CS245 Paper Survey Project 5

Team - Big Data Group
(Jason Kao, Sripath Mishra, Boya Ouyang, Vishnu Devarakonda)
Team - Ideas
(Panqiu Tang, Huiling Huang, Chenyang Wang, Haochen Yin, Yijing Zhou, Danfeng Guo)

University of California, Los Angeles

1 Background

Graph based deep learning has been widely used in research. Recent works on graph/hypergraph based deep learning methods include Graphical Neutral Network (GNN) [1] which designs graph based neural networks with semi-supervised learning. HGNN [2], the first hypergraph neural network model, is built based on graph neural networks, and allows dynamic modifications on the graph structure, which had been a major drawback for graph neural networks.

Spatial graph convolution is a category of semi-supervised learning on graphs in contrast to spectral graph convolution, utilizing spatial samplers and aggregators to generate neighborhood feature embedding. MoNet[3] introduces a spatial convolution deep learning framework on non-Euclidean domains. GraphSAGE [4] defines different spatial aggregator and sampler using LSTM-GAT[5] introduces the self-attention mechanism to generate attention coefficients between nodes.

COVID-19 (COV19) is caused by a coronavirus called SARS-CoV-2. It is an ongoing pandemic starting in Dec, 2019. COV19 is highly infectious mainly spreads through the air. Up to now, it has caused over a million deaths. Let alone the fact that it has brought uncountable loss to economy. To prevent COV19 from further spreading, it is crucial to identify infected carriers and isolate them. However, as the epidemic spreads, the demand for COV19 tests outgrows the availability. Given the limited supply of test reagents, only a fraction of capacity vendors can be tested. Hence, it becomes essential to select the maximum probable epidemic carriers. This raises a prime query: How to rank the test targets prioritize those with the most change of geting into contact with virus. This prioritization problem is an essential instance of a list of challenges: learning to govern diffusive techniques over networks through nodal interventions. In these

types of cases, the dynamics of the device can be steered by the usage of interventions that adjust the states of an (incredibly) small wide variety of nodes.

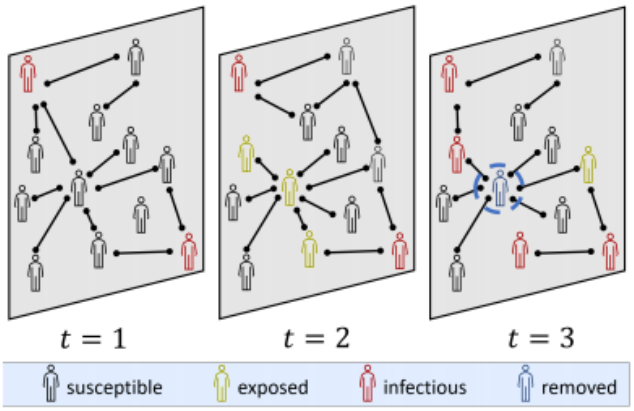

Fig. 1. A viral infection process on a graph and an intervention aimed to stop its spread. Here, graph nodes represent people and edges represent interactions. [6]

For instance, infected people can be requested to self-quarantine, preventing the spread of sickness, at-chance computer systems may be patched through safety updates, and customers can be decided on and be exposed to new records to steer their opinion. The problem of controlling the dynamics of a machine the use of localized interventions may be very difficult, and for numerous reasons. First, it requires to make a selection in constantly converting surroundings with complicated dependencies. Second, to clear up the trouble one must determine the capability downstream ripple impact for any particular node that becomes inflamed, and stability it with the chance that the node indeed becomes inflamed. For the ripple effect we would have to uncover the patient zero. This is difficult to uncover this without GNNs as we have to backtrack from a infected graph at time t. It is observed [7] that GNNs can identify Patient 0 close to the theoretical bound on accuracy, without explicit input of dynamics or its parameters. In addition, GNN is over 100 times faster than classic methods for inference on arbitrary graph topologies. This allows better forecasting and risk management.

Finally, models have to manage noise and partial observability. Many approaches have been developed to predict pattern of COV19.The present-day work combines 3 research fields: dynamic approaches on temporal

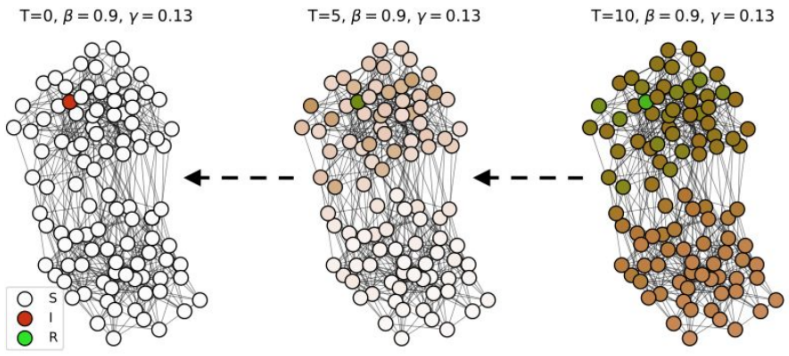

Fig. 2. visualization the patient zero problem [7]

graphs, deep graph learning, and reinforcement getting to know. Combining these three into a cohesive model raises several new demanding situations. Most importantly, the model ought to learn to cope with two styles of dynamics: learn to deduce the capability hazard of no longer getting rid of a node, and learn to expect the possibility that a node becomes infected. Popular approaches to model COV19 or epidemiological modeling include the mechanistic approach, which models transmission dynamics on the population level, and the time-series based approach, which applies curve-fitting or deep learning on time-series data. However, these models assume data dependencies local to the specific region. Spatio-temporal graphs model time and space connectivity between nodes, and have received recent success in a spatio-temporal traffic graph [8] and spatio-temporal influenze forecasting [9]. The latter model leverages Recurrent Neural Network (RNN) and attention matrices to learn hidden states for locations, which is followed by a GNN to predict results. Graph neural networks have been especially useful in this case since they are able to utilize adjacent node information to inform the future hidden states for a node. However, one downside of such models is that they can only predict known trends from the input data.

In this survey paper, we examine several models applicable or have been applied to COV19 pandemic prediction. We move from the basic GNN to its derivatives.

2 Methodologies

2.1 Introduction

GNNs are deep neural networks which can tackle graph-based facts. GNNs was shown beneficial for fixing a ramification of duties such as social community evaluation and molecule property prediction. Perhaps the maximum popular GNN models are Message Passing Neural Networks (MPNN), which function by time and again updating the feature vector of each node by aggregating records from its neighborhood. Several works integrate RNNs with GNNs to study temporal graph records. [**?**] used graph-structured RNN for coarse spatial prediction of epidemic spread. Those works version the epidemic spread and do not intervene with the diffusive technique. More usually, several latest research address a setup wherein each nodes and edges range through the years, with applications in social network evaluation and other fields. Further statistics can be observed in.

Ranking on graphs The problem of ranking on graphs is a essential problem in Computer Science, wherein the assignment is to rank the nodes of a given graph in line with a few criteria. It has various applications together with internet page ranking and expertise graph search.

Reinforcement learning of and graphs Recently, a surge of work combining Reinforcement Learning (RL) and graphs emerged. These works may be split into two principal classes: leveraging graph shape for general RL problems and making use of reinforcement getting to know strategies for graph problems. Our paintings falls into the latter. An vital line of labor uses Reinforcement Learning to resolve NP-difficult combinatorial optimization troubles described on a graph. Another commonplace software is using RL for path searching in a information graph. Reinforcement mastering become also shown in a few other graph issues, consisting of chemical reaction prediction.

