Ensemble Approaches for Streaming Networking Classification

Anak Wannaphaschaiyong

I. Introduction

Often, real world problems involve entities and their relationship. This is best represented by graph data structures. In the case where graphs evolve overtimes, dynamic graph can be used to model them.

In the field of machine learning, dynamic graph modeling is still under explored, but its applications has been explored more recently in the past few years. Applications involve tasks such as graph protection, link prediction, entity/relation prediction, node clustering, node classification, and node attribute prediction (aka node regression), just to name a few [9]. Dynamic graph models are used to solve real world application involves traffic prediction [24], recommendation system [9], knowledge graph completion [2], ecological network, distributed computing, interactions of economic agents, brain networks, [6]

Dynamic graph modeling also has application in reinforcement learning. Reinforcement learning was used to solve tasks such as graph protection problems on dynamic graph [22, 21]. For example, Meirom at el. [13] uses graph neural network to propagate information and reinforcement learning to rank highly infectious candidate. As of recently, during the Covid-19 pandemic, Kapoor et al. [8] applies graph neural network on mobility data, which is represented as dynamic graph data, to predict state cases (represented as node value). The task to predict node's value is called node attribute prediction and also known as node regression task. It is important to make it clear that, at the moment, dynamic graph is overly used in many fields to mean different things. The meaning of dynamic graph is too diluted. For this reason, we will define dynamic graph and related terms. In general, dynamic graph is an ordered list of node and link events. These events include deletion and addition of nodes and edges after a interval of time. Dynamic graph has many (imperfectly) equivalent names with conflicting meaning as streaming graph and temporal graph. Embedding dynamic network surveys have been published to clarify terminology by purposing their own taxonomies [1, 9, 17]. We adopt taxonomy presented by Skarding et al. [17] because it puts emphasis on taxonomy of dynamic graph neural network (DGNN), in particular, rather than deep learning on dynamic graph in general. To understand dynamic graph, one must understand static network. In static network, one must consider type of network relationship (e.g idealize network, proximity network.), scale of network (e.g. a node as a single entity, a node as a group of entities.), and network variation (link types e.g. homogeneous network, heterogenous network, multilayer network). Each mentioned factors encounters its own unique challenges. Importantly, these factors must be considered before model designing phase begins otherwise network based models cannot be expected to be compared fairly with other models. Moreover, it provide mental framework to guide designing process. GNN is considered a modern solution to static graph modeling. GNN can be categorized into spectral GNN and spatial GNN. Layer of spectral GNN is a k-localized convolution filter where the filter is in Fourier domain, such as Laplacian matrix (k is 1) [10], to diffuse information between nodes via connected links. Later, spatial filter is invented by introduction of message passing framework [4]. Spatial filter can be compute locally without the whole graph input. This means spatial GNN are inductive while spectral GNN are tranductive. In practice, the local computation can be compute in parallel to speed up computational process.

Another problem that unique to dynamic graph is representation. Holme et al. [7] discuss that temporal properties depends on underlying dynamic graph representation, such as different network, aggregated network, time-varying network (a graph where edges are labeled based on time interval where they present in the graph), and multilayer network. For this reason, graph construction should be considered as an extremely important models design decision of data driven approaches such as deep learning. Deep learning models are directly trained on graph representation. Nonetheless, this topic doesn't get as much attention as it should. Furthermore, according to Holme et al. [6], problem of representation of graph has been solved yet. There is no way to include all relevant dimension of dynamic graph into one complete dynamic graph representation. At the moment, which representation to choose depends on representation's accuracy and information one is willing to include or discard. That's it. Hence, one must chooses to represent dynamic graph in lossy and lossless ways. In addition to direct impact on the models of representation, illustrations are great to support discussion, without specifying assumption of the representation clearly, motivation of the purposed model and problem that the author solves can be misinterpreted or, even worse, misleading. For more information please refer to Holme et al. [6]

Dynamic graph representation can also be modified. This is done by adding synthetic edges. Synthetic edges can be added to dynamic graph representation which involves adding edges between nodes that has physical connection. Kapoor et al. [8] constructs graph as discrete graph and synthetic edges are added between the same nodes across multiple aggregated graph. Before design static graph models, one must select input construction approach and modeling approach. Dynamic graph

extends static graph to include time variables which added time dimension to the problem. Adding time dimension, option of input construction approach and modeling approach increases. In addition to static graph modeling approach, one must consider dynamic behavior of the graph which is classified into "dynamic behavior on graph" and "n-dimension network topological evolution" [6, 1]. Dynamic behavior on graph are determined by node dynamic [17], edges types, and temporal granularity [17].

