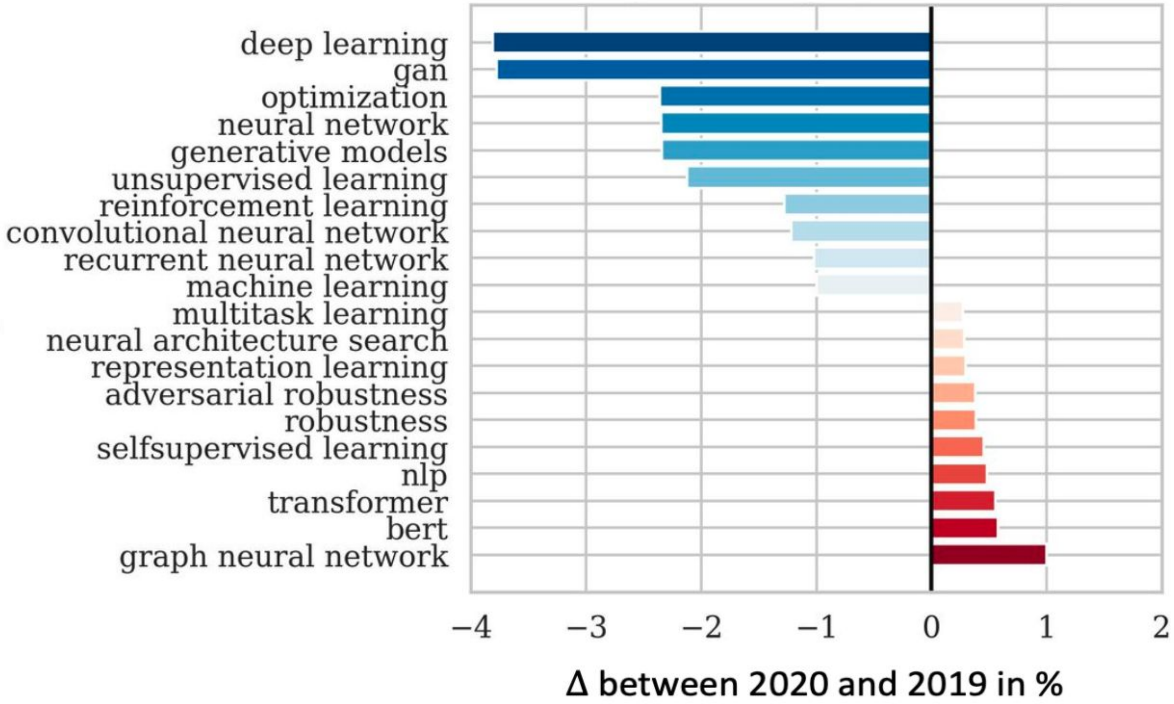
Machine Learning on Dynamic Graphs and Temporal Graph Networks

Emanuele Rossi, Twitter & Imperial College

In collaboration with Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti and Michael Bronstein

Background

Graph Neural Networks are a Hot Topic in ML!