Dynamic process on graphs Modeling diffusive tactics is an active studies subject. Key fashions which include Susceptible-Infected-Removed (SIR) and Susceptible-Infected-Susceptible (SIS) to the current Susceptible-Exposed-Infectious-Removed (SEIR) COVID-19 epidemic model have established beneficial in modeling the spread of contagions. The application of these models is prolonged, and varies from early epidemic detection, to steer maximization and network security. Recently, researchers have also used deep learning to forecast epidemic increase. The hassle of node manipulation (e.G., vaccination) for controlling epidemic procedures on graphs was intensively studied. This hassle is frequently addressed in the

setup of the fire-fighter trouble and its extensions. Common tactics consist of growing centrality measures designed to focus on bottleneck nodes or the usage of spectral techniques for allocating resources. An alternative line of studies advanced heuristics for the equal challenge

2.2 Graph Neural Networks

Referring to [10], graphs can be modeled as a set of nodes where there are edges representing relationship between nodes. Given the properties of Graphs, they have been widely used in the areas like social networks, physical systems, and protein-protein interaction networks. Graph Neural Networks are deep learning-based methods to tackle this kind of non-Euclidean data structure, and with its high interpret ability, GNNs have been widely used to perform node classification, link prediction, and clustering. Let G = (V, E) denote a graph where V = $\{v_1, v_2, ..., v_n\}$, n is the number of nodes. Let the feature vector of node v_i be X_i. We denote the adjacency matrix of a graph as A $\in R^{n\times n}$ with $A_{ij} = 1$ if (v_i, v_j) $\in$ E and 0 otherwise. The degree matrix associated with A is defined as D = diag(d_1, d_2, ..., d_n) where $d_i = \sum_{j=1}^{n} A_{ij}$. GNNs learn the representations of nodes and graphs by the graph structure and the node features. Modern GNNs follow a neighborhood aggregation strategy, where the representation of a node is updated by aggregating the representation of its neighbors. Let $h_i^{(k)}$ denote the representation of v_i at k-th layer, where $h_i^{(0)} = X_i$. The general framework of GNN can be described as follows, where

$$h_i^{(k)} = AGGREGATE^{(k)}(h_i^{(k-1)}, h_j^{(k)} \in N(v_i)), \tag{1}$$

where N(v_i) is the set of nodes adjacent to v_i. Different graph neural networks can be obtained by choosing different AGGREGATE functions.

The development of GNNs comes from different aspects. First of all, we all know that Convolutional Neural Networks (CNNs) can deal with images and texts, which can be modeled as 2D grids and 1D grids. These can be thought as graphs in Euclidean domain, however when it comes to the non-Euclidean domain, CNNs cannot provide satisfactory results. What's more, when dealing with graph learning, one common concept is Graph Embedding. The basic idea is that to learn to put together nodes, edges and sub-graphs into lower dimensional vectors. However, when dealing with dynamic graphs or generalization to new graphs, this method cannot work due to its lack of ability of generalization. And compare to standard neural networks like CNN and RNN, GNN has

several advantages that makes it worth investigating. CNNs and RNNs require nodes to be inputted by a specific order, and obviously there is not a pre-defined order of nodes in a graph, thus to solve this problem with CNN and RNN, every possible order of nodes need to be visited thus requiring huge computing power. GNN deals with this problem in a way that it propagates on each node respectively, thus the input order of nodes does not matter. Moreover, standard CNNs treated the edges, which give the information of dependency between nodes, as one of many features, while GNNs can do propagation guided by the graph structure not by just the edges. Thus, we can see that when dealing with graph problems, using GNNs can be a better choice than using standard CNNs, thus research on GNNs are worth to conduct.

The core idea is to learn a function map f, through which a node v_i in the map can aggregate its feature x_i and its neighbor feature $x_j(j \in N(x_i)$ to generate a new representation of the node . GNN is the basis of many complex graph neural network models, including auto-encoders, generative models, and spatio-temporal networks.

GNN methods can be divided into two major categories, spectral-based and spatio-based. Spectrum-based methods introduce filters from the perspective of graph signal processing to define graph convolution, where the graph convolution operation is interpreted as removing noise from the graph signal. The spatio-based method expresses the graph convolution as the aggregation of feature information from the neighborhood. When the GNN algorithm runs at the node level, the graph pooling module can be interleaved with the graph convolution layer to coarsen the graph into high-level substructures. This architecture design can be used to extract all levels of graph representation and perform graph classification tasks.

Spectral-based GNN comes from the idea of frequency domain analysis using Fourier Transform. In the spectrum-based GNN, the graph is assumed to be an undirected graph. A robust mathematical representation of the undirected graph is the regularized graph Laplacian matrix.

$$L = I_n - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

where A is the adjacency matrix of the graph, D is the diagonal matrix and we have:

$$D_{ii} = \sum_j (A_{ij})$$

The regularized graph Laplacian matrix has the property of real symmetric positive semidefinite. Using this property, the regularized Laplacian matrix can be decomposed into:

$$L = U\Lambda U^T$$

where $U = [u_0, u_1, .., u_{n-1}] \in \mathbf{R}^{N\times N}$ U is a matrix composed of the eigenvectors of L, Λ is a diagonal matrix, and the value on the diagonal is the eigenvalue of L. The eigenvectors of the regularized Laplacian matrix form a set of orthogonal basis.

In the process of graph signal processing, the signal of a graph is a feature vector composed of each node of the graph and x_i represents the i-th node.

The Fourier transform of the graph X is thus defined as:

$$F(x) = U^T x$$

The inverse Fourier transform is:

$$F^{-1}(\hat{x}) = U\hat{x}$$

Where $\hat{x}$ is the result of the Fourier transform. To better understand the Fourier transform of the graph, from its definition, we can see that it does project the input graph signal to the orthogonal space, and the base of this orthogonal space is determined by the regularized graph. Laplace's feature vector composition.

The elements of the converted signal $\hat{x}$ are the coordinates of the graph signal in the new space, so the original input signal can be expressed as:

$$x = \sum_i \hat{x}_i u_i$$

This is exactly the result of the inverse Fourier transform.

Now we can define the graph convolution operation on the input signal x

$$\begin{aligned} x * g =& F^{-1}(F(x) \odot F(g)) \\ =& U((U^T x) \odot (U^T g)) \end{aligned}$$

where g is the filter we defined, and $\odot$ means Hadamard product. If we define such a filter by:

$$g_\theta = diag(U^T g)$$

Then our graph convolution operation can be simplified as:

$$x * g_\theta = U g U^T x$$

Spectral-based graph convolutional networks all follow this pattern, and the key difference between them is that they choose different filters.

Spatial-based GNN is mainly derived from traditional CNN. The difference is that the space-based graph convolutional neural network defines graph convolution based on the spatial relationship of nodes.

To associate an image with a graph, an image can be regarded as a special form of a graph. Each pixel represents a node directly connected to nearby pixels. Through a 3×3 window, the neighborhood of each node has 8 pixels around it. The positions of these 8 pixels indicate the order of node neighbors. Then, by weighting and averaging the pixel values of the center node and its neighboring nodes on each channel, a filter is applied to the 3×3 window. Due to the specific order of adjacent nodes, trainable weights can be shared across different locations. Similarly, for a general graph, the space-based graph convolution aggregates the central node representation and the adjacent node representation to obtain a new representation of the node.

A common practice is to stack multiple graph convolutional layers together. According to different methods of convolutional stacking, space-based GNN can be further divided into two categories: recurrent-based and composition-based space GNN. The recurrent-based method uses the same graph convolutional layer to update the hidden representation, and the composition-based method uses a different graph convolutional layer to update the hidden representation.

As the earliest graph convolutional network, the spectrum-based model has achieved impressive results in many graph-related analysis tasks. These models have a solid theoretical basis in graph signal processing. By designing a new graph signal filter, we can theoretically design a new graph convolution network. However, the spectrum-based model has shortcomings that are difficult to overcome. We will elaborate them on the aspect of efficiency, versatility, and flexibility.

In terms of efficiency, the computational cost of spectral-based models increases sharply with the size of the graph, because they either need to perform feature vector calculations or process the entire graph at the same time, which makes them difficult to apply to large graphs. Space-based

models have the potential to handle large graphs because they perform convolution directly in the graph domain by clustering neighboring nodes. The calculation can be performed in a batch of nodes, rather than in the entire graph. When the number of adjacent nodes increases, sampling technology can be introduced to improve efficiency.

In general, spectral-based models assume a fixed graph, making it difficult for them to add new nodes to the graph. On the other hand, the space-based model performs graph convolution locally at each node, and can easily share weights between different locations and structures.

In terms of flexibility, the spectrum-based model is limited to undirected graphs. The Laplacian matrix on directed graphs is not clearly defined. Therefore, the only way to apply the spectrum-based model to the directed graph is to convert a directed graph to an undirected graph. The space-based model handles multi-source inputs more flexibly, which can be combined into aggregate functions. Therefore, in recent years, space models have attracted more and more attention.