Temporal granularity concerns time information that are kept in a graph which broken down to unweighted static graph (no time information is preserved), weighted static graph, discrete graph (stack of multiple weighted static graph), continuous graph (all time information is preserved) [17]. Temporal properties concerns temporal features of an entity that compose a graph such as nodes, edges, and motifs. The temporal dimension can yield non-trivial temporal features such as interaction distribution (distribution of all edges over-time), interevent distribution (distribution of an edges/event over time), among others [7, 6]. Interevent time distribution is the frequency distribution between the events. If the events are independent and drawn from a uniform distribution, then the inter-event time distribution will be exponential. However, empirical data set usually has fat-tailed, scale-free rather than uniform distribution. Coefficient of variation, bustiness, is used to characterized scale-free degree inter-event distribution. Dynamic behavior on graph concerns non-graph temporal features, such as communication behavior between nodes and substructure like motifs, and process on the network such as diffusion cascades. Lastly, n-dimension topological evolution involves structure evolution, features evolution, role evolution of nodes in a graph.

For static network, degree distribution is established itself as one of the most fundamental statistic. However, the argument does not hold true even for the simplest form of temporal network [6]. Simple temporal feature such as time of node and edges first appearance, time interval of node/edges presents from beginning to the end under some conditions are more important factor concerning nodes dynamic and evolution of graph structure.

Using data-driven approach, collected data usually contains too few data point to accurately measure temporal structure. Moreover, temporal structure, such as link burstiness, between node often has a fat-tailed distribution which is a problem when average over the value and most link occurs too little to be good representation of burstiness. Hence, we want to emphasize that quality of dataset that machine learning and deep learning models are trained on need to be improved by controlling quality over temporal properties of collected data. [6]

In general, performance between dynamic network based models are compared based on two main tasks link prediction and node classification. This is because these tasks are down-stream task that can be tested on off-the-shelf approach. In static graph, link prediction task goal is to predict existence of pre-existence edges. On the other hand, according to Barros et al. [1], link prediction on dynamic graph task can be categorized into temporal link prediction and link completion. Similar to link prediction on static graph, link completion predicts existence of pre-existence edges at timestep t. Temporal link prediction task, on the other hands, predict new edges. In this paper, we evaluate models on temporal link prediction tasks.

Dynamic node classification are less common compared to dynamic link prediction. This is because popular dataset for dynamic network tasks doesn't consider node labels. Commonly used dataset (within deep learning on dynamic graph domain) such as Reddit data provided node labels, but it is highly imbalance. Reddit data is used in the paper. In Dataset section, we will discuss the reproducible approach to create node labels for Reddit data. In attempt to solve general dynamic graph tasks models were evaluated on common tasks which includes link prediction (either temporal link prediction or link completion prediction) and dynamic node classification. Even when models were proposed to solve specific applications. It is still necessary that these papers provide evaluation on these common tasks. As a first step, dynamic network models literature extends existing static network models.

[This is confusing to write because any DGNN can be used for link prediction tasks. What do I need to add here exactly. I move all of the dynamic graph solution to related-work/dynamic graph modeling] So far, we have mentioned design aspects of dynamic graph models that are overlooked by community of deep learning on dynamic graph. In addition to that, relevant literature on the topics still uses simply train-test split to evaluate models performance. Dynamic graph is a sequential data and it is more appropriate to be evaluated with sliding window approach. Sliding window evaluation is a well known technique that is a gold standard for evaluating sequential data such as time series data. Furthermore, we found that models capacity directly depends on sliding window parameters such as window size, epoch per window etc. Therefore, without adopting sliding window evaluation as a standard to evaluate performance of dynamic network, one cannot create a fair environment to compare performance between dynamic network based models [18]. For this reason, we adopt window sliding window evaluation to evaluate temporal link prediction and dynamic node classification. The paper analyzes variations of ensemble approaches which can only be done when consider parameters from sliding window evaluation. Considering temporal factors mentioned above, the paper compare, analyze, establishes a generalized approach to implement ensemble models for dynamic graph models. [What are brief results of the proposed design?] It is not yet clear to me what I should write for this.

