PopNet: Real-Time Population-Level Disease Prediction with Data Latency

Junyi Gao
junyii.gao@gmail.com
IQVIA
China

Cao Xiao
danica.xiao@amplitude.com
Amplitude
USA

Lucas M. Glass
lucas.glass@iqvia.com
IQVIA
USA

Jimeng Sun
jimeng@illinois.edu
Department of Computer Science
University of Illinois Urbana-Champaign
USA

ABSTRACT

Population-level disease prediction estimates the number of potential patients of particular diseases in some location at a future time based on (frequently updated) historical disease statistics. Existing approaches often assume the existing disease statistics are reliable and will not change. However, in practice, data collection is often time-consuming and has time delays, with both historical and current disease statistics being updated continuously. In this work, we propose a real-time population-level disease prediction model which captures data latency (`PopNet`) and incorporates the updated data for improved predictions. To achieve this goal, `PopNet` models real-time data and updated data using two separate systems, each capturing spatial and temporal effects using hybrid graph attention networks and recurrent neural networks. `PopNet` then fuses the two systems using both spatial and temporal latency-aware attentions in an end-to-end manner. We evaluate `PopNet` on real-world disease datasets and show that `PopNet` consistently outperforms all baseline disease prediction and general spatial-temporal prediction models, achieving up to 47% lower root mean squared error and 24% lower mean absolute error compared with the best baselines.

CCS CONCEPTS

• **Applied computing** → **Health informatics**; • **Information systems** → *Data mining*; *Spatial-temporal systems*.

KEYWORDS

Spatio-temporal prediction; Graph attention network; Population health prediction

1 INTRODUCTION

Population-level disease prediction is of great significance to society since early forecasting of new disease counts at each location can help government or healthcare providers better optimize medical resources [28] or inform where to build clinical trial sites [9, 29]. Compared with individual-level disease prediction, which predicts disease risk for each patient based on their health records [2, 4, 10–12, 20], population-level disease prediction is usually based on frequently updated online historical disease statistics data collected from certain locations or population groups [8, 9].

Many machine learning or deep learning models have been developed to leverage patient data for individual disease prediction [2–5, 10, 12, 20]. However, they cannot be applied to population-level disease prediction due to the need for accessing individual patient data. Meanwhile, existing population-level prediction models are mostly developed for infectious diseases. For example, epidemiology models such as the Susceptible-Infectious-Recovered (SIR) model were proposed for population-level infectious disease prediction [17, 27, 37]. Recently, several works further proposed to augment such epidemiology models with deep neural networks for capturing spatial and temporal patterns [8, 9].

Existing population-level prediction models often assume that their model inputs (e.g., historical disease statistics) are reliable and accurate, which is often not true. In practice, data collection is time-consuming and has time delays, thus disease statistics require continuous updates to become more accurate [7, 30, 34]. Such a data latency issue needs to be considered in population-level predictions. However, tackling this issue is not straight-forward. There are two main challenges:

- **Incorporating the updated data into the real-time model**. From the temporal perspective, a real-time model needs to be updated whenever an update is made to the historical data. From spatial perspective, different locations may be updated at different frequencies. We need to handle these idiosyncratic updates in our model.
- **Extracting data updating patterns**. Data latency could be induced by various reasons, for example, geographic and demographic proximity between different locations, which causes the

complexity of data updating patterns and brings difficulty for the model to utilize and make predictions. The noise and spatial-temporal correlation of different data streams also add to the difficulties of extracting data updating patterns.

To address these challenges, we propose a population-level disease prediction model (PopNet) which captures data latency and incorporates the updated data for improved predictions. PopNet is enabled by the following technical contributions.

- **Dual data modeling systems to incorporate updated data into the real-time model**. PopNet models real-time data and updated data using two separate systems, each capturing spatial and temporal effects using hybrid graph attention networks (GAT) and recurrent neural networks (RNN). PopNet then adaptively fuses the two systems using both spatial and temporal latency-aware cross-graph attentions in an end-to-end manner. To the best of our knowledge, we are the first work to incorporate updated data in spatio-temporal models.
- **Extract data updating patterns to enrich the spatial and temporal latency-aware attention**. We identify three major data updating patterns. (1) *Spatial Correlation.* Geographically close locations may have similar data updating patterns and locations with similar populations may also have similar characteristics [9, 23]; (2) *Seasonality.* The data updating patterns may be temporally periodic, and (3) *Disease Correlation.* Disease comorbidities may lead to similar updating patterns. We enrich the spatial and temporal latency-aware attentions with these patterns, allowing the model to incorporate these patterns adaptively.
- **Efficient model update**. PopNet can be trained efficiently on the newly added data via better initialization for hidden states of RNN. As a result, PopNet can utilize previous historical patterns without reprocessing old data, which improves efficiency when the training sequences are long and brings convenience for deployment in real-world healthcare systems.

We evaluate PopNet on real-world online medical claims datasets with real-time and update records and a simulated synthetic dataset. Compared to the best baseline model, PopNet achieves up to 47% lower root mean squared error (RMSE) and 24% lower mean absolute error (MAE) on two real-world disease prediction tasks.

2 RELATED WORKS

Over the years, spatial-temporal prediction models have been developed for application tasks such as traffic prediction [13, 15, 38, 39], disease prediction [8, 9, 16], regional demand prediction [38] and general time-series prediction [1]. The recent success of deep learning models, especially GNNs and RNNs, brings promises to better model complex spatial and temporal features. Many research combines graph structures with disease statistics to model regional and temporal disease propagation and achieves more accurate predictions. For example, Deng et al. [8] proposed a location attention mechanism and a graph message passing framework to predict influenza-like illness for different locations. Gao et al. [9] incorporated clinical claims data in graph attention network to predict COVID-19 pandemics and use disease transmission dynamics to regularize RNN predictions. These models achieve good performance on their well-collected datasets. Compared with general

spatio-temporal prediction works, our work more focuses on incorporating updated data into the spatio-temporal model. Since in practice, the input data is not always reliable due to latency or errors and may get updated in the future. We believe this scenario is common in web data and real-world settings.

Consider broader spatial-temporal prediction models in other fields such as traffic prediction, most works also utilize graph neural networks to extract spatial features and use RNNs or attention mechanisms to extract temporal features [19, 25, 32, 35, 36]. Those works also do not have the consideration or model design for data latency. For example, the traffic prediction model GMAN [39] leverages the node2vec approach to preserve graph structure in node embeddings and then samples the neighboring nodes to obtain the embedding. Guo et al. [13] proposed ASTGCN to extract multi-scale temporal features by training three network branches to receive hour-level, day-level, and week-level data. In our work, we enrich the model with spatial and temporal background information, making the model adaptively extract spatial relationships of both close nodes and distant but similar nodes, also from multiple time scales.

3 PROBLEM FORMULATION

Definition 1 (Disease statistics data). The disease statistics data are collected from medical claims or online reports of local health departments from different locations. They can be represented as a 3D tensor $\mathcal{X} \in \mathbb{R}^{N \times T \times F}$, where N denotes the number of locations, T is the number of total timesteps, F is the number of features (i.e., diseases). Matrix $\mathbf{X}^t \in \mathbb{R}^{N \times F}$ and $\mathbf{X}_i \in \mathbb{R}^{T \times F}$ denote slices from the $\mathcal{X}$ tensor from time dimension and location dimension. Vector $\mathbf{x}_i^t \in \mathbb{R}^F$ denotes a slice from the $\mathbf{X}^t$ matrix at i-th location.