In the following sections, we will explore GNN based models which can be applied to CVO19 pandemic prediction task. For this task, a geographical map can be considered as a graph with locations of different areas such as city, county, state, and country treated as nodes and edges between areas which share a border. The location nodes and edges are assumed to remain unchanged over time. The only dynamic attributes are information including number of new cases, number of recovered, and number of new deaths. Thus, GNN based on static graphs are not applicable in this specific problem. Dynamic GNN (DGNN) should be considered. A DGNN framework for this task takes node feature and adjacency matrix as input and outputs the risk factor for every location. Some DGNN frameworks have already been proposed. For example, diffusion Convolutional Recurrent Neural Network (DCRNN) [11] and Spatio-temporal Graph Convolutional Networks (STGNN) [12] try to first collect spatial information by using GNNs, and then feed the outputs into CNNs, while on the other hand, Structural-RNN and STGNN tries to collect spatial and temporal messages at the same time. Since this is an active research area, many frameworks and concepts apply to different specific problems are available for us to study, we will present a survey on them in the following parts.

2.3 Dynamic Models

EvolveGNN[13] captures the graph's underlying dynamic sequence through the use of RNN to update GNN's parameters. The model consists of two

parts: a GNN and an RNN. At each timestep t and layer l, model takes the adjacency matrix A_t, the node embedding matrix $H_t^{(l)}$, and the weight matrix $W_t^{(l)}$ to update the node embedding matrix to the next state $H_{t+1}^{(l)}$. In addition, initial embedding is initialized to be the input node features, $H_t^{(0)} = X^T$, and activation function at the final layer could be softmax for node classification or identity for a high-level representation of the graph nodes derived from the initial ones.

The weight matrix is then updated based on the current and historical node information. There exist two variations for this task. One is to treat the weight matrix as a hidden state of the system and update it using a Gated Recurrent Unit (GRU) with the current node embedding and the weight matrix from the last timestep. Another is to consider the $W_t^{(l)}$ as an output of the system and use a Long Short-Term Memory (LSTM) cell to update it only with the weight matrix from previous timestep.

Combining the GNN unit and a recurrent unit, the resulting Evolving Graph Convolution Unit (EGCU) is able to take A_t, H_t^l, and $W_{t-1}^{(l)}$ and output the node embedding at next timestep and the weight matrix at the current timestep.

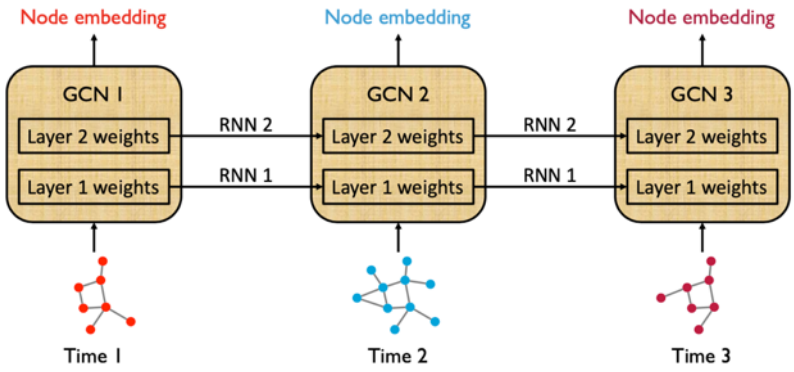

Fig. 3. Block Diagram of EvolveGNN Architecture[13]

The selection of RNN cell is highly data-dependent. With informative node features, GRU cells allows models to focus more on the features, whereas LSTM cells allows focusing more on the structure and its changes. The EvolveGNN architecture outperforms other similar DGNNs. A major advantage is the algorithm can handle dynamic data better due to its inherent structure. The authors see that EvolveGNN performs better than

the static GNN, but less than GNN-GRU. Dynamic models are more effective. An unlearned event causes performance degrade for all methods, with non-dynamic models suffering the most.Dynamic models are not able to perform reliably, because the emerging event has not been learned.The following table shows the mean average precision and the mean reciprocal rank.

	mean average precision					mean reciprocal rank				
	SBM	BC-OTC	BC-Alpha	UCI	AS	SBM	BC-OTC	BC-Alpha	UCI	AS
GCN	0.1987	0.0003	0.0003	0.0251	0.0003	0.0138	0.0025	0.0031	0.1141	0.0555
GCN-GRU	0.1898	0.0001	0.0001	0.0114	0.0713	0.0119	0.0003	0.0004	0.0985	0.3388
DynGEM	0.1680	0.0134	0.0525	0.0209	0.0529	0.0139	0.0921	0.1287	0.1055	0.1028
dyngraph2vecAE	0.0983	0.0090	0.0507	0.0044	0.0331	0.0079	0.0916	0.1478	0.0540	0.0698
dyngraph2vecAERNN	0.1593	**0.0220**	**0.1100**	0.0205	0.0711	0.0120	**0.1268**	**0.1945**	0.0713	0.0493
EvolveGCN-H	0.1947	0.0026	0.0049	0.0126	**0.1534**	**0.0141**	0.0690	0.1104	0.0899	**0.3632**
EvolveGCN-O	**0.1989**	0.0028	0.0036	**0.0270**	0.1139	0.0138	0.0968	0.1185	**0.1379**	0.2746

Fig. 4. Performance of link prediction.[13]

Dynamic Hypergraph Neural Network (DHGNN) [14] tries to solve the issue of the HGNN model, which includes low performance and the inability to exploit higher dimensional features on structures. It introduces a dynamic hypergraph neural network model. The concept of dynamic hypergraph is demonstrated in Fig. 5. The hyperedges evlove with the embeddings. The details of DHGNN is as Fig. 6. DHGNN includes two layers: the dynamic hypergraph construction (DHG) and hypergraph convolution (HGC). While graphs can only represent pair-wise relations, hypergraph is a generalized model for representing non-pair-wise relations by including a set of vertices and a set of hyperedges. A hyperedge may contain various number of vertices, and the number of nodes it contains it defined as its degree. A hypergraph with a degree of two is reduced to a simple graph where each edge connects two vertices.

In the DHG module of the model, k-nearest-neighbors (KNN) and k-means clustering techniques are used to extract global and local information in a single inference process. KNN is used to construct hyperedges while k-means extends the adjacent hypergraphs set. This method is performed on each layer and allows dynamic updates on the vertices and hypergraphs to represent high-order data relations. The HGC module utilizes a set of vertex and hyperedge convolution methods. Vertex convolution uses transform matrices to permute and weight vertices in a hyperedge. Current state-of-the-art methods use precomputed transformation matrices. However, they do not model distinct vertex features well. The model instead uses multi-layer perception and 1-d convolution

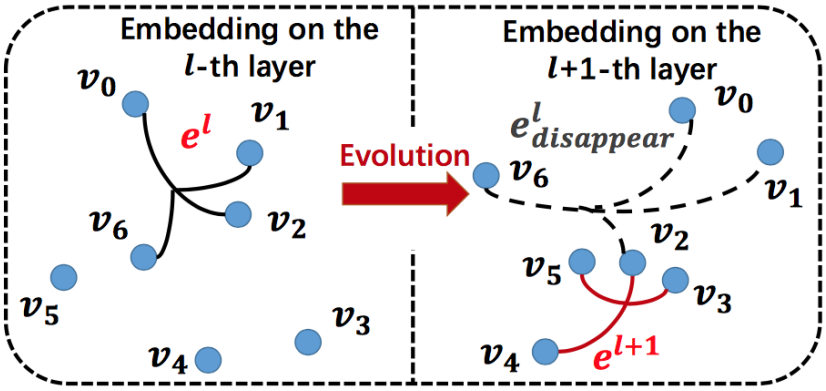

Fig. 5. An example of Dynamic Hypergraph. When the embeddings evolve from l-th layer to $l+1$-th layer, the hyperedge $\{v_0, v_1, v_2, v_6\}$ also evolves to become $\{v_2, v_3, v_4, v_5\}$ [14]