II. Related Work

A. Static Graph Modeling

At the beginning of graph based deep learning model, literature has tried to generalized convolution filter by generalized CNN grid filter for graph input. This only works for specific kind of graph that modified CNN grid is designed for. Another way to explore convolution filter is to convert graph from graph domain into frequency domain or Fourier domain. Filter in Fourier in domain is called spectral filter proposed by

Defferard et al. [3] by using K-localized convolutional neural network on graph. Based on [3], Kipf el at. [10] purposed GCN where K is 1 and approximation of convolution filter is not learned by neural network instead filters parameters are calculated with Chebyshev approximation. GCN is one of the first GNN architecture that successfully applied as semi-supervised model. Downside of GCN is designed for transductive setting because graph Laplacian is known during the training. GraphSage [5] solves the problem by using generalized neighbor information aggregation function (message passing framework), instead of diffuse information to neighbor with graph Laplacian. This also helps reduce overfitting. Models using graph Laplacian or alike such as GCN is called spectral-based GNN while models that use neighbor aggregation function is called spatial-based GNN.

B. Dynamic Graph

1) Taxonomies of Dynamic Graph: At the time of writing, multiple taxonomies of dynamic graph models has been proposed. In this related work section, we will discuss previous attempts to categorize dynamic graph models into groups. Before discussing previous attempt, one should understand types of dynamic behavior that can affect dynamic graph models. There are two types of dynamic behaviors which are referred to in referenced literature by different names, nonetheless, we will refer to the two types as "dynamic behavior on graph" and "dynamic behavior over graph". One can think of dynamic behavior on graph as communication between nodes that happens via edges. Dynamic behavior over graph can be think of as changes of graph as a whole over time. Intuitively, "dynamic behavior on graph" concerns micro (node/edges) levels while "dynamic behavior over graph" concern macro level — concern graph as a whole. An example to emphasize on the difference, given that there exist a group of individuals, Evolution of individuals (nodes) "role" depends on when and how they interact. At the macro level, a member of a group may leave and join. This behavior also depends on time interval that experiment considers.

Furthermore, design of models directly depend on dynamic behavior involved in dynamic graph. Hence, due to the factor mentioned above, it is very important to create an environment that is fair to make comparison between dynamic graph models. In addition to factor mentioned above, there are other factors that directly influence behavior on/over a graph including size of graph, node scale, etc, which beyond the scope of the paper. Empirical experiment has shown that combination of factors previously mentioned produces different temporal characteristic of dynamic graph either on/over the graph e.g. bustiness property [7] among other.

Barros et al. [1] categorized dynamic graph based on output embedding, model approaches, and dynamic behavior over graph. On the other than, Kazemi et al. [9] discuss in-depth mathematical formulation of encoder-decoder. The discussion also cover other types of models that are more specialized such as dynamic knowledge graph and spatio-temporal graph.

Skarding et al. [17] takes interesting approach to categorized dynamic graph based on edges duration into interaction networks, temporal networks, evolving networks, and strictly evolving networks. Furthermore, the paper classifies dynamic network models into statistic models, stochastic actor oriented models, and dynamic network representation learning model. In comparison, Skarding et al. [17] and Kazemi et al. [9] provides two different ways to categorize dynamic graph models. In contrast to Kazemi et al, Skarding et al. focus mainly on taxonomies of dynamic graph neural network including pseudo-dynamic model, edge-weighted model, discrete model, continuous models.

Note that meaning of temporal networks is ambiguous outside of skarding et al's paper [17] context. In "Temporal Network" paper, Holme et al. [7] introduce "time-respecting" path as a property of temporal network. Graph with time-respect path contains edges whose weight value represents time when edges forms. We will adopt taxonomy presented in [17] because including adopting temporal network definition. This is unambiguous because time-respecting path has not explored at all in the machine learning at the time of writing. Furthermore, all types of dynamic graph can be represented as a form of multilayer graph. [12]

2) Dynamic Graph Modeling: Before designing dynamic graph models, one must consider construction of dynamic graph input based on a given dataset. Then, models can be designed on top of constructed input. Dynamic graph construction is out of scope of this paper, but it is important to emphasize that model architecture is heavily dependent on input. Example of input graph construction are aggregated graph (edge-weighted graph [14]), synthetic link between static graph [8], and different graph. When designing dynamic graph models, one must consider node dynamic, link duration, and temporal granularity. Node dynamic concerns presents of nodes. Link duration concerns presents of edges, and temporal granularity concern either discrete or continuous occurrence of events [9].