Definition 2 (Updated disease data). The real-time statistics maybe unreliable due to time delays during data collection process, therefore every tensor element in $\mathcal{X}$ may be updated at a future timestep. For example, for a specific location at timestep t, after we obtain the initial disease statistics for this timestep, we may constantly receive updates for the statistics of timestep t in future timesteps $t + 1$, $t + 2$, …. All the updated values consist of the updated disease data $\mathcal{U} \in \mathbb{R}^{N \times T \times F}$, which is a 3D tensor. Similar to the original disease data, we also use $\mathbf{U}^t \in \mathbb{R}^{N \times F}$, $\mathbf{U}_i \in \mathbb{R}^{T \times F}$ and $\mathbf{u}_i^t \in \mathbb{R}^F$ to denote different slices from the updated data tensor. Here $\mathbf{u}_i^{t_1}$ refers to data updated for location i at a future time t_1. Suppose it will replace the original disease data $\mathbf{x}_i^{t_0}$, note that $t_1 > t_0$, we define this update latency as $\Delta t_i = t_1 - t_0$. All update latency is aggregated to a 2D matrix $\Delta \mathbf{t} \in \mathbb{R}^{N \times T}$. Value 0 in $\mathcal{U}$ means no updates for those tensor elements.

Definition 3 (Location graph). A location graph can be modeled as an undirected graph $\mathcal{G} = (V, E, \mathbf{A})$, where $\mathbf{V}$ is the set of $|V| = N$ location nodes, E is the set of edges, $\mathbf{A}$ denotes the adjacency matrix of the graph. The edges are computed based on the geographical and demographic proximity between locations, which will be detailedly introduced in following sections.

Problem 1 (Spatial-temporal disease prediction). Given historical original disease statistics $\mathcal{X}$ and updated disease data $\mathcal{U}$, the population-level spatial-temporal disease prediction task is a regression task, which is to predict the future ground-truth number of cases for a certain disease $\mathbf{Y} \in \mathbb{R}^N$ for all N locations at $T + 1$

timestep. We also support multiple-step prediction for next k steps from $T + 1$ to $T + k$ timesteps.

4 THE POPNET MODEL

As shown in Fig. 1, PopNet models real-time data $\mathcal{X}$ and updated data $\mathcal{U}$ using two separate systems, and then adaptively fuses the two systems using both spatial and temporal latency-aware cross-graph attention. Below we introduce PopNet in more details.

4.1 Dual Graph Attention Network

We model the real-time data $\mathcal{X}$ and updated data $\mathcal{U}$ using separate systems for better modal capacities of pattern extractions in both data sources. We employ graph attention networks (GAT) [33] to leverage spatial relations between locations. This way, the prediction for a target location can be improved by utilizing spatial disease patterns discovered in nearby or similar locations. We use two graph attention networks to process $\mathcal{X}$ and $\mathcal{U}$ respectively.

Here $\mathcal{X}$ and $\mathcal{U}$ share the same undirected graph design $\mathcal{G}(V, E, \mathbf{A})$. In graph $\mathcal{G}$, the nodes indicate locations; while the edge connecting node i and j, denoted as w_{ij}, is the similarity between node i and j such that $w_{ij} = p_i^\alpha p_j^\beta exp(-\frac{d_{ij}}{\gamma})$. Here d_{ij} is distance between node i and j, p_i is the population size of node i, and α, β, γ are hyper-parameters. We use a threshold value ω to calculate the graph adjacency matrix as in Eq. (1),

$$\begin{cases} A_{ij} = 1, \ if \ w_{ij} \geq \omega \\ A_{ij} = 0, \ if \ w_{ij} < \omega \end{cases} \tag{1}$$

For simplicity, we focus our discussion in this section on a specific timestep and thus will omit the superscript t. Accordingly the attention score between node i and j will be computed as in Eq. (2),

$$\begin{aligned} \mathbf{z}_i &= \mathbf{W}_z \mathbf{x}_i, \ \mathbf{z}_i^u = \mathbf{W}_z^u \mathbf{u}_i \\ e_{ij} &= \sigma(\mathbf{W}_a(\mathbf{z}_i | \mathbf{z}_j)), \ e_{ij}^u = \sigma(\mathbf{W}_a^u(\mathbf{z}_i^u | \mathbf{z}_j^u)) \end{aligned} \tag{2}$$

where $\mathbf{W}_z \in \mathbb{R}^{|\mathbf{z}_i| \times F}$, $\mathbf{W}_a \in \mathbb{R}^{|\mathbf{z}_i| + |\mathbf{z}_j|}$, $\mathbf{W}_z^u \in \mathbb{R}^{|\mathbf{z}_i^u| \times F}$, $\mathbf{W}_a^u \in \mathbb{R}^{|\mathbf{z}_i^u| + |\mathbf{z}_j^u|}$ are attention weight matrices for the GAT networks, σ denotes the LeakyReLU activation function, and $(\cdot|\cdot)$ denotes the concatenate operation. Then we use softmax function to normalize the obtained attention score as in Eq. (3),

$$a_{ij} = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}(i)} \exp(e_{ik})}, \quad a_{ij}^u = \frac{\exp(e_{ij}^u)}{\sum_{k \in \mathcal{N}(i)} \exp(e_{ik}^u)} \tag{3}$$

where $\mathcal{N}(i)$ denotes the set of one-hop neighbors of node i.

Likewise, we use the multi-head attention mechanism [33] to enrich the model capacity by calculating K independent attention scores, where K is the number of attention heads. We obtain the aggregated node embedding as given by Eq. (4),

$$\mathbf{g}_i = \sigma(\frac{1}{K} \sum_{k=1}^{K} \sum_{j \in \mathcal{N}(i)} a_{ij}^k \mathbf{W}_g^k \mathbf{x}_j), \mathbf{g}_i^u = \sigma(\frac{1}{K} \sum_{k=1}^{K} \sum_{j \in \mathcal{N}(i)} a_{ij}^{u,k} \mathbf{W}_g^{u,k} \mathbf{u}_j) \tag{4}$$

where $\mathbf{W}_g^k \in \mathbb{R}^{|\mathbf{g}_i| \times F}$ and $\mathbf{W}_g^{u,k} \in \mathbb{R}^{|\mathbf{g}_i^u| \times F}$ are the weight matrices for the k-th attention head in two GATs, respectively. Therefore, for each node i, we will obtain two node embeddings $\mathbf{g}_i$ and $\mathbf{g}_i^u$ for real-time data $\mathcal{X}$ and updated data $\mathcal{U}$ respectively.

4.2 Cross-Graph Embedding Fusion with Spatial Latency-aware Attention

After obtaining all the node embeddings for updated data and real-time data, we would like to utilize the updated historical data to make better predictions. However, this is not a straightforward task. The latency in data updating can vary between two embeddings of the same node. Hence, directly concatenating or summing two embeddings may confuse the prediction network and lead to inferior prediction results. Besides, there is a latency in the updated data because those locations can be updated at a different frequency. To incorporate these complex latency patterns, we design the spatial latency-aware attention (S-LAtt) to fuse spatial embeddings.

The idea of S-LAtt is to use the node embedding as the query to aggregate spatial patterns from nearby or similar nodes (i.e., locations), assuming they have similar data updating patterns. To better quantify such similarity, we learn a spatial information embedding (SIE) $\mathbf{v}^s$ for each node, where the spatial information includes populations, the numbers of hospitals and ICU beds, longitude, and latitude. For node i, SIE is obtained via Eq. (5),

$$\mathbf{v}_i^s = MLP(\mathbf{S}_i) \tag{5}$$

where $\mathbf{S}_i$ denotes the spatial information of node i. Since these spatial information are in general static, same nodes in both GATs share the same $\mathbf{S}_i$. Based on the the node embedding and the SIE, we compute the cross-graph attention score as in Eq. (6),

$$e_{ij} = \sigma(\mathbf{W}_a(\mathbf{W}_g(\mathbf{g}_i | \mathbf{v}_i^s) + \mathbf{W}_u(\mathbf{g}_j^u | \mathbf{v}_j^s))) \tag{6}$$

where $j \in \mathcal{N}(i)$, $\mathcal{N}(i)$ is the neighboring nodes set of node i in the location graph $\mathcal{G}$.