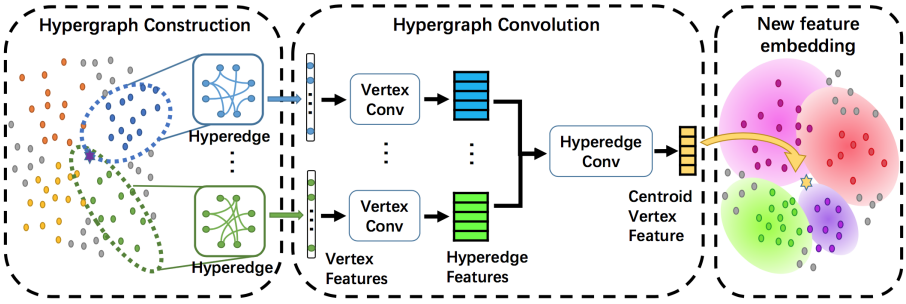

Fig. 6. DHGNN framework. The first frame describes the hypergraph construction process on centroid vertex (the star) and its neighbors. In the second frame, features of vertices contained in a hyperedge are aggregated to hyperedge feature through vertex convolution and features of adjacent hyperedges are aggregated to centroid vertex feature through hyperedge convolutoin. After performing such operations for all vertices on current layer feature embedding, we obtain the new feature embedding where new hypergraph structure will be constructed, as is shown in the third frame.[14]

to learn the transformation matrix from features. This enables modeling of inter-vertex and intra-vertex information. Hyperedge convolution uses the attention mechanism and multi-layer perceptron to aggregate weights from adjacent hyperedges of a centroid vertex. Combing vertex and hyperedge convolution, a hypergraph convolutional layer can be constructed by stacking together hyperedge features, transformed from sampled vertex information in each hyperedge by applying vertex convolution. Then hyperedge convolution transforms hyperedge features to vertex features, followed by a linear and non-linear activation layer. This process also creates new feature embeddings and allows for a new hypergraph to be constructed. A DHGNN is composed of several of these hypergraph layers stacked together.

On a network based classification task with inherent data graph structures, the model outperforms current state-of-the-art methods and adapts to different data distribution better, especially smaller training sets. On a social media sentiment prediction task, the model shows performance improvements against state-of-the-art methods and also yields better time complexity.

DyRep[15] Representation Learning is a fundamental task in the process of graph learning. It learns node representations to effectively convert high-dimensional and non-Euclidean graph information into lower dimensional vectors. DyRep investigates two major questions for representation learning over dynamic graphs. The first one is what can serve as an elegant model for dynamic processes over graphs, and the second one is how can we leverage such model to learn dynamic node representations that are effectively able to capture evolving graph information over time. In this paper, dynamic graphs are going to be modeled in two different dynamic processes. Growing or shrinking of the nodes and edges over time is considered as topological evolution. Activities between nodes that may or may not be connected is considered as node interactions. With the model defined, the next challenge is to effectively model and learn representations that capture the key dynamical properties of such system with highly nonlinear evolution. This paper proposes a general framework trying to tackle these two major challenges.

Two fundamental processes evolving at different time scales are proposed by DyRep in order to express any dynamic graph. The first one is called Association Process which changes the graph structure and thus leads to everlasting information exchange between nodes. The second one is called Communication Process which describes the activates between nodes thus leads to temporary information exchange between nodes. The

key idea is that a concept of latent mediation process is proposed. This process is used to connect the two observed processes. DyRep models complex evolutions of the two observed processes by using the proposed concept of latent mediation process.

The basic framework of DyRep can be summarized into three steps. In the first step, a double scale deep temporal point process is built in order to obtain the temporal dynamics if the two observed processes are in continuous-time domain. Then a conditional intensity function of the temporal point process is parameterized with a deep inductive representation network that learns some functions to compute the node representations. The last step is to propose a new Temporal Attention Mechanism to combine the structural and temporal components together so that the dynamics can be captured over time. This framework is trained by an end-to-end unsupervised training procedure.

Special notice should be made on the part when learning latent mediation process using the proposed temporal attention mechanism. DyRep states that a deep recurrent architecture is built to parametrize the intensity function and learns to compute node representations. The recurrent architecture is neural-network-based and consists of three parts. Self-propagation is defined as a minimal component of the dynamics that drives a node's evolution. Exogenous Drive is defined as some outside force smoothly update the node's current features. And Localized Embedding Propagation is defined as basically the event that a pathway is formed for the information to propagate from one node to the other. Then by putting these three components together into the recurrent network, the representations can be computed.

By conducting different experiments, the paper states that this model do provide improved performance compared to other state-of-the-art methods, and the authors also point out that it is the first generic representation learning framework that can be applied on different dynamic graph characteristics, thus we think that for future work and for the purpose of this project, we can investigate further to see whether this framework can fit in our problem statement.

Graph Neural Network - Recurrent Neural Network (RLGN) [6] By using the SEIR model the authors can denote a state to each node. Every node can represent a person. Their states can be Y = S, L, I, R, namely: susceptible – a healthy, yet uninfected person (S state), exposed/latent – infected but cannot infect others (L state), infectious – may infect other nodes (I state), or removed – self-quarantined and isolated from the graph (R state). The state of a node v ∈ V at time t is a

random variable denoted by ST_v(t). The problem is to reduce the number of S state nodes by reducing the probability of infection and to reduce the I stated nodes. The observation space is the path states of the nodes.

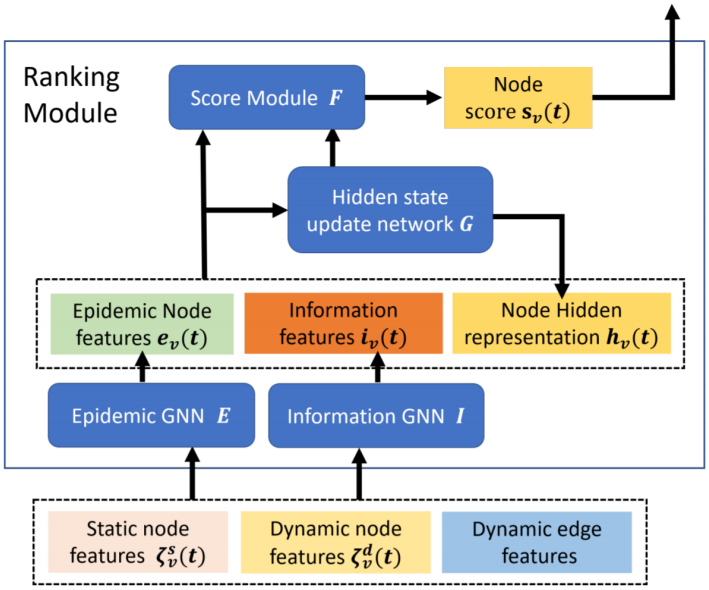

Fig. 7. Block diagram of the ranking module. [6]

The RL agent gets as input the node and side capabilities of the temporal graph, and procedures them the usage of its ranking module. A probability distribution over graph nodes is generated from the ranking module's scores, and the agent samples a subset of k nodes for checking out. Namely, the rankings encode the agent's policy. They use Proximal Policy Optimization set of rules as to optimize their agent. They follow the probable next state and log the (state of the node, probable next state) tuple in a buffer, and train the model primarily based at the PPO loss term.

The GNN-RNN module serves to update the representation of a node v, and (h_v(t)), and its score s_v(t). This score is used for selecting nodes to be acted on. The ranking module is implemented in a recurrent style at different time step. The rating module consists of GNNs: (1) E, which updates the node state, and (2) I, which updates the statistics state. It also incorporates other networks, G and F, which replace node representations

and node rankings by using the usage of the node state and records the future state as nicely as the previous node representations. The input to the ranking module includes three feature sorts:

(1) Static node capabilities ζ_v^s(t): topological graph centralities (betweness, closeness, eigenvector and degree centralities) and random node features.

(2) Dynamic node features ζ_v^d(t): All test outcomes that had been finished up at current timestamp (such as high-quality and bad take a look at outcomes).They denote all nodes features as a concatenation ζ_v(t) = [ζ_v^s(t), ζ_v^d(t)]

(3) Edge capabilities and the shape of the temporal graph E(t): All previous interactions up to the contemporary step, inclusive of the transmission possibility for each interplay.