History of deep learning solution of dynamic graph models can be traced back to 2016. At the time, literature explored methods of aggregating information on graph from node neighbor with varying weight, such as using tree like structure for NLP tasks and grid like structure. Furthermore, RNN had been used to learn temporal features while structure features are learned by CNN, GNN, or random walk. This can be done either by simply stacking temporal layer to structure layer or integrate temporal and structure components in to one layer [16]. Note that 2016 is around the peak of RNN hype. Around the same time, research effort was put toward the development of convolution filters. Later, Xu et al. [23] purposed G-GCN. The models disregard time and take into consideration only topology changes. This is done by extending variational Graph Autoencoder (VGAE) [11] to predict unseen node. Dynamic graph models that disregard time components such as G-GCN are considered as pseudo-dynamic models [17].

More modern solution of dynamic graph modeling develop around dynamic graph neural network (DGNN) architecture. Early development of DGNN models use stack of aggregated static graph as input. The aggregated static graph are represented as weighted graph. Gu et al. [14] constructs dynamic graph input into stack of weighted static graph by aggregating

graph within fixed interval and feed the input to modified GCN model. More recently, models concerning continuous temporal granularity was purposed, in particular, continuous dynamic graph neural network (continuous DGNN). Continuous DGNN updates information for every time an event (edge instance) occurs [17]. Continuous DGNN take temporal difference, time interval between event, as input features. Time point process based DGNN is a continuous DGNN that use neural network to approximate point process parameters. Alternatively, neural network can be used to encode temporal pattern by learning representation of time embedding vector. This approach is called time encoding DGNN. TGAT was the first continuous DGNN to encode time by utilizing functional time embedding similar to time2vec. TGAT use information retrieval based attention which is parameterized by query, key, value — first proposed by transformer [20]. TGN [15] adds memory module to TGAT. Our ensemble models is build on top of TGN.

C. Sliding Window Evaluation

In the time of writing, dynamic graph model literature still uses simple train-val-test split as a model evaluation standards. We provide examples of well accepted paper to make a point. Tian et al. [19] uses train-val-test split to evaluate self-supervised learning on strictly evolving graph and compare with models. Performance of models are evaluated based on two tasks: link prediction and node classification. The comparison is limited to static graph models, and dynamic random walk. Details to extend static graph models to dynamic graphs are not discussed. Similarly, using the same dataset and train-val-test split, Rssi et al. [15]. Rossi et al. compares its own, temporal graph neural network (TGN) to one other dynamic graph, DyRep. The comparison is acceptable because the same dataset is used in the experiment. Dataset used in mentioned papers is presented as undirected interaction network.

To the best of my knowledge, Skarding et al. wrote "BENCHMARKING GRAPH NEURAL NETWORKS ON DYNAMIC LINK PREDICTION" [18] which is the only paper to compare dynamic network based models using sliding window evaluation. Directed and undirected interaction network is used. Interaction network can be easily aggregated to form "graph snapshot." Hence, using interaction network, one can pass in continuous network to continuous model and discrete network to discrete models. Performance of each model varies across metric score. Hence, the paper concludes that optimizing the hyperparamters are essential for obtaining a representative score. Furthermore, Skarding et al. observes that using window of size 5 or 10 consistently produce best results particularly among discrete models.

III. Approaches

A. Sliding Window Evaluation

dynamic graph diagram

Figure **??** illustrate, at the bottom, dynamic graph diagram and, at the top, snapshot of a dynamic graph at specific timestep. This version of dynamic graph diagram is useful for visualizing sliding window on dynamic graph setting which we discuss in this section. In Figure **??**, Dynamic graph diagram box denotes edge/node deletion and edge/node creation notation while the dynamic graph box at the top illustrate events that happens to the dynamic graph. Node creation is represented as the beginning of the thick horizontal line while the end of the line represent node deletion. Horizontal lines represent edges creation event. Arrow head of the horizontal line is direction of flow of edge event. No arrow means event can flow both way. Existence of edge event are represented as length of horizontal line attaches to horizontal line.

Sliding window approaches turn any time series dataset into a supervised learning problem. Given that an instance in a dataset is an event with timestamp, train-test-split are a subset of sliding window where only one window is used for training and prediction. Consider dynamic networks observed at discrete time steps, $1, 2, ..., T$, the model is trained model on window w_t where $t = 1, 2, ..., T-1$ to predict score of $w_{\hat{t}}$ where $\hat{t} = 2, 3, ..., T$, respectively. Because temporal properties of time window, w, depends on window size, ws, and interval of time, Δt, evaluating performance based on sliding window approach evaluate model's performance under various temporal conditions which depends on temporal properties such as temporal frequency, seasonality, cycles (business cycles, economy cycle, war, etc), serial correlation.