In addition to spatial similarity, we also notice that the longer the latency is, the smaller the marginal influence the new data will have on our final prediction, thus we use time latency to regularize this attention score. To be specific, we utilize the temporal latency Δt_{ij} between node i and j, and design a heuristic function to use this temporal latency Δt as in Eq. (7),

$$f(\Delta t_{ij}) = \frac{1}{\log(1 + \exp(\Delta t_{ij}))}. \tag{7}$$

Then the attention weight a_{ij} is regularized as in Eq. (8),

$$\hat{e}_{ij} = e_{ij} f(\Delta t_{ij}), \quad a_{ij} = \frac{\exp(\hat{e}_{ij})}{\sum_{k \in \mathcal{N}_u(i)} \exp(\hat{e}_{ik})}, \tag{8}$$

and we can get the aggregated updated embedding as in Eq (9).

$$\hat{\mathbf{g}}_i^u = \sigma(\sum_{j \in \mathcal{N}_u(i)} a_{ij} \mathbf{g}_j^u) \tag{9}$$

Finally, we concatenate the node embedding $\mathbf{g}_i$, the aggregated updated embedding $\hat{\mathbf{g}}_i^u$ and the original input data $\mathbf{x}_i$ as in Eq. (10).

$$\hat{\mathbf{g}}_i = (\mathbf{g}_i | \hat{\mathbf{g}}_i^u | \mathbf{x}_i) \tag{10}$$

4.3 Recurrent Neural Network with Temporal Latency-aware Attention

In addition to spatial patterns, we also employ the gated recurrent unit networks [6] to extract the temporal patterns based on multivariate time series from each node. To simplify, we focus our
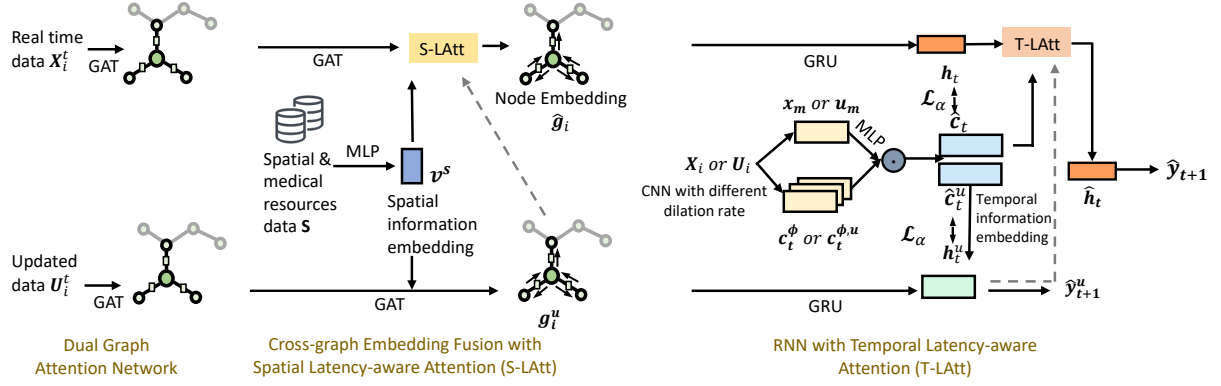
Figure 1: Our PopNet model consists of two graph attention networks to receive real-time data ($\mathcal{X}$) and updated data ($\mathcal{U}$) respectively. It uses spatial latency-aware attention (S-LAtt) to fuse two graphs and generate node embedding for each location. The spatial latency-aware attention is enriched by spatial information embedding (SIE) v^s learned using location-wise geographical and medical resource features. The node embeddings in two graph networks are fed into two GRUs respectively to extract temporal relations. PopNet also utilizes temporal latency-aware attention (T-LAtt) to fuse temporal embeddings. Similarly, T-LAtt is enriched by temporal information embeddings (TIE) $\hat{c}_t$ and c_t^u, which can adaptively embed the most informative multi-scale disease patterns to improve predictions. PopNet also aligns the hidden states of GRU $\hat{h}_t$, h_t^u with the learned TIE $\hat{c}_t$ and c_t^u respectively to achieve efficient update. Finally, PopNet will output predictions $\hat{y}_{t+1}$ using fused temporal embedding.

discussion on one location and omit the subscript node index i. The real-time and updated embeddings are fed into GRUs as in Eq. (11),

$$\begin{aligned} \mathbf{h}_t &= GRU(\hat{\mathbf{g}}_1, \hat{\mathbf{g}}_2, ..., \hat{\mathbf{g}}_t) \\ \mathbf{h}_t^u &= GRU_u(\mathbf{g}_1^u, \mathbf{g}_2^u, ..., \mathbf{g}_t^u) \end{aligned} \quad (11)$$

We use the hidden states of the GRU $\mathbf{h}_t$ and $\mathbf{h}_t^u$ as the temporal embeddings. Similarly, we design a temporal latency-aware attention mechanism (T-LAtt) to fuse two embeddings and deal with the latency between $\mathbf{h}_t$ and $\mathbf{h}_t^u$. As previously discussed, utilizing temporal-related data updating patterns may benefit the predictions. This involves extracting complex temporal patterns such as increasing or declining from different time scales. Besides, we also consider the updating patterns of comorbidities of target diseases. To extract and leverage these patterns, we enrich the T-LAtt with temporal information embeddings (TIE).

First, for TIE to extract temporal patterns from multiple timescales, we use dilated convolutional networks [24] with different dilation rates to extract temporal patterns from different time scales. Concretely, at each location, the input disease data sequence $\mathbf{X} = [\mathbf{x}^1, \mathbf{x}^2, ..., \mathbf{x}^t]$ is fed into the CNN as in Eq. (12),

$$\mathbf{c}_t^{\phi} = \mathbf{m}(L, \phi) * \mathbf{X}, \quad (12)$$

where $*$ denotes the convolution operation, $\mathbf{m}(L, \phi)$ is the 1D convolution filter with size L and dilation rate ϕ. The larger the ϕ is, the larger the filter's receptive field is, making the convolution filter extract temporal patterns from a broader time scale. In our experiments, we use a combination of different ϕ to extract patterns in different scales, from small to large. The feature maps are concatenated to get the final feature map vector $\mathbf{c}_t \in \mathbb{R}^C$, C denotes the number of convolution filters. Each value in $\mathbf{c}_t$ represents an extracted temporal feature.

Following previous CNN-based models [12, 14, 21], we also try to select the most informative patterns in $\mathbf{c}_t$ based on attention

weights. Here, we first use mean pooling over time dimension for $\mathbf{X}$ as $\mathbf{x}_m = MeanPool(\mathbf{X})$, $\mathbf{x}_m \in \mathbb{R}^F$. $\mathbf{x}_m$ can be regarded as a summary for F diseases. This vector is used to calculate the attention score for the temporal patterns as in Eq. (13),

$$\mathbf{a}_t^c = \sigma(MLP(\mathbf{x}_m)) \quad (13)$$

where σ denotes the sigmoid activation. We use the multi-layer perceptron to do the mapping $\mathbb{R}^F \to \mathbb{R}^C$, and the sigmoid activation to generate importance score between 0 and 1. The obtained score vector $\mathbf{a}_c$ is used to re-calibrate the feature map vector as in Eq. (14),

$$\hat{\mathbf{c}}_t = \mathbf{c}_t \odot \mathbf{a}_t^c \quad (14)$$

The obtained $\hat{\mathbf{c}}_t$ is the final temporal information embedding (TIE). We can also get the TIE for updated series $\hat{\mathbf{c}}_t^u$ in this way.