Fig. 7 illustrates the fundamental facts go with the flow inside the ranking module. The spread of epidemic thru point touch is modeled with the aid of a GNN E. If the probability of transmission during an interaction between to nodes (v,u) at time t can be denoted by p_{vu} and multilayer perceptrom(MLP) is denoted by M. For each v they can generate a feature vector e_v(t) from E from the following formula.

$$e_v(t) = \sum_{u \sim_t v} p_{vv'}(t) * M_e(\zeta_v(t), \zeta_v'(t); \theta_{m_e}),$$

The score of a node is influenced both by the spread elements and by the data accessible to the operator. One may trust that since the previous has a known timescale (days), on a short time scale (single day) the score of the node would just be influenced by its neighboring hubs. This, nonetheless, is not genuine in light of the fact that data can engender significant distance in the chart quickly. As a basic model, consider hubs in an associated chain of (untested) hubs and note that they are measurably needy. Accordingly, uncovering the condition of one node promptly influences the dispersion over all hubs in the chain. They planned a data GNN, I, which speaks to the data condition of every node.

As talked about above, refresh data on a node u a couple of bounces from node v may unexpectedly change their convictions on the condition of v. Moreover, this change may happen regardless of whether v and u didn't interact in the last time step yet rather some time back. For updating the information state they use a cumulative multi-graph G' where the edges represent the interactions between v and u in the last τ steps ($\xi = \cup_{t' \in [t-\tau, t]} \xi_G(t)$). If the features of each edge of $\xi_{G'}$ is denoted by

ϕ_{vu}(t') are in the time intersection of t-t' with the transmission probability of $p_{v,v'}$(t'). Then the l^{th} layer of a k-layer GNN can be denoted by:

$$x_v^l(t) = \sum_{v' \sim_t v} M^l(x_v^{l-1}(t), x_{v'}^{l-1}(t), \phi_{vv'}(t); \theta_M^l)$$

Here the M^l is MLP with $x_v^0(t) = \zeta_v(t)$ and $x_v^k(t) = i_v(t)$ as the final node feature. This allows us to define the hidden state function as:

$$h_v(t) = G(h_v(t-1), \zeta_v(t), e_v(t), i_v(t); \theta_g)$$

after updating the $h_v(t)$ of all the states, using an MLP denoted by F, and G denoting an MLP or a recurrent model such as Gated recurrent Unit (GRU), they updated the scores with the following function:

$$s_v(t) = F(h_v(t), h_v(t-1), \zeta_v(t); \theta_f)$$

With the updated scores $s_v(t)$ the authors could iteratively sample without updating by first updating the scores of the nodes a probability distribution using a score-to-probability distribution function. After which they can sample the nodes and then adjusting the distribution by removing its weight and repeating this process k times.

The activity space of picking a subset of k hubs out of n hubs is enormous in any event, for little n and k. Utilizing an activity esteem approach like Q-learning suggests that an estimated esteem is allocated to each conceivable activity, however the activity space size is restrictively excessively enormous for activity esteem strategies. All things being equal, they use a strategy inclination calculation and model the issue as a positioning issue. The calculation figures out how to rank hubs utilizing a defined model, and afterward utilizes an inspecting method to pick a subset of k hubs. Numerous on-strategy inclination calculations use entropy to characterize a trust district.

Registering the entropy requires summing $\binom{K}{|V|}$ terms at each progression, and it is computationally costly. A more adaptable arrangement is the fair-minded entropy assessor of, however the fluctuation of that assessor is high. As another option, PPO trust locale isn't in light of an unequivocal assessment of the entropy, and performed better in their trials. They likewise assessed A2C, which didn't proceed just as PPO in their tests. PPO, as an actor-critic calculation, requires a critic module to assess the worth capacity in a given state. They build the actor utilizing a design that is like the positioning module, yet apply to element wise max procedure on the lines (representing the nodes) of the contribution to the

score module F (Figure 5). This decreases F's contribution to a single row of features, and the yield received is then a scalar instead of a vector.

Significantly, the critic is parametrized by an alternate arrangement of loads than the positioning module (actor). Ordinarily, node scores are changed over to a conveyance over activities utilizing a softmax. This methodology is risky for their case since node probabilities reduce dramatically with their scores, prompting two significant disadvantages. It demoralizes investigation of low-score hubs, and limits affect ability to the highest point of the appropriation, rather than at the top-k selected. To solve it they use the following equation to define the probability to sample an action a_i:

$$PR(a_i) = x'_i / \sum x'_i, where$$

$$x'_i = x_i - min_i x_i + \varepsilon$$

where x_i is the set of scores and ε is a constant. By not utilizing a remarkable as in softmax, the likelihood contrasts between low scoring hubs and high scoring hubs become less extraordinary. Moreover, the constant controls the underlying investigation proportion. In standard DNN initialization plans (e.g., XAVIER), the underlying worth of xi is expected to be in [-1,1]. if $\varepsilon \gg 1$ than the ε predominant term in the condition above. This advances investigation at first, as all activities are probably going to be inspected in the early preparing stages.

RNN are notable to experience the ill effects of the issue of detonating or evaporating angles. This issue is exacerbated in a RNN-GNN structure utilized for RL calculations, in light of the fact that they might be applied for subjective long scenes, making inner state become unbounded. This issue is especially extreme if the underlying graph contains hubs. One way to deal with this issue, is by including a RNN like a GRU module, where the concealed state go through a sigmoid layer. As the greatness of the information develops, gradient become more less inclined. Without scale networks contain with high likelihood "hub" nodes that have serious level, in particular O(n) neighbors. The presence of these center points further exasperates this issue. As a basic case, consider a star diagram with countless hubs.

In a GNN system, it gets refreshes from countless neighbors and its inner state increments in extent. Whenever that the GNN module is applied (e.g., at the following RL step), the developing inward state expands the size of the inner condition of its neighbors. This prompts a positive feedback circle that makes the inward state portrayal wander. Since RL calculations

might be applied for discretionary extensive stretches, the interior state may become unbounded except if remedied. This issue can be tackled by legitimately normalizing every node hidden state. They explored different avenues regarding different standardization strategies, and found that L_2 standardization worked best, as appeared in the following area. On account of the COVID-19 pandemic, the progress probabilities can be assessed utilizing the connection properties, for example, term and between close to home separation, utilizing known epidemiological models. This was finished by the administration organization which gave them contact following organization.

Then again, one can gain proficiency with the transmission likelihood as a relapse issue from known communications (e.g,utilizing information from post-contamination addressing). At last, if this data isn't open, it is conceivable to exclude the plague model E from the proposed structure and utilize just the element vector made by the data module I. They further assessed how models prepared on medium-size charts sum up to perform deduction on much bigger charts. In particular, They prepared RLGN and SL+GNN (three model introduction for each) on a special connection network with 1000 nodes and assessed its presentation on different bigger diagrams. They observed the movement of the pestilence under either RLGN or supervised+GNN calculations. It was noted that the growth rate of the percentage of infected nodes was less as compared to GamesecRO, Enron, and ca-GrQc.The following graphs show the observed infected percentage as compared to the number of steps.