Sliding window is specially important in dynamic based graph when applying ensemble models on top of dynamic graph models, as we will show later, overall performance depends on size of window, number of epoch per window, number of windows, number of batch per window, number of window, and time budget. Furthermore, sequence of windows allows one to apply a higher level of abstraction over sequence of events which may influence models design. Therefore, comparison between dynamic graph models are not fair without considering sliding window parameters and dynamic graph model parameters together.

It is very important to understand that how DGNN update events — stream data, one instance at a time, or in batch — imply type of DGNN where continuous DGNNs train an event at a time while discrete DGNNs train batch of events at a time.

B. Temporal Graph Neural Network (TGN)

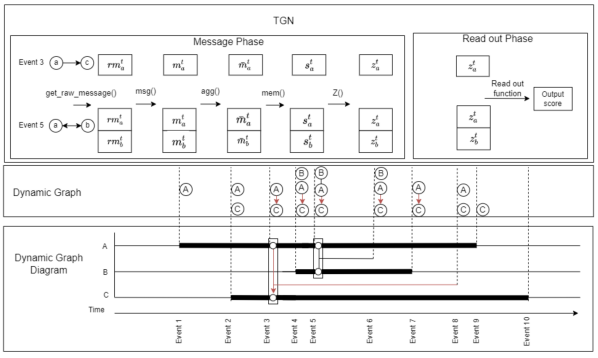

Fig. 1. TGN implementation

Our ensemble model is based on TGN [15]. Conceptually, TGN can be think of as encoder-decoder model where encoder

maps dynamic graph to node embedding while node embedding to task specific prediction, such as node classification, and link prediction. To put it simply, TGN models dynamic graph by updating node embedding with edges input or interaction event.

According to taxonomy from Skarding et al. [17], TGN can be either continuous node-dynamic evolving graph neural network or continuous node-dynamic temporal graph neural network. Continuous prefix indicates that TGN models over continuous time. This type of model can directly embed time via positional encoding which allows for more accurate modeling over changes during time interval. Next, node dynamic prefix indicates that TGN is an inductive models. In the other word, TGN can predict dynamic graph tasks for sequence of input involves node addition/deletion events. TGN does this by processing a batch of interactive event without information of changing graph topology, instead, information of graph topology is included in nodes embedding. Note that TGN does not have access to information of evolution of graph topology which are global (graph-wise memory). Node embedding only aware of local changes (node-wise memory). Lastly, because TGN is an inductive model, TGN can be applied to either evolving network, a dynamic graph that only has node and edges additional events, or interaction network, a dynamic graph that has node/edges deletion/addition events.

The main contribution of TGN is an introduction of memory vector embedding whose goal is to keep track of history of graph evolution. Memory is a generalized concept for continuous node-dynamic evolving/temporal graph neural network. Similar to how RNN introduces state embedding to keep track of changes of fully-connected network's hidden state over time. TGN replaces hidden embedding of message passing neural network (MPNN) with memory embedding to keep track of changes of message passing framework's message embedding over time. Lets i and j be nodes connected by incident edges where i is the target node and j is its neighbor. Message phase of MPNN [4] is formulated as followed:

$\mathbf{m}_i(t) = \mathrm{msg}_{\mathrm{t}}\left(\mathbf{h}_i\left(t\right), \mathbf{h}_j\left(t\right), \mathbf{e}_{ij}(t)\right)$

$\overline{\mathbf{m}}_i(t) = \mathrm{agg}\left(\mathbf{m}_i\left(t_1\right), \ldots, \mathbf{m}_i\left(t_b\right)\right)$

where m_i is a message of node i. $msg()$ is a message function (transition function) and $agg()$ is an aggregation function.

Replacing hidden embedding with memory embedding and add time as input, TGN modified original GCN formula as followed.

$\mathbf{m}_i(t) = \mathrm{msg}_{\mathrm{s}}\left(\mathbf{s}_i\left(t^-\right), \mathbf{s}_j\left(t^-\right), \Delta t, \mathbf{e}_{ij}(t)\right)$

$\overline{\mathbf{m}}_i(t) = \mathrm{agg}\left(\mathbf{m}_i\left(t_1\right), \ldots, \mathbf{m}_i\left(t_b\right)\right)$

$\mathbf{s}_i(t) = \mathrm{mem}\left(\overline{\mathbf{m}}_i(t), \mathbf{s}_i\left(t^-\right)\right)$

where s_i is a memory and Δt represents interval of time between previous and current interaction event (edges). Due to inductive nature of TGN, TGN must deal with staleness problem, in particular, memory staleness problem. This is because memory embedding is the only embedding responsible for learning temporal features (of nodes). As pointed out by Kazemi et al. [9], an embedding becomes stale when long time as pass without the embedding getting updated, hence, the information of the embedding is not up to date. In the other word, staleness occurs when no observation involve the node for a long time. Hence, the staleness problem is solved by applying learnable function over an interval when the nodes has been staled. TGN solve staleness by updating node embedding as followed.

