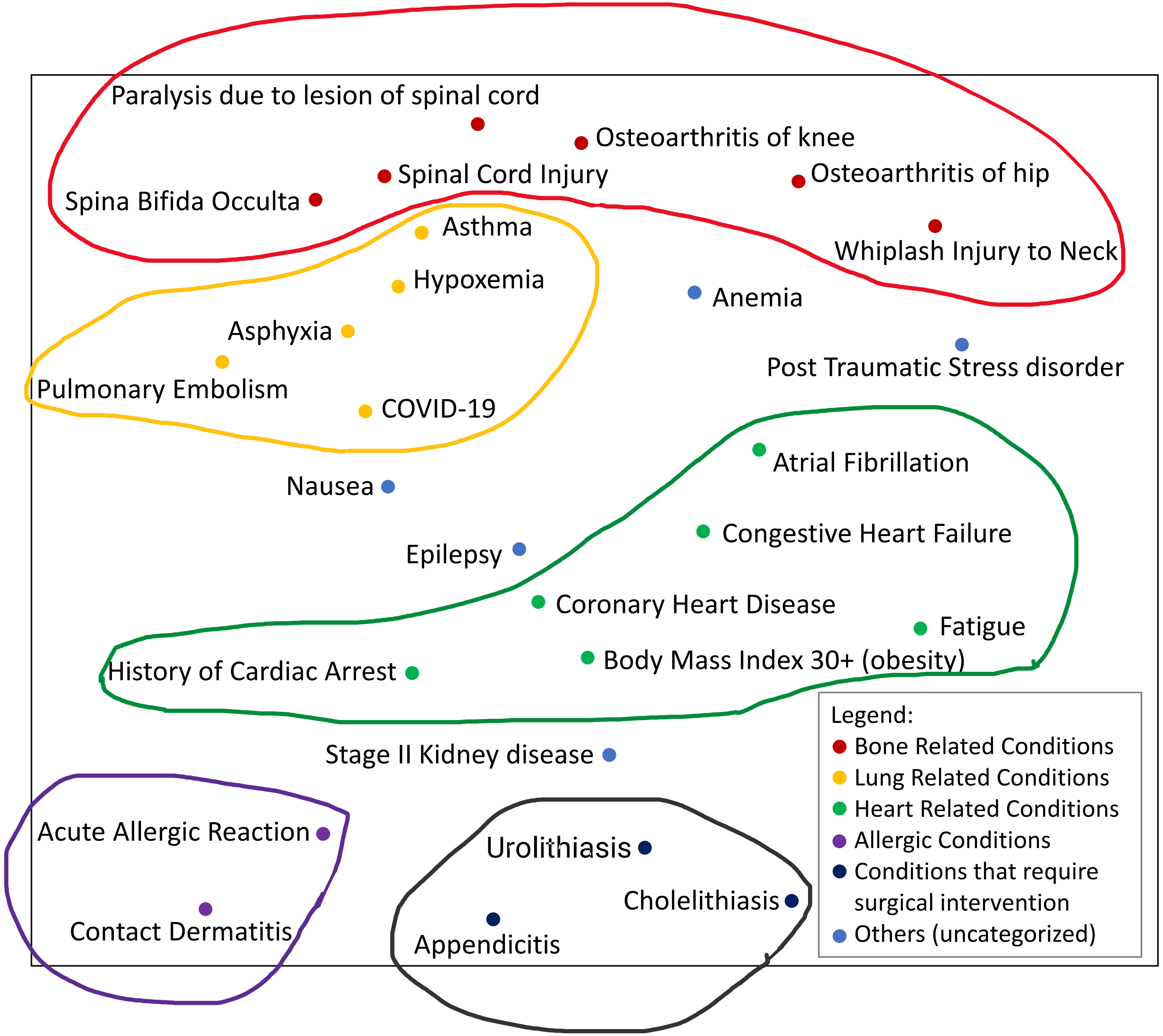
Training Method	Event Set Prediction given time (DSC)	Time Prediction given event (MAE)
Trained from scratch	0.47	0.70
Finetuned (from model pretrained on Synthea)	0.52	0.17

Intensity Prediction Given History

We give the sequence of hospitalization history of a female patient with a history of diabetes and the model predicts how the intensities of various diseases will evolve over time



t-SNE of Contextual Event Embeddings



Summary

- We introduce Temporal Event set Modeling by extending TPPs.
- We show a method to model the same using deep learning.
- We empirically demonstrate the necessity of Temporal Event set Modeling by comparing to strong TPP based baselines.
- We also try to understand the significance for each component in the algorithms through appropriate ablation experiments.

