Ensemble Approaches for Streaming Networking Classification

Anak Wannaphaschaiyong

I. Introduction

Dynamic graph is a vaguely used term. 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 is known in other named as streaming graph and temporal graph. Concretely, dynamic graph can be categorized into taxonomies. A few embedding dynamic network surveys have attempted to provide these taxonomies [**barrosSurveyEmbeddingDynamic2021**, **kazemiRepresentationLearningDynamica**, **skardingFoundationsModelingDynamic2021**].

Many 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. This phenomenon is observed all the time. For example, relationship between species in an ecosystem and communication between computer in distributed computing, among others.

More related to the field of machine learning, dynamic graph modeling is still under explored, but its applications has been explored more in the past few years. Applications involve tasks such as graph protection, link prediction, entity/relation prediction, time prediction, node classification, and node regression, just to name a few [**kazemiRepresentationLearningDynamica**]. Actual application involves traffic prediction [**yuanSurveyTrafficPrediction2021**], recommendation system [**kazemiRepresentationLearningDynamica**], and knowledge graph completion [**cai2022temporal**]. Reinforcement learning was used to solve graph protection problems on dynamic graph [**wijayanto2018learning**, **wijayanto2019effective**]. Meirom at el. [**meirom2021controlling**] 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. [**kapoor2020examining**] applies graph neural network on mobility data, which is represented as dynamic graph data, to predict state, represented as node, cases. This tasks of task to predict node's value is called node regression task.

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 (e.g. homogenous network, heterogenous network, multilayer network). The mentioned factors provide unique challenges. These factors must be considered before model designing phase starts otherwise network based models cannot be compared fairly. Moreover, it provide mental framework to guide designing process.

Dynamic graph extends static graph to include time variables. This add more degree of freedom to the problem. Additional degree of freedom include network status (how is information about network aggregated over time.), dynamic behavior on graph (communication behavior between nodes) and graph evolution. (structure,features,role evolution of a graph) Furthermore, it is empirically observed that some phenomenon only appears in dynamic network setting, such as bustiness property.

In general, performance between dynamic network based models are compared based on two main tasks link prediction and node classification. In static graph, link prediction task goal is to predict existence of pre-existence edges. On the other hand, according to Barros et al. [**barrosSurveyEmbeddingDynamic2021**], 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. :noexport:

Currently, in the field of machine learning on dynamic network, simply train-test split is used to conclude models performance. This is not a good idea because dynamic network data is a sequential data. It is more appropriate to use sliding window evaluation. Sliding window evaluation is a well known technique that is a gold standard for 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. For this reason, in this paper, we adopt window sliding window evaluation to evaluate link prediction and node classification using dynamic network as input. The paper analyze multiple ensemble approaches which can only be adopted via sliding window evaluation. This provides another tool to be used within dynamic graph environment.

[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

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. [**defferrard2016convolutional**] by using K-localized convolutional neural network on graph. Based on [**defferrard2016convolutional**], Kipf el at. [**kipf2016semi**] purposed GCN where K is 1 and approximation of convolution filter is not learned by neural network rather than explicitly parameterize 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 [**hamilton2017inductive**] 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 over-fitting. 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.

[discuss history of static GNN developement after geeralization of message passing.]

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, et cetera, 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 [**holme2012temporal**] among other.

Barros et al. [**barrosSurveyEmbeddingDynamic2021**] categorized dynamic graph based on output embedding, model approaches, and dynamic behavior over graph. On the other than, Kazemi et al. [**kazemiRepresentationLearningDynamica**] discuss in-depth mathematical formulation of encoder-decoder, one of many model approaches. The discussion also cover other types of models that are more specialized such as dynamic knowledge graph and spatio-temporal graph.

Skarding et al. [**skardingFoundationsModelingDynamic2021**] 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 statical models, stochastic actor oretied models, and dynamic network representation learning model. In comparison, Skarding et al. [**skardingFoundationsModelingDynamic2021**] and Kazemi et al. [**kazemiRepresentationLearningDynamica**] 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 [**skardingFoundationsModelingDynamic2021**] context. In "Temporal Network" paper, Holme et al. [**holme2012temporal**] 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 [**skardingFoundationsModelingDynamic2021**] 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. [**kivela2014multilayer**]

2) Dynamic Graph Modeling: Before designing dynamic graph models, one must consider construction of dynamic graph input based on 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-weight graph [**qu2020continuous**]), synthetic link between static graph [**kapoor2020examining**]. 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 [**kazemiRepresentationLearningDynamica**].

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 [**seo2018structured**]. Note that 2016 is around the peak of RNN hype. Around the same time, research effort was put toward the development of convolution filters. We discuss related work on this topic in Static Graph Modeling section. Later, Xu et al. [**xu2019generative**] purposed G-GCN. The models disregard time and take into consideration only topology changes. This is done by extending variational Graph Autoencoder (VGAE) [**kipf2016variational**] to predict unseen node.