$\mathbf{z}_i(t) = \mathrm{emb}(i,t) = \sum_{j \in n_i^k([0,t])} Z\left(\mathbf{s}_i(t), \mathbf{s}_j(t), \mathbf{e}_{ij}, \mathbf{v}_i(t), \mathbf{v}_j(t)\right)$

where z_i is a temporal embedding of node, i. Notice that emb is a function of target node, i, and time, t. Variation of emb are identity, time projection, temporal graph attention, and temporal graph sum. Please refer to TGN paper [15] for more information. In the paper, we will be using temporal graph attention as its shown to have the best performance.

Raw message is introduced as a parameters that provide necessary information to computer $msg()$ function which are information processed interactions up until the current iteration (current interactions are not included.) See Illustration **??** for flow of TGN implementation.

C. Ensemble Framework for Sliding Window

TABLE I
PARAMETERS SYMBOLS AND DESCRIPTIONS

	parameters	description
window parameters	w_i	i-th window
	ws	window size
	ei	i-th indices of edge
	$\|w\|$	number of window used during training
	bs	batch size for a given window where $bs < ws$
temporal parameters	s	window stride
	pn_n	predict instances that are in window that is n wi
	kl	number of window to keep as window slides for
	$total\_train$	total number of instances to be trained for
	pw	granularity of prediction. Prediction length during
ensemble parameters	C_i	i-th classifier model in ensemble
	$\|C\|$	number of models used in ensemble
	$train\_w_i$	i-th window is the first window to begin training

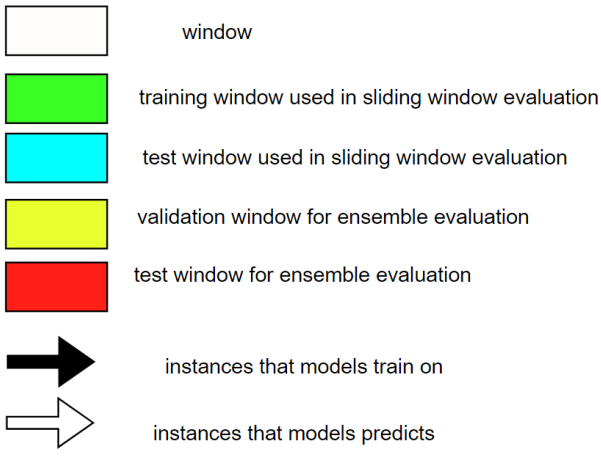

Fig. 2. symbols for ensemble framework

Ensembles benefits from combining models with diverse predictive information. Table I provides list of parameters to consider to maximize diversity of predictive information in ensemble models.

According to Table I, we categorize parameters of sliding window evaluation into the following categories: windows

Require: $data$,ws,bs,s,$|C|$, n_epoch
Ensure: $score$
1: assert ws % bs == 0
2: assert s % bs == 0
3: $w\_count = (data.size \div ws) - 1$ { -1 accounts for test set.}
4: $init\_ws = |C| \cdot ws$ { Set initial window size }
5: $i\_edges = init\_ws - 1$ { Set edges index }
6: $E =$ empty set { Initialize ensemble }
7: **for** $(count, i)$ in $enumerate(range(|C|))$ **do**
8: $(i\_edge - ((i+1) * ws), i\_edge)$ append to $E[i]$["data"]
9: $(i+1) * ws$ append to $E[i]$["data size"]
10: $E[i]$["classifier"] = empty set
11: **end for**
12: **for** w in w_count **do**
13: **for** $(count, i)$ in $enumerate(range(|C|))$ **do**
14: **for** e in n_epoch **do**
15: **for** j in $range(E[i]$["data size"$][-1] \div bs)$ **do**
16: $begin\_index = E[i]$["data"$][-1][0]$
17: $end\_index = begin\_index + ((j+1) * bs)$
18: **if** $E[i]$["classifier"] is None **then**
19: $E[i]$["classifier"$] = TGN(data[begin\_index : end\_index])$
20: **end if**
21: $E[i]$["classifier"$].train(data[begin\_index : end\_index])$
22: **end for**
23: **end for**
24: $(end\_index, end\_index + ws)$ append to$E[i]$["data"]
25: $score = E[i]$["classifier"$].predict(E[i]$["data"$][-1])$
26: $score$ append to E["ensemble"]
27: **end for**
28: $score = E$["ensemble"$]/|C|$
29: **end for**