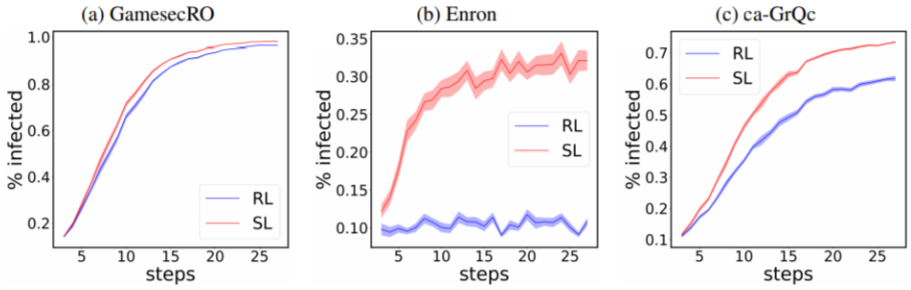

Fig. 8. The fraction of infected nodes as a function of time step t [6]

This method shows that consolidating RL with GNNs gives an amazing way to deal with controlling spreading measures on charts. With regards to COVID-19 spread, they exhibited that utilizing a RL+GNN approach permits them to bind the spread of a scourge that is around 30 percent

more infectious (i.e., R0 that is 30 percent higher) with similar assets as a standard directed learning-based methodology. Their outcomes show that organizing tests utilizing RL on fleeting charts can expand the quantity of solid individuals by 25 percent and contain the scourge 30 percent more frequently than managed approaches and 2.5× more regularly than non-learned baselines utilizing similar assets. The following tables compares the results provided by the paper with the models:

	%contained	# training epochs
Sigmoid	0.84 ± 0.05	1210
GRU	0.91 ± 0.03	810
L_2 norm.	$\mathbf{0.93 \pm 0.02}$	**500**

Table 1. L_2 normalization has the lowest training time and fraction of contained epidemic. [6]

$n = 300$	Init. infection size 5%		Init. infection size 7.5%		Init. infection size 10%	
	%healthy	%contained	%healthy	%contained	%healthy	%contained
SL, $k = 1\%$	27 ± 2	15 ± 5	21 ± 2	4 ± 2	18 ± 1	1 ± 1
SL, $k = 1.33\%$	41 ± 3	37 ± 6	27 ± 2	12 ± 4	24 ± 2	6 ± 3
SL, $k = 2\%$	66 ± 4	76 ± 6	48 ± 3	55 ± 7	37 ± 2	32 ± 6
RLGN, $k = 1\%$	50 ± 2	78 ± 7	43 ± 2	58 ± 1	40 ± 1	48 ± 6

$n = 500$	Init. infection size 5%		Init. infection size 7.5%		Init. infection size 10%	
	%healthy	%contained	%healthy	%contained	%healthy	%contained
SL, $k = 1\%$	24 ± 2	7 ± 4	20 ± 1	2 ± 1	19 ± 1	0 ± 1
SL, $k = 1.6\%$	48 ± 3	54 ± 6	35 ± 2	27 ± 7	29 ± 1	11 ± 1
SL, $k = 2\%$	67 ± 3	83 ± 5	46 ± 2	53 ± 4	38 ± 2	37 ± 7
RLGN, $k = 1\%$	52 ± 1	97 ± 2	44 ± 2	75 ± 11	42 ± 1	66 ± 6

$n = 1000$	Init. infection size 5%		Init. Infection size 7.5%		Init. infection size 10%	
	%healthy	%contained	%healthy	%contained	%healthy	%contained
SL, $k = 1\%$	25 ± 2	5 ± 3	21 ± 1	0 ± 1	19 ± 1	0 ± 0
SL, $k = 1.5\%$	42 ± 2	49 ± 6	30 ± 1	10 ± 3	27 ± 1	4 ± 2
SL, $k = 2\%$	66 ± 1	84 ± 5	45 ± 2	59 ± 5	37 ± 1	30 ± 1
RLGN, $k = 1\%$	52 ± 1	97 ± 2	44 ± 2	75 ± 11	42 ± 1	66 ± 6

Table 2. RLGN performance is higher than SL+GNN in each case. The evaluation was performed on a preferential attachment network with mean degree 2.8. [6]

	CA-GrQc	Montreal	Portland	Enron	GEMSEC-RO
Degree Centrality	25.52 ± 0.01	12.83 ± 0.01	0.67 ± 0.01	71.14 ± 0.02	2.43 ± 0.01
Eigenvector Centrality	25.37 ± 0.01	8.06 ± 0.01	0.04 ± 0.01	55.10 ± 0.02	2.41 ± 0.01
SL	29.79 ± 0.02	23.09 ± 0.03	1.57 ± 0.01	68.45 ± 0.05	4.26 ± 0.01
RLGN	**42.69±0.03**	**39.68±0.03**	**3.71±0.01**	**89.19±0.02**	**6.52±0.01**

Table 3. The mean percentile of healthy nodes after a 20 steps. RLGN performance is higher on all datasets. [6]

2.4 Spatio-temporal Models

Spatio-temporal graph neural network [12] focuses on improving accuracy of COVID-19 forecasting by incorporating real-time inter-regional and intra-regional mobility data from GPS enabled devices and developing a unified approach for both spatial and temporal data. The model consists of spatial edges in a layer which models mobility and temporal edges between stacked layers that model time information.

The convolution layers are based on the spectral graph convolution model proposed by [1]. Each layer is computed by MLP with a spectral normalized adjacency matrix, a learned weight matrix, and a non-linear activation layer, concatenated with a learned embedding of temporal node features. There are also skip-connections made over layers to prevent diluting self nodes.

The input data consists common node features such as state, county, day, past cases, and aggregated mobility research dataset by Google, along with community mobility reports. The model was run using a one-week time horizon. Data from March and April 2020 in the US was used as training dataset, and data during May 2020 was used as test dataset. The top 20 counties by populations were observed when evaluating the model. The model achieved better performance than the baseline model based on RMSLE and Pearson Correlation. The addition of mobility data did improve predictions. The merit of this work is that the input node signal is coupled with the information propagated by its neighboring nodes, which facilitate the process of inferring hidden states. However, there are some limitations in this study. One of the disadvantages is the limitations of the data sources, as Google mobility data only includes smartphone users who turned on the location history feature. Also, other environmental conditions such as the rate of people wearing masks and weather conditions are not included in this model. This model can be further generalized and expanded on larger datasets and various types of data features.

Another graph neural network model that captures both temporal and spatial data [16] predicts the confirmed cases of COV19 that leverages Model-Agnostic Meta-Learning to transfer knowledge learned from stabilized regions of outbreak to newly spread regions. Three variants of the model are introduced below.

– **MPNN**: Each country is represented as a series of graphs, with vertices representing the regions and edges within representing mobility. Each graph in the series represent data on different dates and therefore the stack of graphs provide the temporal information. The model uses a family of GNNs known as the message passing neural networks (MPNN) [17]. They contain neighborhood aggregation layers that can update each layer of graphs based on the normalized weight adjacency matrices following Kipf and Welling [1]. This model is applied to all layers in a graph that differ by time, with unique adjacency matrices and node representation matrices for layers, and shared weights between layers. The model captures more global information as the number of neighborhood aggregation layers increases, however, there should be a balance between maintaining local and global information. The node representation matrices are concatenated to create skip connections between each layer and the output layer. This will allow the network to encode multi-scale structural information. Then the output of the network is followed by a ReLU. The following figure shows the proposed MPNN architecutre. [18].

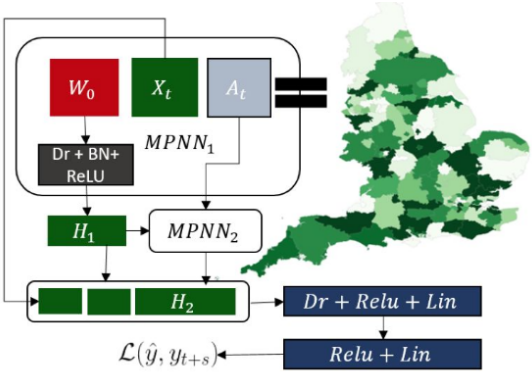

Fig. 9. proposed MPNN architecutre [18]

- **MPNN+LSTM**: To utilize temporal information, the sequence of graphs representing different time stamps are processed by MPNN, and the resulting sequences of node representation matrices are utilized by two layers of Long-Short Term Memory network (LSTM). This model can then capture long term temporal information in hidden states.
- **MPNN+TL**: Another model is revised based on the Model-Agnostic Meta-Learning (MAML) algorithm [16] to capture patterns in a regional outbreak from the start and transfer it to other regions. Different waves of COV19 may share information, however this also means the model has fewer training samples and should start predicting from as early as the 15th day. The weight matrices and bias learned from all layers in the MPNN models can be used as datasets and extract feature parameters for initializing states in other regions. The MAML model randomly initializes parameters and uses gradient descent to locally minimize the loss based on different prediction tasks.