Similar to the spatial latency-aware attention, we use the TIE to enrich the attention and use time latency between current temporal embedding $\mathbf{h}_t$ and historical updated temporal embeddings $[\mathbf{h}_1^u, \mathbf{h}_2^u, ..., \mathbf{h}_t^u]$ to regularize the attention score as in Eq. (15),

$$\begin{aligned} e_{ti} &= f(\Delta t_{ti}) * \sigma(\mathbf{W}_a(\mathbf{W}_{h1}(\mathbf{h}_t|\hat{\mathbf{c}}_t) + \mathbf{W}_{h2}(\mathbf{h}_i^u|\hat{\mathbf{c}}_t^u))) \\ a_{ti} &= \frac{\exp(e_{ti})}{\sum_{j=1}^{t} \exp(e_{tj})} \end{aligned} \quad (15)$$

And the aggregated updated temporal embedding is given by Eq. (16),

$$\hat{\mathbf{h}}_t^u = \sum_{i=1}^{t} a_{ti} \mathbf{h}_i^u. \quad (16)$$

Finally, we concatenate the aggregated updated temporal embedding $\hat{\mathbf{h}}_t^u$, the original temporal embedding and the TIE to calculate the final temporal embedding as in Eq. (17),

$$\hat{\mathbf{h}}_t = (\mathbf{h}_t|\hat{\mathbf{h}}_t^u|\hat{\mathbf{c}}_t). \quad (17)$$

4.4 Efficient Iterative Training and Prediction

Model update challenge: In clinical practice, an online population-level disease prediction model needs routine updates when new data become available. For model updating with new data, it usually requires model retraining which is time-consuming as the data sequence becomes longer, or directly fine-tuning on the new data which discards historical patterns. Some also initialize a model using new data and the last hidden state of RNN trained using old data. However, this solution assumes there is no large time gap between the two datasets. Otherwise, capturing the continuous behavior of the system becomes more difficult for the model and leads to even worse performance [22].

Our Solution. PopNet introduces an alignment module to address this issue via providing a better initialization for the hidden states of the RNN on the new data without assuming the data continuity. This is achieved by learning a mapping function between the TIE $\hat{\mathbf{c}}_t$, $\mathbf{c}_t^u$ and the hidden states of RNN $\hat{\mathbf{h}}_t$ and $\mathbf{h}_t^u$ at each timestep respectively. When applying to new data, directly using the last hidden states is not optimal due to the new disease patterns may be different. But convolutional features can provide a better initialization for the RNN since they are not strictly sequential-dependent. Concretely, we first use a mapping function m_θ parameterized by θ to map the learned TIE to another latent space as in Eq. (18),

$$\hat{c}_t^m = m_\theta(\hat{\mathbf{c}}_t) \tag{18}$$

Then we calculate the probability distribution of the mapped TIE embedding and the current hidden state of the RNN $\mathbf{h}_t$ using softmax function as in Eq. (19),

$$p(\hat{c}_t^m) = softmax(\hat{c}_t^m); \quad q(\mathbf{h}_t) = softmax(\mathbf{h}_t) \tag{19}$$

Then we define the alignment loss function between $p(\hat{c}_t^m)$ and $q(\mathbf{h}_t)$ using Kullback-Leibler divergence as in Eq. (20),

$$\mathcal{L}_a = \sum_{i=1}^{n} p(\hat{c}_t^m) \log\left(\frac{p(\hat{c}_t^m)}{q(\mathbf{h}_t)}\right) \tag{20}$$

Note that here we use the Kullback-Leibler divergence since its asymmetric characteristic naturally fits our design: we expect the loss term can help the model learn a close estimation to $\mathbf{h}_t$ using $\hat{\mathbf{c}}_t$. Besides, since the dimensionality of two embeddings is large, using KL divergence instead of L1 or L2 distance can also help avoid the learned m_θ simply mapping the embeddings to random normal distributions. When applied to new data, the model will first calculate the TIE using the entire sequence and then use m_θ to provide the initialization for the hidden states of the RNN. The detailed algorithm is shown in Alg.1.

Finally, we use a two-layer perceptron to generate predictions via $\hat{y}_{t+1} = MLP(\hat{\mathbf{h}}_t)$. We also let the GRU_u to make predictions as $\hat{y}_{t+1}^u = MLP(\mathbf{h}_t^u)$. Note that $\hat{y}_{t+1}^u$ is the prediction for the day after the update point, so it may be earlier than current timestep t. However, we use this as an auxiliary task to better optimize the GRU and GAT. At testing time, only $\hat{y}_{t+1}$ is the model output. We use mean squared error as the loss function as in Eq. (21),

$$\mathcal{L}_r = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_{i+1} - y_{i+1})^2; \quad \mathcal{L}_u = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_{i+1}^u - y_{i+1}^u)^2 \tag{21}$$

We finally optimize the entire model using Eq. (22).

$$\mathcal{L} = \mathcal{L}_r + \mathcal{L}_u + \mathcal{L}_a \tag{22}$$

Algorithm 1 The PopNet model

Input:
  Real-time disease statistics $\mathcal{X}$, updated disease statistics $\mathcal{U}$, update intervals $\Delta t = [\Delta t_1, \Delta t_2, ..., \Delta t_T]$, prediction targets $Y = [Y_1, Y_2, ..., Y_T]$ and location graph $\mathcal{G}$.
Training:
  for $i = 1$ to T **do**
    Input $\mathcal{X}$ and $\mathcal{U}$ and get node embeddings using Eq. 4;
    Aggregate $\hat{\mathbf{g}}^u$ to $\hat{\mathbf{g}}$ using Eq. 9 and 10;
    Input spatial embeddings to GRU networks using Eq. 11;
    Calculate temporal information embeddings using Eq. 14;
    Aggregate $\mathbf{h}_i^u$ to $\mathbf{h}_i$ using Eq. 16 and 17;
    Make predictions for $i + 1$ timestep;
  end for
  Generate distributions using Eq. 18 and 19;
  Calculate KL divergence using Eq. 20;
  Optimize model parameters by minimizing loss in Eq. 22.
Iterative training:
  Calculate TIE for the input sequence using Eq. 14;
  Use learned m_θ to generate the initial hidden state of GRU;
  Repeat the normal training steps.

---

5 EXPERIMENT

We evaluate PopNet by comparing against several spatial-temporal prediction and disease prediction baselines using real-world datasets.

5.1 Experimental Setup

Data We extract disease statistics from patients' claims data in a real-world patient database from IQVIA. The patients' claims data are collected from 2952 counties in the US starting from 2018. We aggregate the ICD-10 codes in claims data into 21 categories, which include 17 diseases and 4 other codes (see detailed category descriptions in Appendix). We use week-level statistics. Since patients' claims data cannot be completed collected at one time, the disease statistics in a certain week will be updated over several weeks. We use the statistics collected in the first week as the real-time data, and we use the data collected from future weeks as the updated data. We conduct experiments to predict two diseases:

(1) **Respiratory Disease Dataset**: Respiratory diseases include ICD10 codes J00-J99, which are common and most of them are contagious. The number of cases is larger than most other diseases. Therefore, the claims data collection procedure is also longer, so that the disease statistics for one week will be fully collected in the following up to 13 weeks. We filter out locations that have very few cases (less than 100). Finally, we get 1,693 counties for respiratory diseases prediction.

(2) **Tumors Dataset**: Tumors include ICD10 codes C00-D49. Compared to respiratory diseases, the tumors have fewer cases, and the data update period is also shorter. Most statistics of one week can be fully collected in the following 7 weeks. We also

filter out locations with very few cases (less than 10), and we get 1,829 counties for tumor prediction.

In addition, we conduct experiments on other 15 diseases and a synthetic dataset generated based on real-world disease update distributions. The detailed statistics and the results can be found in the Appendix. The code and the synthetic dataset is publicly available in [1].

Baselines We evaluated PopNet against the following spatio-temporal prediction baselines: **SARIMAX**, **GRU** [6], **ASTGCN** [13], **GMAN** [39], **EvolveGCN**, **ColaGNN** [8] and **STAN** [9]. The detailed descriptions of baselines can be found in Appendix.