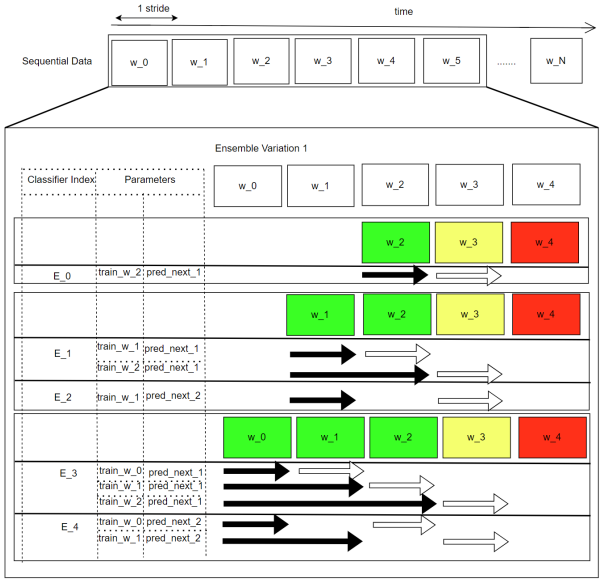

Fig. 3. Ensemble Framework for Sliding Windows

parameters, temporal parameters, and ensemble parameter. According to figure **??**, at the top, the figure illustrate sequential data the oldest event w_0 to newest event w_N where w_i is a batch of events. At the bottom of the figure in the rectangle titled "Ensemble variation 1" (Ensemble variation 2 is illustrated in figure 4) the figure illustrates five classifiers E_i where $i = 0, 1, 2, 3, 4$. Each classifier, E_i, is created from different combination parameters. Furthermore, figure **??** illustrate how $train\_w_i$ and $pred\_next_i$ work together. Figure **??** shows meaning of symbols used to illustrate ensemble framework. According to algorithm **??**, temporal parameters of each classifiers are automatically generated based on input arguments. This step is shown in algorithm **??** at line 6. Depends on input arguments, classifiers will be trained on different set of temporal parameters. According to **??**, there are two set of temporal parameters that can be set: one for sequential data, and one for ensembles. Due to large combination of parameters set, one may consider using time budget to reduce size of solution space.

IV. Dataset

Reddit dataset: Reddit dataset are a bipartite network of interaction network involving two groups of nodes: Reddit threads and users. Row of the dataset is a tuple of including user-id, thread-id, timestamp, whether user is banned after this event, and pre-compute embedding score with 172 dimensions. There are 672448 instances of interaction (aka edges) which is

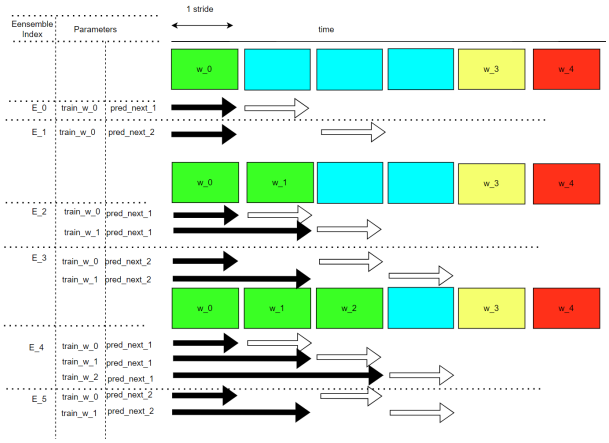

Fig. 4. ensemble variation 2

collected in one month time interval with total 11,000 nodes. Property of Reddit dataset is shown in Table II.

TABLE II
DATASETS

	Reddit
# Nodes	11,000
# Edges	672,447
# Edges Features	172
Timestapn	1 month
positive label percentages	0.05 %

V. RESULTS

REFERENCES

[1] Claudio D. T. Barros et al. "A Survey on Embedding Dynamic Graphs". In: *arXiv:2101.01229 [cs]* (Jan. 2021). Comment: 40 pages, 10 figures. arXiv: 2101.01229 [cs]. URL: https://arxiv.org/pdf/2101.01229.pdf.

[2] Borui Cai et al. "Temporal Knowledge Graph Completion: A Survey". In: *arXiv preprint arXiv:2201.08236* (2022).