  The model is evaluated on three countries in Europe and done on short, mid, long term range of times. Different models were trained specifically for a time prediction range. Prediction results were compared with baselines models on the following evaluation metrics: average number of cases with respect to time, average number of cases in a past time frame, ARIMA: autoregressive average moving model [19], LSTM, and Prophet: a state-of-the-art forecasting model [20]. These models in most cases yielded smaller average errors per region compared to the baseline models. In the three model variants, MPNN+TL performed better with limited data in small time ranges and overall, where MPNN was able to capture more over longer time ranges. The predictions from three models are expected to converge in the end. When experimented on data from different regions, the models produced small relative errors (less than 20%) in independent regions, especially in regions with more data, thus they capture the epidemic accurately and can be further aided in policy decision making. Upon experimentation the following errors were calculated between the three methods.The authors see that in most cases, the proposed models yield below average errors. Among the three variations, MPNN+TL is the best-performing model. It at first outflanks the different methodologies just barely that increments further after the subsequent day. Even basic baselines can be accurate enough at foreseeing the following day's number of cases since proximal tests for similar area from similar periods of the pandemic will in general have comparative number of cases. In any case, a forecast that goes further in time needs to recognize more steady

examples. On account of our model, the authors intended to catch unregistered cases moving from one district to the next or spreading the infection in their new locale. These cases would definitely take a couple of days to show up, because of the postponement of side effects related with COVID-19. This is the reason MPNN performs well all through the 14-days window. The outcomes likewise exhibit the advantage of move learning procedures since MPNN+TL beats MPNN in all cases.

Model	Next 3 Days			Next 7 Days			Next 14 Days		
	England	Italy	Spain	England	Italy	Spain	England	Italy	Spain
AVG	9.75	21.38	45.10	9.99	22.23	45.87	10.09	23.09	47.63
AVG_WINDOW	6.52	15.17	32.19	7.34	16.81	36.06	8.54	19.45	42.79
LSTM	8.50	21.70	48.48	8.37	22.03	48.09	8.52	22.00	46.17
PROPHET	10.58	24.86	54.76	12.25	27.39	62.16	16.24	33.07	79.42
ARIMA	13.77	35.28	40.49	14.55	37.23	41.64	15.65	39.65	46.22
MPNN	6.13	14.35	30.54	6.58	15.31	32.83	7.24	17.12	34.81
MPNN+LSTM	6.37	15.59	33.41	6.80	17.02	35.92	7.53	19.72	37.96
MPNN_TL	**5.96**	**14.12**	**29.27**	**6.18**	**14.55**	**30.14**	**6.32**	**14.97**	**31.36**

Table 4. Average error for delta t = 3, 7 and 14, in number of cases per region [18]

STAN [21] is another spatio-temporal model. It designs nodes and edges based on demographical and geographical similarities between regions. It also takes in various kinds of data including recovered and death cases, local medical resource and disease data, and integrated pandemic transmission patterns with deep learning models. Each node represents a region and has static and dynamic attributes from real world data, while edges represent demographical and geographical similarities. The model incorporates attention mechanism into graph convolution networks for interaction between neighbors. Predictions are made in a fixed time range, while keeping physical constraints that underly transmission dynamics in the real world.

STAN is shown in Fig. 10. The model consists of a graph neural network for spatial information, an RNN that captures temporal information, and physical law constraints for different time ranges. A two-layer GNN captures spatial features from the latest data and part of historical data within a sliding window. Graph attention mechanism (GAT) can learn hidden embeddings of nodes and calculates attention coefficients, followed by self-attention and aggregation for each node. This can model the knowledge that different regions can have different infectious impact. Then the result is aggregated from all nodes and fed into Gate Recurrent

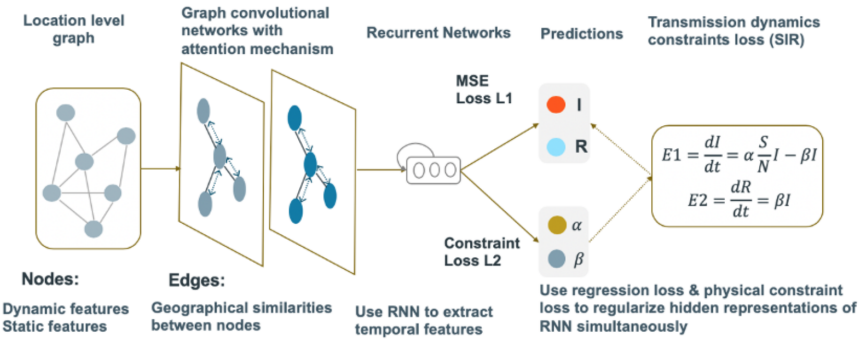

Fig. 10. The STAN model: Location graph is constructed using location-wise dynamic. Static features and geographic proximity are constructed as nodes and edges. The graph is fed into GNN with attention mechanism to extract spatio features and learn the graph embedding for the target location. Then the graph embedding is fed into the GRU to extract temporal relationships. The hidden states of GRU will be used to predict future number of infected and recovered cases. We use an additional physical loss based on pandemic transmission dynamics to optimize the model.[21]

Unit (GRU) [22] to produce an embedding that contain both spatial and temporal information. The physical constraints include predicting transmission/recovery rates over time with MLP, number of infected/recovered cases, and physics constraints loss from calculated data and SIR differential equations.

STAN achieved better performance than epidemiological modeling methods and deep learning methods with up to 87One limitation of the network is the sliding window prediction which requires more accurate data input, which can be further solved by dynamic smoothing to the data. Another limitation is that the physical constraint models may be too simple for the realistic situations, and it can be improved by incorporating more population groups and transmission equations, similar to improving the SIR model.

Geographic and Long-term Temporal Graph Convolutional Recurrent Neural Network (GLT-GCRNN)[23] is also a model that considers both geographic and long-term temporal information when adopting GNN to mining the spatial dependency.

In order to predict the traffic, GLT-GCRNN encodes the road network as a graph with vertices and edges, where each vertex represents the links or censor stations, and edges between the vertices represent the relevance of these links. In order to include temporal information, there is this notion of time step, where the paper picked 5 minutes to be a time step,

and thus there are 288 time steps during one day. At each time step, the state of the traffic is encoded as a vector of traffic states of all links, so the goal of the model is to produce a function such that given an array of vectors of previous traffic states, the function should be able to produce an array of vectors predicting the state of the traffic links for the next H time steps given the road network G.

More specifically, in the geographic aspect, GLT-GCRNN adopts the adjacency matrix $A \in \mathbb{R}^{N \times N}$ to represent the correctness of links. Thus, the k-hop similar matrix in geographic aspect S_G can be computed:

$$S^k_{G_{i,j}} = min((A + I)^k_{i,j}, 1) \tag{2}$$

In the long-term temporal aspect, the paper presents a novel method. The long-term temporal different matrix Q is constructed. Each element of Q is defined by the Euclidean distance of any two link's average speed distribution $\hat{v}(i)$ and $\hat{v}(j)$ across the training set after the combination every three time steps. Note that this combination will reduce the dimension of the link's one day speed distribution vector from 288 to 96. The similar matrix in the long-term temporal aspect S_{LT} can be computed as this using the Q notation just introduced:

$$S_{LT_{i,j}} = \begin{cases} 1, Q_{i,j} \in Q_i, top\ \gamma\ small\ elements \\ 0, otherwise \end{cases} \tag{3}$$

Then, the paper formulates the k-hop geographic and long-term temporal similar matrix S_{GLT} using the above two notations:

$$S^k_{GLT} = S^k_G + S_{LT} \tag{4}$$

As the last step, the paper added the free flow speed similar matrix and form the ultimate similar matrix for the GNN:

$$S^k_U = S_{GLT^k} \odot S_F \tag{5}$$

And the model can be represented as:

$$g^k_t = (W^k_g \odot S^k_U)x_t \tag{6}$$

where g^k_t is the feature extracted by GLT graph convolution operation, and the W^k_g is the trainable weight matrix.

2.5 Other Models

DeepInf DeepInf [24] was orginally developed for applying deep neural network techniques to learn users' latent feature representation for predicting social influence. The definition of social influence can be very broad, and the paper defined it as the phenomenon that a person's emotions, opinions, or behaviors are affecting other people. And this concept is critical in the applications such as online recommendation and advertising. Thus, the overall goal of this paper is to design a framework that can characterize, understand, and quantify the underlying mechanisms and dynamics of social influence. In particular, the goal for this paper is to predict the action of a user given the action of the user's near neighbors and local structural information. And an end-to-end deep learning based framework DeepInf is proposed to try to discover hidden and predictive signals in social influence.