We also compare PopNet with the reduced version as the ablation study.

(1) **PopNet-LAtt** We reduce both S-LAtt and T-LAtt mechanisms from PopNet. PopNet-LAtt is essentially two branches that receive real-time and updated data independently, and the outputs of two networks are concatenated to make final predictions.
(2) **PopNet-SLAtt** We only reduce the spatial latency-aware attention from PopNet.
(3) **PopNet-TLAtt** We only reduce the temporal latency-aware attention from PopNet.
(4) **PopNet-$\mathcal{L}_\alpha$** We reduce the alignment module and the loss term $\mathcal{L}_\alpha$ from PopNet. Since this term is only related to the iterative training, it will only be evaluated in Q3 section. PopNet-$\mathcal{L}_\alpha$ simply uses normal initialization for RNN hidden states.

Metrics. Following the similar work [9, 33], we use the following regression metrics to evaluate all the models: The **root mean squared error (RMSE)**, **mean absolute error (MAE)** and the **mean absolute percentage error (MAPE)** measures the difference between predicted values and true values:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \qquad (23)$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| \qquad (24)$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{y}_i - y_i|}{y_i} \qquad (25)$$

All metrics are calculated after projecting the values into the real range.

Evaluation Strategy. We split the data into training, validation, and testing sets. The training set is 60 weeks, starting from January 2018 to March 2019. The validation set is 20 weeks, starting from April 2019 to July 2019. The testing set is 20 weeks, starting from August 2019 to December 2019. All the models use the same training data and are also evaluated and tested using the same sets. To be fair, the training, validation and test data for all models are the same. For baseline models, we concatenate the updated data at each time step with the real-time data for model inputs. Compared to all baselines, PopNet does not access any extra data. In order to reduce the variance caused by time shift, we save three model checkpoints

---

[1] https://github.com/v1xerunt/PopNet

with (1) lowest loss on the training set; (2) lowest MSE on the validation set; (3) last training epoch. We test each checkpoint on the test set and report the best performance.

Implementation Details. All methods are implemented in PyTorch [26] and trained on an Ubuntu 16.04 with 64GB memory and a Tesla V100 GPU. We use Adam optimizer [18] with a learning rate of 0.001 and trained for 200 epochs. The hyper-parameter settings of each baseline model can be found in the appendix.

5.2 Results

Q1. Performance on Disease Prediction

The prediction results on respiratory disease and tumors are listed in Table 1. We also conduct two-tailed student's T-test of MAE between PopNet and other baseline models to test the significance of performance improvement. The p-values are also in Table 1. PopNet outperforms all baseline methods on all metrics. On respiration disease dataset, PopNet achieves 47% lower RMSE, 23.2% lower MAE, and 29.4% lower MAPE, and on tumors dataset, PopNet achieves 29% lower RMSE, 24% lower MAE, and 13.2% lower MAPE, both compared with the best baseline ColaGNN.

Table 1: Disease Prediction Performance

Respiratory Diseases Prediction			
Model	**RMSE ($\times 10^5$)**	**MAE**	**MAPE**	**P-value**
SARIMAX	15.54	542.5	51.3	0.0
GRU	10.89	340.2	43.2	5e-20
GMAN	9.10	329.8	37.4	4e-15
ASTGCN	10.33	303.6	39.9	4e-13
EvolveGCN	9.85	312.4	36.9	8e-9
STAN	9.54	305.7	36.8	4e-9
ColaGNN	8.06	291.3	33.7	5e-5
PopNet-LAtt	9.82	311.6	39.2	4e-10
PopNet-TLAtt	6.32	271.3	29.9	5e-4
PopNet-SLAtt	8.85	297.5	31.3	7e-8
PopNet	**4.29**	**223.8**	**23.8**	-
Tumors Prediction			
Model	**RMSE ($\times 10^5$)**	**MAE**	**MAPE**	**P-value**
SARIMAX	21.41	426.0	69.8	0.0
GRU	16.20	313.3	60.7	0.0
GMAN	13.12	315.7	53.4	7e-25
ASTGCN	15.59	332.5	56.2	0.0
EvolveGCN	8.94	269.0	50.4	5e-11
STAN	6.56	215.8	48.7	7e-9
ColaGNN	4.75	172.5	42.3	9e-5
PopNet-LAtt	8.92	283.4	51.5	4e-10
PopNet-TLAtt	3.25	142.9	38.1	3e-4
PopNet-SLAtt	4.50	165.8	40.1	5e-5
PopNet	**2.90**	**131.6**	**36.7**	-

In addition, we evaluate the performance on other disease categories (grouped by ICD code). We report the detailed test MAE in Appendix, and show the test MAPE of PopNet against the best baseline in Fig. 2. The results show that PopNet outperforms all
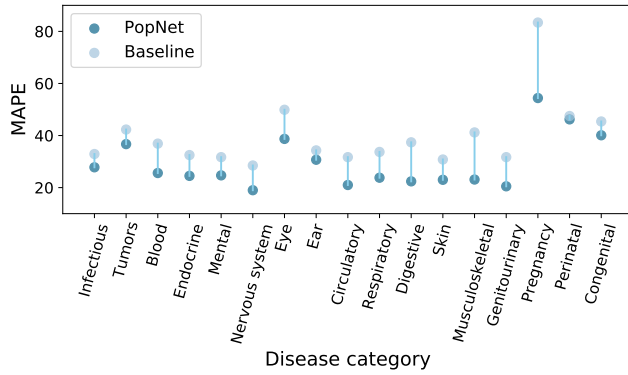
Figure 2: Comparison of the test MAPE between PopNet and the best baseline on all disease code categories

baseline models for all disease categories, which indicates the potential broader utility of PopNet. It is worth noting that for some disease codes, PopNet achieves much better performance than baseline models, for example, musculoskeletal disease or pregnancy prediction. These data often receive more frequent updates due to the large number of cases or the particularity. Therefore, PopNet achieves better performance since it can better extract and utilize update patterns.

Q2. Performance at Different Locations

This section further explores the performance of PopNet on different locations We report the number of locations that each model has the best performance in Table. 2. Here 'Others' sums up the results of SARIMAX, GMAN, GRU, ASTGCN and EvolveGCN.

Table 2: # of locations where each model performs the best.

Respiratory Diseases			
Model	# of Locations	% of Locations	Mean $\Delta MAPE$
STAN	42	2.5%	3.5%
ColaGNN	33	1.9%	2.7%
Others	6	0.4%	4.1%
PopNet	**1612**	**95.2%**	**12.4%**
Tumors			
Model	# of Locations	% of Locations	Mean $\Delta MAPE$
STAN	40	2.2%	8.5%
ColaGNN	122	6.7%	4.6%
Others	22	1.2%	5.5%
PopNet	**1645**	**89.9%**	**10.7%**

It is easy to see PopNet achieves the best performance on over 90% of locations for both tasks. For respiratory diseases, PopNet has 12.4% improved MAPE on average than the best baseline. For tumors prediction, PopNet achieves the best performance on 1645 locations, and the average MAPE improvement is 10.7%. Even for the locations that baselines perform the best, the MAPE gap is small.

Q3. Performance with Iterative Training

To evaluate whether the iterative training of PopNet improves the efficiency of model updating, we simulate the following deployment setting: train on the original dataset (week 1-50), and deploy into practice (week 50-80). Then refresh the model using newly collected data during the deployment phase (week 60-80). Finally, we re-deploy the model to test the performance (week 80-100). The entire splitting process is shown in Fig. 3.
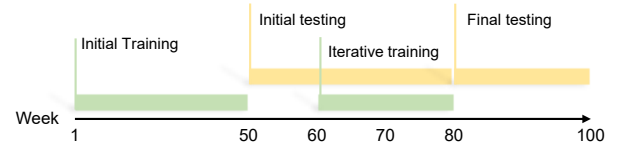

Figure 3: Data splitting process to simulate iterative training

We report the performance on the final testing phase in Table 3. PopNet outperforms baselines, achieving 39% lower RMSE, 34% lower MAE, and 25.6% lower MAPE on respiratory disease prediction. and 70% lower RMSE and 49% lower MAE on tumor prediction, compared with the best baseline. Compared with the reduced model PopNet-$\mathcal{L}_\alpha$, we can see the alignment loss can indeed help improve the predictive performance by providing RNN with a better initialization.