[3] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. "Convolutional neural networks on graphs with fast localized spectral filtering". In: *Advances in neural information processing systems* 29 (2016). URL: https://proceedings.neurips.cc/paper/2016/file/04df4d434d481c5bb723be1b6df1ee65-Paper.pdf.

[4] Justin Gilmer et al. "Neural message passing for quantum chemistry". In: *International conference on machine learning*. PMLR. 2017, pp. 1263–1272. URL: https://arxiv.org/pdf/1704.01212.pdf.

[5] Will Hamilton, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs". In: *Advances in neural information processing systems* 30 (2017). URL: https://cs.stanford.edu/people/jure/pubs/graphsage-nips17.pdf.

[6] Petter Holme. "Modern temporal network theory: a colloquium". In: *The European Physical Journal B* 88.9 (2015), pp. 1–30. URL: https://link.springer.com/content/pdf/10.1140/epjb/e2015-60657-4.pdf.

[7] Petter Holme and Jari Saramäki. "Temporal networks". In: *Physics reports* 519.3 (2012), pp. 97–125.

[8] Amol Kapoor et al. "Examining covid-19 forecasting using spatio-temporal graph neural networks". In: *arXiv preprint arXiv:2007.03113* (2020). URL: https://arxiv.org/pdf/2007.03113.pdf.

[9] Seyed Mehran Kazemi et al. "Representation Learning for Dynamic Graphs A Survey". In: (), p. 73. URL: https://jmlr.org/papers/volume21/19-447/19-447.pdf.

[10] Thomas N Kipf and Max Welling. "Semi-supervised classification with graph convolutional networks". In: *arXiv preprint arXiv:1609.02907* (2016). URL: https://arxiv.org/pdf/1609.02907v4.pdf.

[11] Thomas N Kipf and Max Welling. "Variational graph auto-encoders". In: *arXiv preprint arXiv:1611.07308* (2016). URL: https://arxiv.org/pdf/1611.07308.pdf%5D.

[12] Mikko Kivelä et al. "Multilayer networks". In: *Journal of complex networks* 2.3 (2014), pp. 203–271.

[13] Eli Meirom et al. "Controlling graph dynamics with reinforcement learning and graph neural networks". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 7565–7577.

[14] Liang Qu et al. "Continuous-time link prediction via temporal dependent graph neural network". In: *Proceedings of The Web Conference 2020*. 2020, pp. 3026–3032. URL: https://sci-hub.se/10.1145/3366423.3380073.

[15] Emanuele Rossi et al. "Temporal graph networks for deep learning on dynamic graphs". In: *arXiv preprint arXiv:2006.10637* (2020). URL: https://arxiv.org/pdf/2006.10637.pdf.

[16] Youngjoo Seo et al. "Structured sequence modeling with graph convolutional recurrent networks". In: *International Conference on Neural Information Processing*. Springer. 2018, pp. 362–373. URL: https://arxiv.org/pdf/1612.07659.pdf?ref=https://githubhelp.com.

[17] Joakim Skarding, Bogdan Gabrys, and Katarzyna Musial. "Foundations and Modeling of Dynamic Networks Using Dynamic Graph Neural Networks: A Survey". In: *IEEE Access* 9 (2021), pp. 79143–79168. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3082932.

[18] Joakim skarding et al. "Benchmarking Graph Neural Networks on Dynamic Link Prediction". In: (2021). URL: https://openreview.net/pdf?id=I2KAe7x67JU.

[19] Sheng Tian et al. "Self-supervised Representation Learning on Dynamic Graphs". In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2021, pp. 1814–1823.

[20] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017). URL: https://arxiv.org/pdf/1706.03762.pdf.

[21] Arie Wahyu Wijayanto and Tsuyoshi Murata. "Effective and scalable methods for graph protection strategies against epidemics on dynamic networks". In: *Applied Network Science* 4.1 (2019), pp. 1–31.

[22] Arie Wahyu Wijayanto and Tsuyoshi Murata. "Learning adaptive graph protection strategy on dynamic networks via reinforcement learning". In: *2018 IEEE/WIC/ACM*

International Conference on Web Intelligence (WI). IEEE. 2018, pp. 534–539.

[23] Da Xu et al. “Generative graph convolutional network for growing graphs”. In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 3167–3171. URL: https://sci-hub.se/10.1109/icassp.2019.8682360.

[24] Haitao Yuan and Guoliang Li. “A Survey of Traffic Prediction: From Spatio-Temporal Data to Intelligent Transportation”. In: *Data Science and Engineering* 6.1 (Mar. 2021), pp. 63–85. ISSN: 2364-1541. DOI: 10.1007/s41019-020-00151-z.