In particular, according to dynamic graph modeling taxonomy [**kazemiRepresentationLearningDynamica**], this paper concerns continuous dynamic graph neural network (continuous DGNN). Continuous DGNN update information for every time an event (edge instance) occurs. Furthermore, these type of model can use temporal difference, time invertal between event, as input parameter. Neural network component can be used to approximate point process parameters. This approach is called temporal point process based model (TPP). On the other hand, neural network can be used to encode temporal pattern by learning representation of time embedding vector. TGN falls into this category which, at the time of writing, it is the state of the art. Our ensemble models is build on top of TGN.

C. Sliding Window Evaluation

*1) **TODO** 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 kind of sliding window where you only have 1 window to train to predict the future. Mathematically, consider dynamic networks observed at discrete time steps,* $1, 2, ..., T$. *For each* $t = 1, ..., T$, *one trains 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 show model's performance under various temporal condition, such as temporal frequency, seasonality, cycles (business cycles, economy cycle, war, etc), serial correlation, hence, comparison between models are not fair without considering appropriate sliding window parameters.:* 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. In this case, sliding window evaluation must be applied to all the models involve to create a fair comparison.

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. [**tian2021self**] use 70-15-15 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, Rossi et al. [**rossi2020temporal**] also use 70-15-15 splits. Rossi et al. compare its own, temporal graph neural network (TGN) to one other dynamic graph, DyRep. The comparison is acceptable because same dataset is used in the experiment. Dataset used in mentioned papers are collected as undirected interaction network.

It is very important to understand that how models receive data — stream data, one instance at a time, or in batch — implies underlying graph type. This is because it implies existence duration of nodes and edges which is used to classify dynamic graph based on taxonomies proposed by Skarding et al. [**skardingFoundationsModelingDynamic2021**]. For detail about taxonomies of dynamic graph can be found in II-B1 section.

To the best of my knowledge, Skarding et al. wrote "BENCHMARKING GRAPH NEURAL NETWORKS ON DYNAMIC LINK PREDICTION" [**skarding2021benchmarking**] 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 is essential for obtaining a representative score. This conclusion applies for both static and dynamic graph models. Furthermore, Skarding et al. observes that using window of size 5 or 10 consistently produce best results particularly among discrete models.

III. Approaches

TABLE I
PARAMETERS SYMBOLS AND DESCRIPTIONS

	parameters	description
window parameters	w_i	i-th window
	ws	window size
	$\|w\|$	number of window used during tra
	bs	batch size for a given window wh
temporal parameters	$stride$	window stride
	$pred\_next_n$	predict instances that are in windo
	$keep\_last_n$	number of window to keep as win
	$total\_training_windows$	total number of instances to be tra
ensemble parameters	E_i	i-th model in ensemble
	$\|E\|$	number of models used in ensemb
	$train\_w_i$	i-th window is the first window to
granularity parameters	PW	granularity of prediction. Predictio

In sliding window evaluation setting, one needs to make sure proposed model and benchmark model is being tested as fair as possible. Furthermore, to extract the most benefit from ensemble models, participated models should provide diverse predictive information. Table I provides list of parameters