Table 3: Prediction performance with iterative traing

Respiratory diseases				
Model	**RMSE ($\times 10^5$)**	**MAE**	**MAPE**	**P-value**
GRU	14.77	495.4	52.5	0.0
GMAN	14.17	450.5	47.3	0.0
ASTGCN	12.45	432.7	46.2	3e-20
EvolveGCN	12.18	429.7	43.1	1e-18
STAN	13.90	448.5	47.0	0.0
ColaGNN	11.82	416.1	39.8	9e-17
PopNet-$\mathcal{L}_\alpha$	11.43	355.3	33.5	5e-10
PopNet	**7.23**	**275.6**	**29.6**	-
Tumors				
Model	**RMSE ($\times 10^5$)**	**MAE**	**MAPE**	**P-value**
GRU	18.95	397.6	52.9	8e-23
GMAN	18.99	401.6	53.2	0.0
ASTGCN	19.72	401.3	55.6	0.0
EvolveGCN	17.26	385.1	50.0	0.0
STAN	10.21	304.7	40.8	6e-15
ColaGNN	8.75	264.4	35.3	1e-8
PopNet-$\mathcal{L}_\alpha$	4.92	182.7	35.8	3e-4
PopNet	**2.60**	**135.4**	**34.7**	-

To further evaluate how $\mathcal{L}_\alpha$ can help improve the iterative training, in Fig. 4, we compare the iterative training results with the models that are trained on the entire sequence (results in Table 1). Compared with the results reported in Table 3 (yellow bars), the results in Table 1 (green bars) are obtained using the same test set
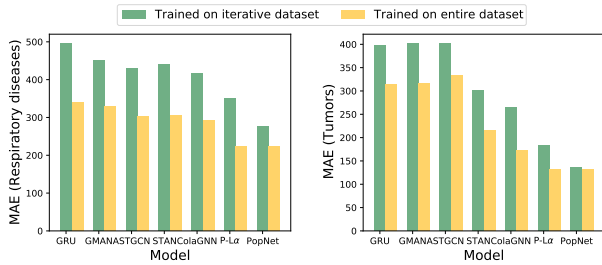
Figure 4: Prediction MAE for models trained on entire dataset and iterative dataset

but trained on the entire historical sequence (week 1-80). The P-$\mathcal{L}_\alpha$ indicates the reduced model PopNet-$\mathcal{L}_\alpha$.

The figure shows that the models trained on the entire dataset can achieve lower MAE because the model can access the entire historical data, which makes the models can extract and utilize more historical patterns. However, compared to all baselines and the reduced model, the performance gap of PopNet is much smaller. Compared to the model trained under the iterative training setting, PopNet trained on entire sequences achieves 23% higher MAE on respiratory diseases prediction and only 3% higher MAE on tumors prediction. In comparison, the best baseline model trained on entire sequences achieves 41% higher MAE on respiratory diseases prediction and 53% higher MAE on tumors prediction. This indicates that by aligning the TIE and hidden states of RNN, the model can indeed utilize historical patterns without accessing the original sequences.

Since the entire sequence is four times longer than the iterative training data, training on the entire dataset could be costly for time and memory. For example, for tumor disease, training a regular spatial-temporal model on the entire dataset generally requires more than 12 GB memory and the average training time is about 5 seconds per epoch. But it only requires less than 4 GB memory and 2 seconds per epoch to train the same model on the iterative dataset. PopNet can achieve almost equivalent performance using just iterative training data, which can be useful for real-world applications and efficient for long-sequence data.

Q4. Performance with Longer Prediction Window

Long-term prediction is also significant for disease prediction in practice. In this work, we also explore the capability of PopNet for long-term disease prediction. We change the output size to make PopNet and other baseline models predict future 5 weeks. We report the performance and the p-value of MAE in Table 4.

The results show that the MAE rises as the prediction window increases since it becomes more difficult to predict longer future trends. However, PopNet still has the lowest MAE increase ratio.

For respiratory diseases, the MAE of PopNet increases 14% as the prediction window length increases from 1 to 5, while the baseline model STAN increases 24% and ColaGNN increases 31%. For tumors, the MAE of STAN and ASTGCN increase 11% and 24%, respectively, while PopNet only increases 6%. The results show that PopNet can consistently outperform all other baseline models under different lengths of prediction window. A longer prediction window has less effect on the predictive performance of PopNet. To

Table 4: Long-term prediction (window = 5 weeks)

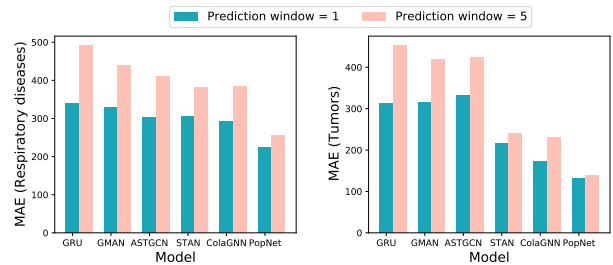| Model	RMSE ($\times10^5$)	MAE	MAPE	P-value
Respiratory Diseases				
GRU	16.90	491.4	34.7	7e-21
GMAN	17.76	440.3	35.2	6e-23
ASTGCN	15.23	410.3	33.4	2e-15
STAN	9.30	380.5	31.5	4e-8
ColaGNN	9.78	384.1	31.7	8e-8
PopNet	**5.71**	**255.4**	**27.8**	-
Tumors				
Model	RMSE ($\times10^5$)	MAE	MAPE	P-value
GRU	21.56	452.6	38.8	0.0
GMAN	19.35	420.1	35.9	5e-20
ASTGCN	19.78	425.1	36.0	6e-23
STAN	6.72	240.3	34.1	3e-9
ColaGNN	5.37	231.9	33.5	5e-7
PopNet	**2.79**	**138.4**	**31.5**	-

Figure 5: Prediction MAE under prediction window 1 and 5

illustrate more clearly, we draw the performance gap of all baseline models with different prediction windows in Fig. 5. The figure shows that compared to baseline models, PopNet can achieve a significantly smaller error gap as the prediction window increases on two diseases. This indicates that PopNet is also suitable for long-term prediction tasks.

6 CONCLUSION

In this work, we propose PopNet for real-time population-level disease prediction with considering data latency. PopNet uses two separate systems to model real-time and updated disease statistics data, and then adaptively fuses the two systems using both spatial and temporal latency-aware cross-graph attention. We augment the latency-aware attention with spatial and temporal information embeddings to adaptively extract and utilize geographical and temporal progression features. We also conducted extensive experiments across multiple real-world claims datasets. PopNet outperforms leading spatial-temporal models in all metrics and shows the promising utility and efficacy in population-level disease prediction. In future works, we will use more flexible way to generate better location graph instead of using hard defined edge weights, which is the major limitation of this work.

7 ACKNOWLEDGMENTS

REFERENCES

[1] Defu Cao, Yujing Wang, Juanyong Duan, Ce Zhang, Xia Zhu, Conguri Huang, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, et al. 2021. Spectral temporal graph neural network for multivariate time-series forecasting. *arXiv preprint arXiv:2103.07719* (2021).

[2] Edward Choi, Mohammad Taha Bahadori, Andy Schuetz, Walter F Stewart, and Jimeng Sun. 2016. Doctor ai: Predicting clinical events via recurrent neural networks. In *Machine learning for healthcare conference*. PMLR, 301–318.