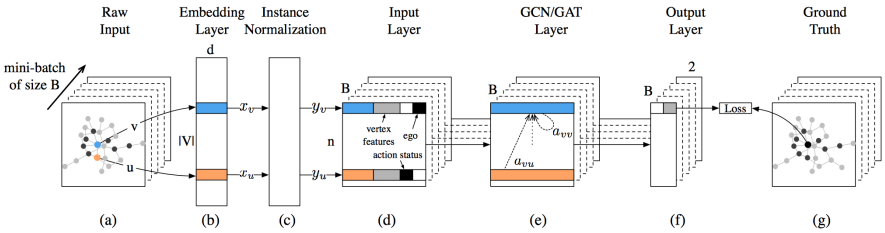

Fig. 11. Model Framework of DeepInf. (a) Raw input which consists of a mini-batch of B instances; (b) An embedding layer which maps each user to her D-dimensional representation; (c) An Instance Normalization layer; (d) The formal input layer which concatenates together network embedding, two dummy features (one indicates whether the user is active, the other indicates whether the user is the ego), and other customized vertex features; (e) A GNN or GAT layer. a_{vv} and a_{vu} indicate the attention coe cients along self-loop (v, v) and edge (v, u), respectively; (f) and (g) Compare model output and ground truth[24]

The structure of DeepInf is in Fig. 11. There are three major components of this framework: network embedding, graph convolution, and graph attention mechanism. The basic idea of this framework is to first sample a user (node)'s local neighbors by random walks with restart, and then after obtaining the local network, graph convolution and attention techniques are used to learn latent predictive signals. Obviously, this is a graph problem, and each user can be view as a node in the graph, and edges connect users that can have influence on each other. The first step

is to do the sampling. This is done by using random walks with restart (RWR) with the assumption that active nodes will tend to influence their neighbors more likely than inactive nodes. The RWR starts with either one of the ego users or its active neighbors, and then iteratively traverse to its neighbors with a probability proportional to the weight of the edges. The RWR stops when a pre-defined number of vertices have been visited. Then the raw input is fed into a neural network. First, the input is fed into the embedding layer, which uses a pre-trained model to convert a node into a lower dimensional representation vector. And then normalization is used to avoid overfitting during training. After that, the input layer constructs a feature vector for each node. Then it is fed into a Graph Convolutional Network (GNN) or a Graph Attention Network (GAT). And then the output layer outputs a 2-D representation for each user, and it is used to compare to the ground truth to optimize the loss function.

Datasets from four different domains, OAG, Digg, Twitter, and Weibo, are used to conduct the experiment. The baselines are Logistic Regression, SVM, and PSCN. The paper states that DeefInf significantly outperforms the baselines with hand-craft features. For our purpose, there is one thing worth investigating is that what if we do not use the SIR model that governing the network dynamics, and let this framework learn directly from the dataset.

2.6 Summary

In conclusion, the models presented can be helpful in predicting dynamic data that evolves not only spatially but also temporally. The DHGNN models dynamic hypergraphs, which in our case can be simplified to a simple graph with setting the degree of the hypergraph to 2. The other models take different approaches to modeling COV19 and epidemic forecasting utilizing spatial and temporal graphs. Models supporting dynamic data, such as STAN, may be the ones which are more plausible for COV19 pandemic predicition task. Further work can be done to combine the characteristics and advantage of these models to predict COV19 with dynamic features in addition to static geographical information.

References

1. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. CoRR **abs/1609.02907** (2016)
2. Feng, Y., You, H., Zhang, Z., Ji, R., Gao, Y.: Hypergraph neural networks. CoRR **abs/1809.09401** (2018)

3. Monti, F., Boscaini, D., Masci, J., Rodolà, E., Svoboda, J., Bronstein, M.M.: Geometric deep learning on graphs and manifolds using mixture model cnns. CoRR **abs/1611.08402** (2016)
4. Hamilton, W.L., Ying, R., Leskovec, J.: Inductive representation learning on large graphs (2018)
5. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. (2018)
6. Meirom, E.A., Maron, H., Mannor, S., Chechik, G.: How to stop epidemics: Controlling graph dynamics with reinforcement learning and graph neural networks (2020)
7. Shah, C., Dehmamy, N., Perra, N., Chinazzi, M., Barabási, A.L., Vespignani, A., Yu, R.: Finding patient zero: Learning contagion source with graph neural networks (2020)
8. Yu, B., Yin, H., Zhu, Z.: Spatio-temporal graph convolutional neural network: A deep learning framework for traffic forecasting. CoRR **abs/1709.04875** (2017)
9. Deng, S., Wang, S., Rangwala, H., Wang, L., Ning, Y.: Graph message passing with cross-location attentions for long-term ili prediction (2019)
10. Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., Sun, M.: Graph neural networks: A review of methods and applications. CoRR **abs/1812.08434** (2018)
11. Li, Y., Yu, R., Shahabi, C., Liu, Y.: Diffusion convolutional recurrent neural network: Data-driven traffic forecasting (2018)
12. Kapoor, A., Ben, X., Liu, L., Perozzi, B., Barnes, M., Blais, M., O'Banion, S.: Examining covid-19 forecasting using spatio-temporal graph neural networks (2020)
13. Pareja, A., Domeniconi, G., Chen, J., Ma, T., Leiserson, C.: Evolvegcn: Evolving graph convolutional networks for dynamic graphs (2019)
14. Jiang, J., Wei, Y., Feng, Y., Cao, J., Gao, Y.: Dynamic hypergraph neural networks. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, International Joint Conferences on Artificial Intelligence Organization (7 2019) 2635–2641
15. Trivedi, R., Farajtabar, M., Biswal, P., Zha, H.: Representation learning over dynamic graphs, ICLR (2018)
16. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. Volume 70 of Proceedings of Machine Learning Research., International Convention Centre, Sydney, Australia, PMLR (06–11 Aug 2017) 1126–1135
17. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. Volume 70 of Proceedings of Machine Learning Research., International Convention Centre, Sydney, Australia, PMLR (06–11 Aug 2017) 1263–1272
18. Panagopoulos, G., Nikolentzos, G., Vazirgiannis, M.: United we stand: Transfer graph neural networks for pandemic forecasting (2020)
19. T, K.: Arima-based forecasting of the dynamics of confirmed covid-19 cases for selected european countries. equilibrium. Quarterly Journal of Economics and Economic Policy **15(2)** (2020)
20. S, M.: Bangladesh covid-19 daily cases time series analysis using facebook prophet model. SSRN **3660368** (2020)
21. Gao, J., Sharma, R., Qian, C., Glass, L.M., Spaeder, J., Romberg, J., Sun, J., Xiao, C.: Stan: Spatio-temporal attention network for pandemic prediction using real world evidence (2020)

22. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling (2014)
23. Sun, Y., Wang, Y., Fu, K., Wang, Z., Zhang, C., Ye, J.: Constructing geographic and long-term temporal graph for traffic forecasting (2020)
24. Qiu, J., Tang, J., Ma, H., Dong, Y., Wang, K., Tang, J.: Deepinf:social influence prediction with deep learning. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining (Jul 2018)

Contribution by the team

A list of 45 papers were initially collected. After the first-stage reading, 24 were related to our task purpose (Monitoring Pandemics And Risk Factors) and included into this survey. These works were conducted by 10 members from the two teams(Team - Big Data Group: Jason Kao, Sripath Mishra, Boya Ouyang, Vishnu Devarakonda. Team - Ideas: Panqiu Tang, Huiling Huang, Chenyang Wang, Haochen Yin, Yijing Zhou, Danfeng Guo).

The work distribution for Comprehensive paper is as follows. All team members who accrued papers gave a summary of the papers they had enlisted. Sripath Mishra (team - Big data group) and Danfeng Guo (team - Ideas) have collected all these summaries and made it coherent into one document. All other active members have contributed towards the representative paper and the project proposal (Team - Big Data Group: Jason Kao, Boya Ouyang, Vishnu Devarakonda. Team - Ideas: Panqiu Tang, Huiling Huang, Chenyang Wang, Haochen Yin, Yijing Zhou).