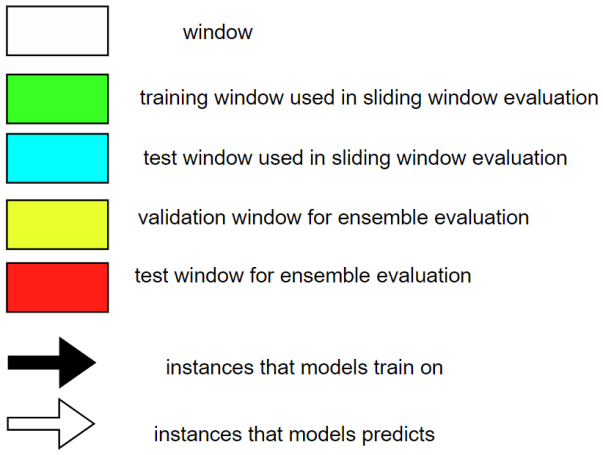

Fig. 1. symbols

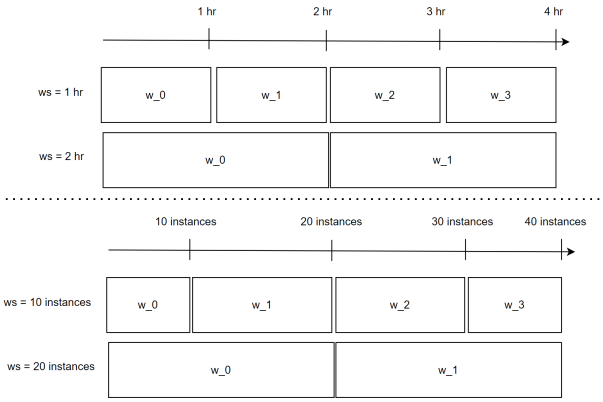

Fig. 2. window parameters

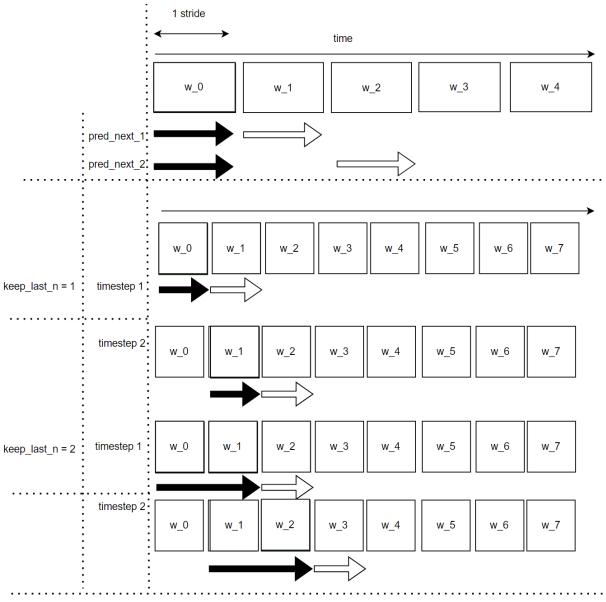

Fig. 3. temporal parameters

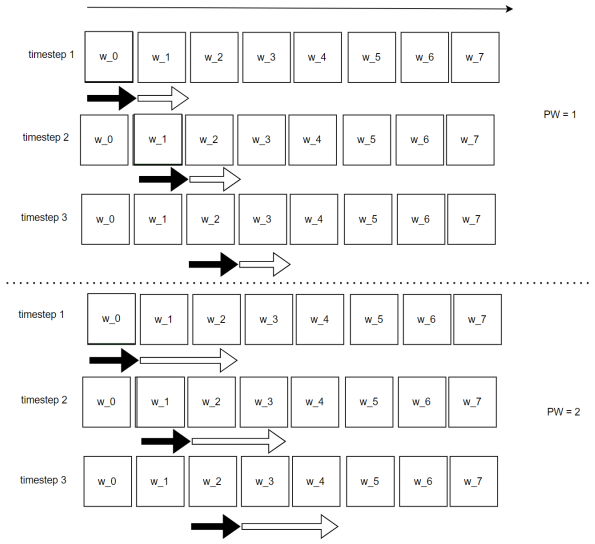

Fig. 4. granularity parameters

that must be considered to maximize diversity of predictive information in ensemble models.

According to Table I, we categorize parameters of sliding window evaluation into four categories: windows parameters, temporal parameters, ensemble parameters, and granularity parameters. Window parameters and ensemble parameters are self-explanatory, but granularity parameters and temporal parameters need clarification. Granularity is determined by prediction length during training. This parameter is important because it tells the model to minimize its mistake for certain time interval. In the other word, a model whose prediction performance is optimized over 10 days will be different to model whose performance is optimized over one day. Larger model that is trained on larger granularity ignores short term stochasticity of temporal dependencies. Illustration of window parameters, temporal parameters, granularity parameters groups are provided in Figure 2, 3, and 4. Symbols used in figures followed Figure 1.

It is important to note that temporal parameters can be applied "during ensemble formation" and "in-between ensemble formation." During ensemble formation referring to the modeling step where, given a fix set of training length, N number of individuals are trained before voting predictive score to finalize an ensemble performance. In contrast, in-between ensemble formation occurs after ensemble performance of the previous timestep is finalized and set of training instance is adjusted before it will be used to train an ensemble model of the next time step.

Using sliding window evaluation approach, there are a lot of combination of parameters that can effect model's predictive information. For this reason, 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

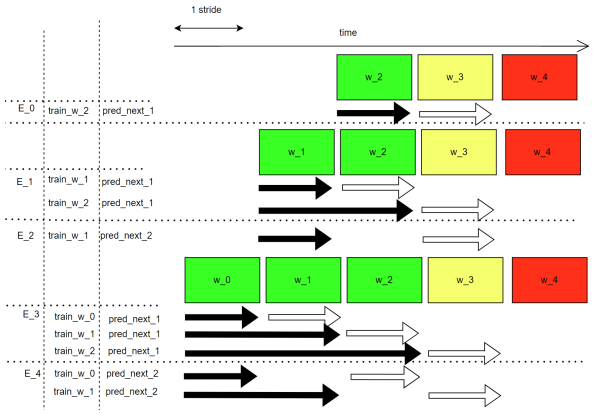

Fig. 5. ensemble variation 1

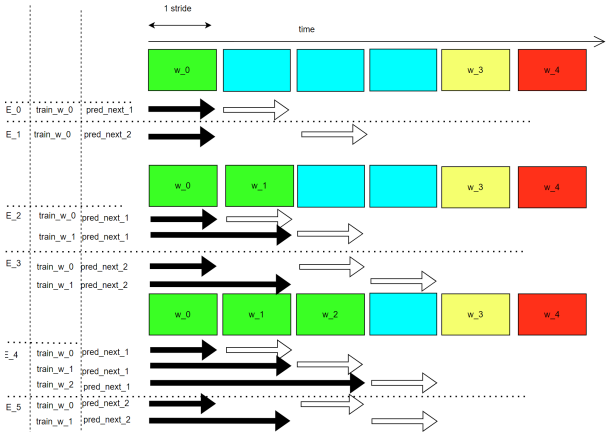

Fig. 6. ensemble variation 2

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 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