[3] Edward Choi, Mohammad Taha Bahadori, Le Song, Walter F Stewart, and Jimeng Sun. 2017. GRAM: graph-based attention model for healthcare representation learning. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 787–795.

[4] Edward Choi, Andy Schuetz, Walter F Stewart, and Jimeng Sun. 2017. Using recurrent neural network models for early detection of heart failure onset. *Journal of the American Medical Informatics Association* 24, 2 (2017), 361–370.

[5] Edward Choi, Cao Xiao, Walter F Stewart, and Jimeng Sun. 2018. Mime: Multilevel medical embedding of electronic health records for predictive healthcare. *arXiv preprint arXiv:1810.09593* (2018).

[6] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[7] Limin X Clegg, Eric J Feuer, Douglas N Midthune, Michael P Fay, and Benjamin F Hankey. 2002. Impact of reporting delay and reporting error on cancer incidence rates and trends. *Journal of the National Cancer Institute* 94, 20 (2002), 1537–1545.

[8] Songgaojun Deng, Shusen Wang, Huzefa Rangwala, Lijing Wang, and Yue Ning. 2020. Cola-GNN: Cross-location Attention based Graph Neural Networks for Long-term ILI Prediction. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 245–254.

[9] Junyi Gao, Rakshith Sharma, Cheng Qian, Lucas M Glass, Jeffrey Spaeder, Justin Romberg, Jimeng Sun, and Cao Xiao. 2020. Stan: Spatio-temporal attention network for pandemic prediction using real world evidence. *arXiv preprint arXiv:2008.04215* (2020).

[10] Jingyue Gao, Xiting Wang, Yasha Wang, Zhao Yang, Junyi Gao, Jiangtao Wang, Wen Tang, and Xing Xie. 2019. Camp: Co-attention memory networks for diagnosis prediction in healthcare. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1036–1041.

[11] Junyi Gao, Cao Xiao, Lucas M Glass, and Jimeng Sun. 2020. Dr. Agent: Clinical predictive model via mimicked second opinions. *Journal of the American Medical Informatics Association* 27, 7 (2020), 1084–1091.

[12] Junyi Gao, Cao Xiao, Yasha Wang, Wen Tang, Lucas M Glass, and Jimeng Sun. 2020. StageNet: Stage-Aware Neural Networks for Health Risk Prediction. In *Proceedings of The Web Conference 2020*. 530–540.

[13] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. 2019. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 922–929.

[14] Jie Hu, Li Shen, and Gang Sun. 2018. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7132–7141.

[15] Rongzhou Huang, Chuyin Huang, Yubao Liu, Genan Dai, and Weiyang Kong. 2020. LSGCN: Long Short-Term Traffic Prediction with Graph Convolutional Networks.. In *IJCAI*. 2355–2361.

[16] Amol Kapoor, Xue Ben, Luyang Liu, Bryan Perozzi, Matt Barnes, Martin Blais, and Shawn O'Banion. 2020. Examining covid-19 forecasting using spatio-temporal graph neural networks. *arXiv preprint arXiv:2007.03113* (2020).

[17] William Ogilvy Kermack and Anderson G McKendrick. 1927. A contribution to the mathematical theory of epidemics. *Proceedings of the royal society of london. Series A, Containing papers of a mathematical and physical character* 115, 772 (1927), 700–721.

[18] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[19] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1269–1278.

[20] Fenglong Ma, Radha Chitta, Jing Zhou, Quanzeng You, Tong Sun, and Jing Gao. 2017. Dipole: Diagnosis prediction in healthcare via attention-based bidirectional recurrent neural networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 1903–1911.

[21] Liantao Ma, Junyi Gao, Yasha Wang, Chaohe Zhang, Jiangtao Wang, Wenjie Ruan, Wen Tang, Xin Gao, and Xinyu Ma. 2020. Adacare: Explainable clinical health status representation learning via scale-adaptive feature extraction and recalibration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 825–832.

[22] Nima Mohajerin and Steven L Waslander. 2017. State initialization for recurrent neural network modeling of time-series data. In *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2330–2337.

[23] Kris A Murray, Nicholas Preston, Toph Allen, Carlos Zambrana-Torrelio, Parviez R Hosseini, and Peter Daszak. 2015. Global biogeography of human infectious diseases. *Proceedings of the National Academy of Sciences* 112, 41 (2015), 12746–12751.

[24] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499* (2016).

[25] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Schardl, and Charles Leiserson. 2020. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 5363–5370.

[26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703* (2019).

[27] Sen Pei and Jeffrey Shaman. 2020. Initial Simulation of SARS-CoV2 Spread and Intervention Effects in the Continental US. *medRxiv* (2020).

[28] Zhaozhi Qian, Ahmed M Alaa, and Mihaela van der Schaar. 2020. CPAS: the UK's national machine learning-based hospital capacity planning system for COVID-19. *Mach. Learn.* (Nov. 2020), 1–21.

[29] Constantinos I Siettos and Lucia Russo. 2013. Mathematical modeling of infectious disease dynamics. *Virulence* 4, 4 (2013), 295–306.

[30] Ruiguang Song and Timothy A Green. 2012. An improved approach to accounting for reporting delay in case surveillance systems. *JP J Biostat* 7, 1 (2012), 1–14.

[31] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[32] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. 2019. Dyrep: Learning representations over dynamic graphs. In *International conference on learning representations*.

[33] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[34] Richard J Verrall and Mario V Wüthrich. 2016. Understanding reporting delay in general insurance. *Risks* 4, 3 (2016), 25.

[35] Yanbang Wang, Yen-Yu Chang, Yunyu Liu, Jure Leskovec, and Pan Li. 2021. Inductive Representation Learning in Temporal Networks via Causal Anonymous Walks. *arXiv preprint arXiv:2101.05974* (2021).

[36] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. 2020. Inductive representation learning on temporal graphs. *arXiv preprint arXiv:2002.07962* (2020).

[37] Zifeng Yang, Zhiqi Zeng, Ke Wang, Sook-San Wong, Wenhua Liang, Mark Zanin, Peng Liu, Xudong Cao, Zhongqiang Gao, Zhitong Mai, et al. 2020. Modified SEIR and AI prediction of the epidemics trend of COVID-19 in China under public health interventions. *Journal of Thoracic Disease* 12, 3 (2020), 165.

[38] Huaxiu Yao, Xianfeng Tang, Hua Wei, Guanjie Zheng, Yanwei Yu, and Zhenhui Li. 2018. Modeling spatial-temporal dynamics for traffic prediction. *arXiv preprint arXiv:1803.01254* (2018).

[39] Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi. 2020. Gman: A graph multi-attention network for traffic prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 1234–1241.

A   DATASET DETAILS

A.1   Real-world dataset

In this section, we report the basic statistics and disease categories in the real-world claims dataset. The patients' claims data are collected from 2952 counties in the US starting from 2018. We aggregate the codes into 17 disease categories (A00-Q99) and 4 other categories (R00-Z99) according to the ICD-10 coding. The detailed disease category is reported in Table 5.

ICD code	Category description	Avg. cases
A00-B99	Certain infectious and parasitic diseases	314.9
B00-D49	Tumors	966.1
D50-D89	Diseases of the blood, blood-forming organs and immune mechanism	344.8
E00-E89	Endocrine, nutritional and metabolic diseases	1508.8
F01-F99	Mental, Behavioral and mental disorders	1470.1
G00-G99	Diseases of the nervous system	744.2
H00-H59	Diseases of the eye and adnexa	385.8
H60-H95	Diseases of the ear and mastoid process	192.2
I00-I99	Diseases of the circulatory system	1717.2
J00-J99	Diseases of the respiratory system	1774.7
K00-K95	Diseases of the digestive system	653.1
L00-L99	Diseases of the skin and subcutaneous tissue	496.9
M00-M99	Diseases of the musculoskeletal system and connective tissue	2226.7
N00-N99	Diseases of the genitourinary system	897.0
O00-O99	Pregnancy, childbirth and the puerperium	152.6
P00-P96	Certain conditions originating in the perinatal period	45.9
Q00-Q99	Congenital malformations, deformations and chromosomal abnormalities	80.2
R00-R99	Symptoms, signs and abnormal clinical and laboratory findings	2480.2
S00-T88	Injury, poisoning and certain other consequences of external causes	683.3
V00-Y99	External causes of morbidity	12.4
Z00-Z99	Factors influencing health status and contact with health services	2653.2

Table 5: ICD codes and disease category descriptions

Due to space limitations, we only report the detailed data statistics of two diseases (i.e., tumors and respiratory diseases) reported in the main text. The statistics are shown in Table 6. The spatial features include populations, number of hospitals, number of ICU beds, longitude, latitude and annual income.