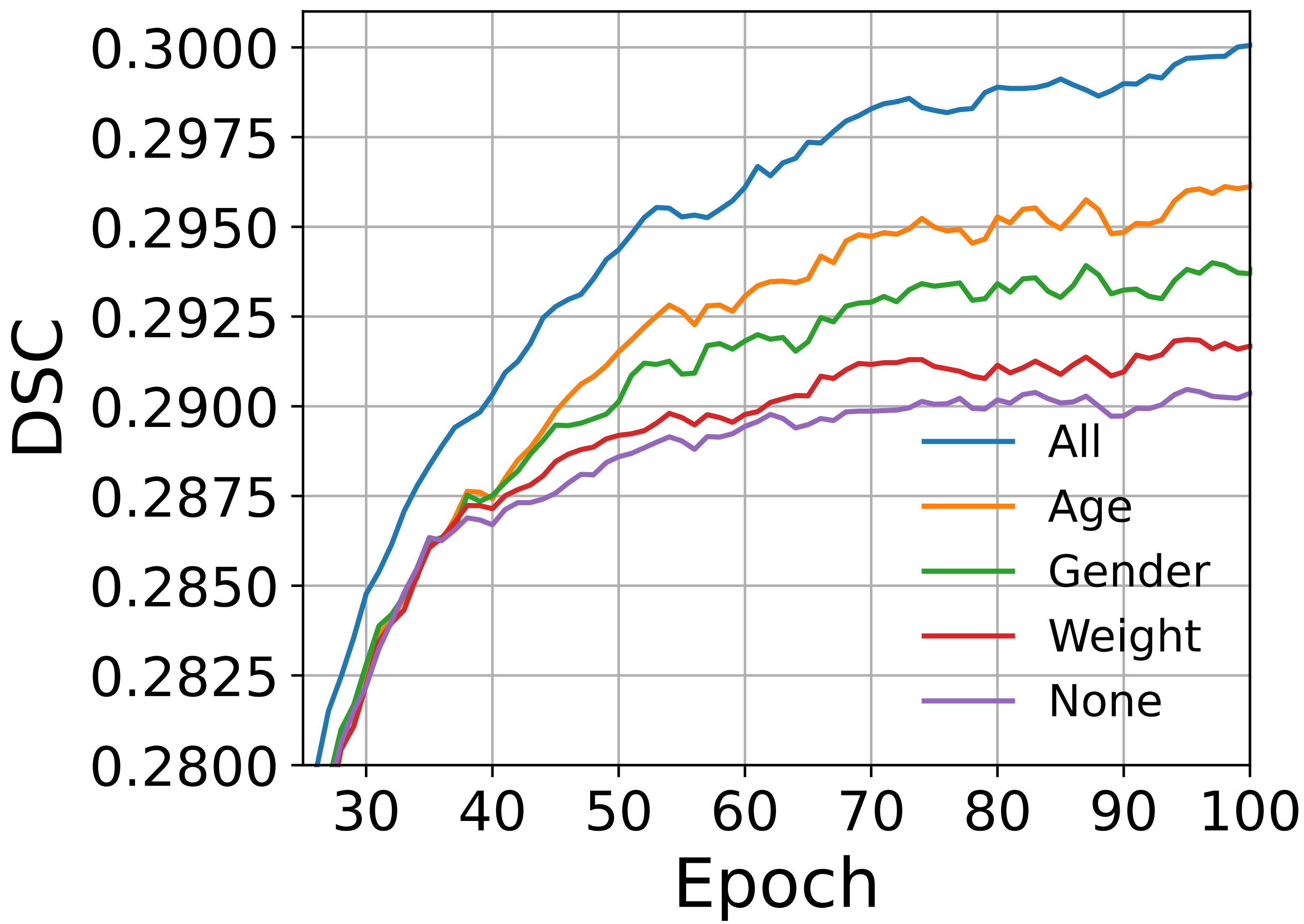
Thank You

SCAN ME

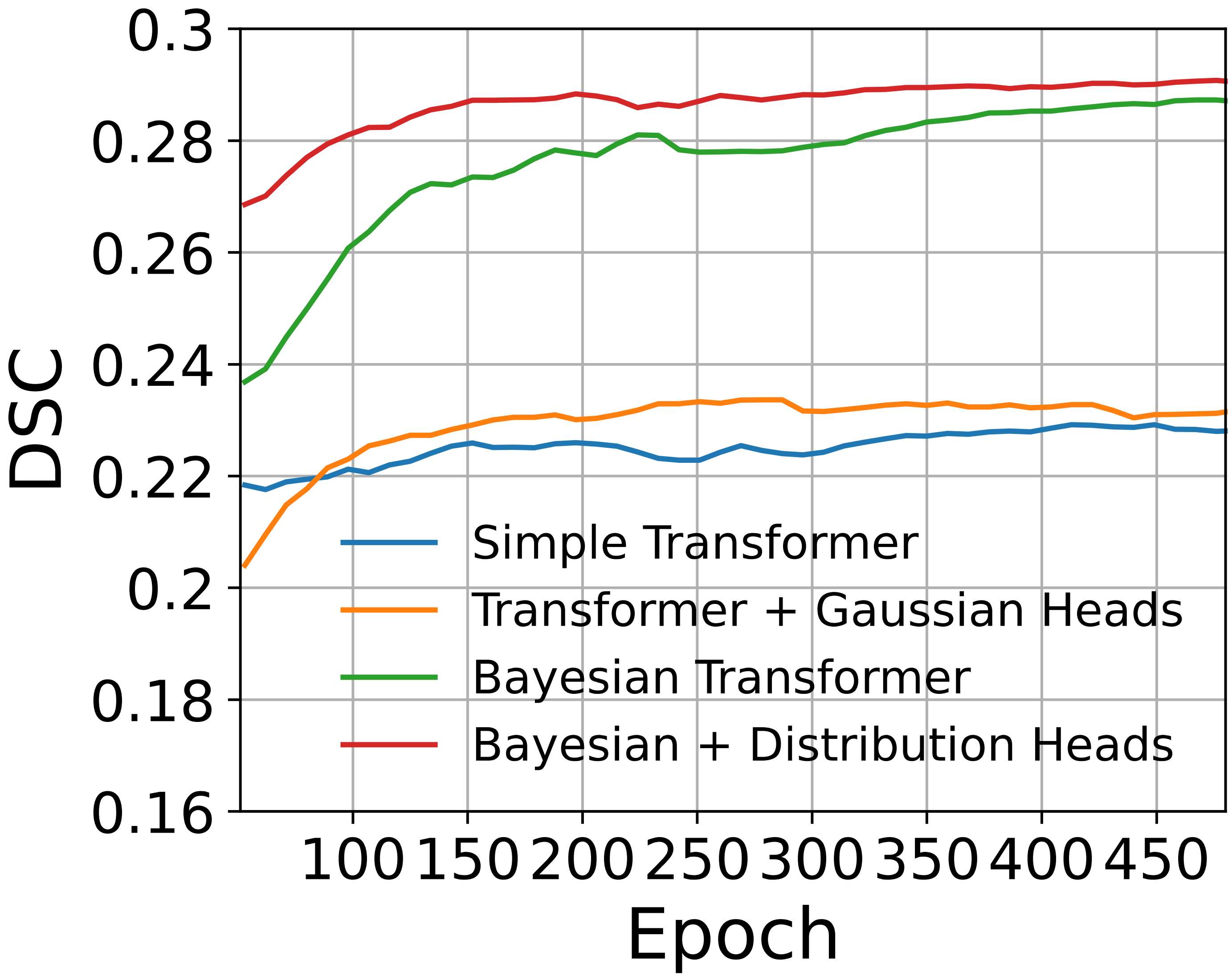
Code available at: https://github.com/paragduttaiisc/temporal_event_set_modeling

Additional Slides

Importance of Domain Specific Features



Importance of Bayesian NNs



Importance of Custom Encodings

Transformer Encodings	Event set Prediction (DSC)	Time Predication (MAE)
Positional Encodings	0.35	0.22
Ours	**0.42**	**0.18**

Training Time and Compute Comparison