A.2   Synthetic dataset

We construct a synthetic dataset from a real-world disease dataset. We first randomly aggregate the data in the real-world dataset from different locations to generate data sequences for the real-time data and prediction targets. For each location, We randomly aggregate data from 1-5 neighboring locations. Then for each timestep, we use up-sampling and down-sampling to aggregate data from 1-3 continuous timesteps while keeping the length of data does not change. Then we add random Gaussian noise ($\mu = 0, \sigma = 1$) to the aggregated data. For the updated data, we assume all locations are updated at regular intervals for the sake of simplicity. We use the

Respiratory diseases	
# of locations	1693
# of features	21
# of edges	4521
Avg. # of edges per nodes	2.67
# of sequence length	63
Avg. # of target cases	1774.7
Avg. # of update frequencies	9.3
Tumors	
# of locations	1829
# of features	22
# of edges	4884
Avg. # of edges per nodes	2.67
# of sequence length	63
Avg. # of target cases	966.1
Avg. # of update frequencies	3.3

Table 6: Data statistics for respiratory disease and tumors dataset

same strategy as the real-time data to generate the updated data. The basic statistics of the synthetic dataset are shown in Table 7.

# of locations	1015
# of features	22
# of edges	6410
Avg. # of edges per nodes	6.32
# of sequence length	63
Avg. # of target cases	1098.7
Avg. # of update frequencies	5.4

Table 7: Synthetic data statistics

B   PREDICTION PERFORMANCE FOR ALL DISEASE CATEGORIES

In this section, we report the predictive performance of `PopNet` for all disease categories. Due to space limitations, we only select two baseline models (i.e., STAN and ColaGNN) to compare, which have generally better performance. For some disease categories such as *Certain conditions originating in the perinatal period*, some locations have no case at most timesteps, so these disease datasets have fewer locations to predict. We report the test MAE in Table 8. `PopNet` can outperform two baselines for all disease code categories.

C   PERFORMANCE ON SYNTHETIC DATASET

We also conducted experiments on the artificially generated synthetic dataset, and report results in Table 9. From the results, `PopNet` outperforms all baselines with a $p = 0.001$ significance level. Compared with the best baseline ColaGNN, `PopNet` has 19.5% lower RMSE, 15.5% lower MAE, and 4% lower MAPE. The SARIMAX model does not perform well on the synthetic dataset since autoregression models are difficult to fit random noises in the data.

ICD code	# of locations	STAN	ColaGNN	PopNet
A00–B99	1,654	173.6	152.7	**53.7**
D50–D89	1,545	210.5	132.2	**70.1**
E00–E89	1,793	310.0	285.4	**268.4**
F01–F99	1,675	325.4	301.3	**287.4**
G00–G99	1,733	225.9	269.1	**124.1**
H00–H59	1,530	129.3	150.7	**113.4**
H60–H95	1,551	97.1	85.2	**43.9**
I00–I99	1,642	439.5	405.5	**321.3**
K00–K95	1,557	280.7	350.6	**124.8**
L00–L99	1,424	192.9	245.8	**122.9**
M00–M99	1,789	371.2	369.7	**355.5**
N00–N99	1,673	288.9	252.0	**163.8**
O00–O99	1,512	170.2	95.3	**31.4**
P00–P96	978	115.8	81.9	**19.8**
Q00–Q99	1,340	157.4	109.7	**22.7**

Table 8: Test MAE for all disease categories

Table 9: Prediction performance on synthetic dataset

Model	RMSE ($\times 10^5$)	MAE	MAPE	P-value
SARIMAX	6.42	252.4	60.0	0.0
GRU	2.37	130.3	36.7	1e-3
GMAN	3.52	142.4	41.3	5e-4
ASTGCN	2.56	128.4	37.2	8e-4
STAN	2.33	122.5	35.7	2e-3
ColaGNN	2.21	115.9	34.2	3e-3
PopNet-LAtt	2.55	135.2	37.2	9e-3
PopNet-TLAtt	2.08	106.9	33.8	1e-3
PopNet-SLAtt	2.15	113.8	34.0	2e-3
PopNet	**1.78**	**97.9**	**33.0**	-

D IMPLEMENTATION DETAILS

All methods are implemented in PyTorch [26] and trained on an Ubuntu 16.04 with 64GB memory and a Tesla V100 GPU. We use Adam optimizer [18] with a learning rate of 0.001 and trained for 200 epochs.

For hyper-parameter settings of each baseline model, our principle is as follows: For some hyper-parameter, we will use the recommended setting if available in the original paper. Otherwise, we determine its value by grid search on the validation set.

- **SARIMAX** stands for seasonal autoregressive integrated moving average, which is a popular time series prediction model. SARIMAX considers seasonal influence with exogenous variables, making it more suitable for our disease prediction task. We use grid-search to determine the hyperparameters of the model at each location.
- **GRU**. We use GRU to conduct temporal prediction without considering the spatial relationships. GRU model cannot utilize spatial relationships and locations are regarded as independent samples to train the GRU model. The hidden units of the GRU cell are set to 128.

- **GMAN** is a recently published spatial-temporal prediction model for traffic prediction. It uses an encoder-decoder structure with spatial-temporal attention to predict future traffic status. The number of attention blocks is set to 3, the dimensionality of each attention head is set to 64, and the number of attention head is 4.
- **ASTGCN** is a recently published spatial-temporal prediction model for traffic prediction. It applies additional convolutional layers and attention mechanisms on GCN. The number of convolutional kernels is set to 64, and the kernel size is set to 3.
- **EvolveGCN** is a general spatial-temporal prediction model. It adapts the graph convolutional network (GCN) model along the temporal dimension without resorting to node embeddings and uses an RNN to evolve the GCN parameters. The hidden units of the GRU cell are set to 128, and the dimensionality of GNN is set to 64.
- **STAN** is a hybrid deep learning and epidemiology spatial-temporal model for epidemic and pandemic prediction. STAN also constructs a location graph based on geographic similarity and uses graph attention network and RNN to predict future cases. Since we do not constraint our prediction target is an infectious disease, we remove the disease transmission dynamics regularization in STAN. The dimensionality of GAT is set to 64 for respiratory diseases prediction and tumors prediction, 128 for the synthetic dataset. The number of hidden units of GRU cell is set to 128.
- **ColaGNN** is a spatial-temporal pandemics prediction model, which uses a location graph to extract spatial relationships for predicting pandemics. The number of hidden units of GRU cell is set to 128. The number of convolutional kernels is set to 64, and the kernel size is set to 3.
- **PopNet**. The α, β, γ is set to 0.35, 0.37, 30. We set the kernel size of convolutional layers to 16 and kernel size to 3. We use a set of dilation rate $\phi = [1, 3, 5]$. The dimensionality of the GAT layer and attention head is set to 32. We use 2 attention heads for respiratory diseases prediction and tumors prediction, 1 for the synthetic dataset. The hidden units of GRU are set to 256. The dimensionality of MLP is set to 128.

We also use a dropout layer [31] before the output layer to prevent overfitting. The dropout rate is set to 0.5.