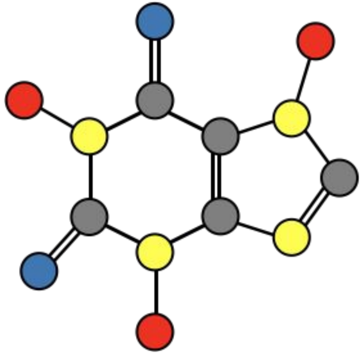


ICLR 2020 submissions keyword statistics

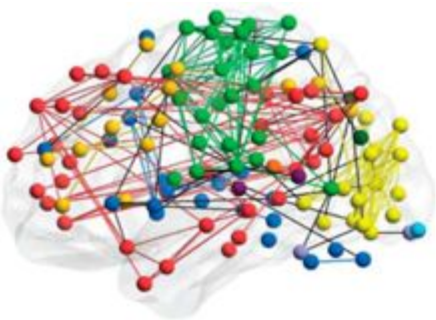
Plot: Pau Rodríguez López

Graphs are everywhere

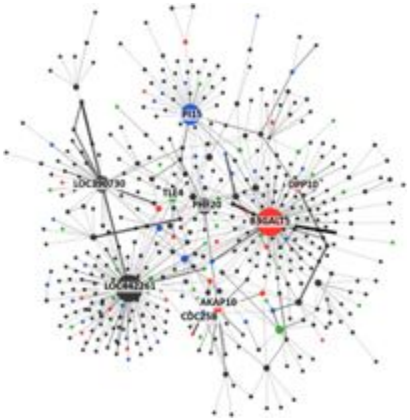
Social Networks



Molecules

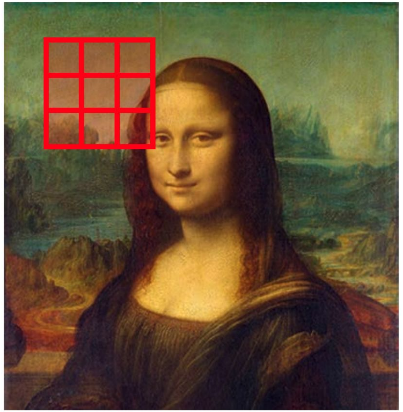


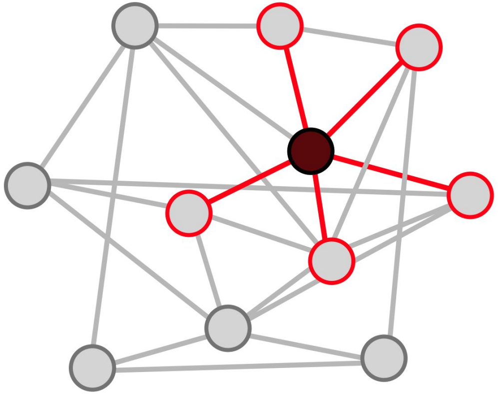
Functional Networks



Interaction Networks

From Images to Graphs





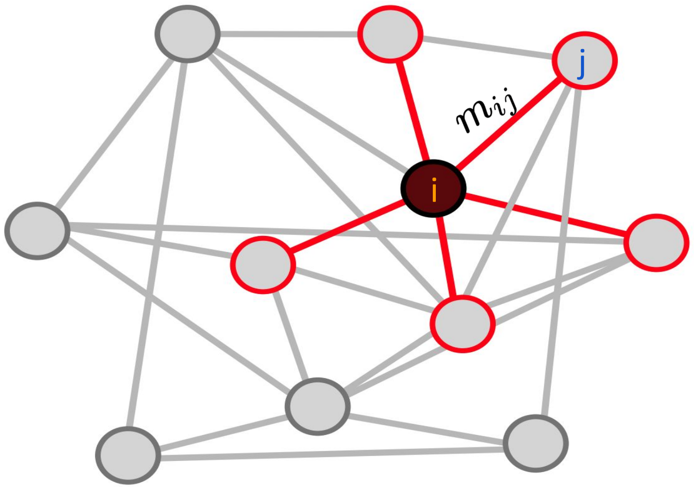
- Constant number of neighbors
- Fixed ordering of neighbors

- Different number of neighbors
- No ordering of neighbors

Graph Neural Networks

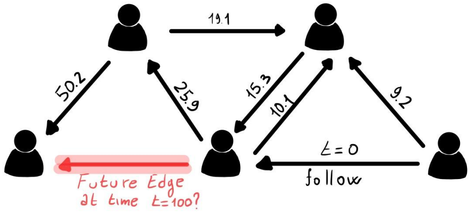
$$\mathbf{m}_{ij} = \mathrm{msg}(\mathbf{v}_i, \mathbf{v}_j, \mathbf{e}_{ij}),$$

$$\mathbf{z}_i = \sum_{j \in \mathcal{N}_i} h(\mathbf{m}_{ij}, \mathbf{v}_i)$$

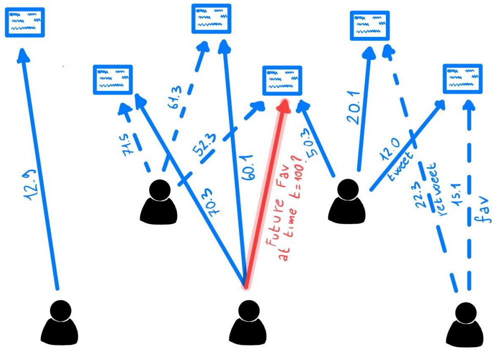


Gilmer et al. 2017

Problem: Many Graphs are Dynamic



Social Networks



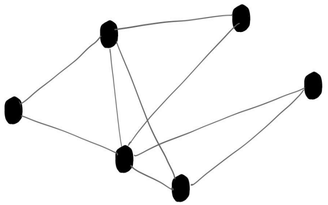
Interaction Networks

Research Questions:

- *How do we make use of the timing information to generate a better representation of nodes?*
- Can we predict when and how the graph will change in the future?
  - When will a user interact with another user?
  - Which users will interact with a given tweet in the next hour?

From Static to Dynamic Graphs

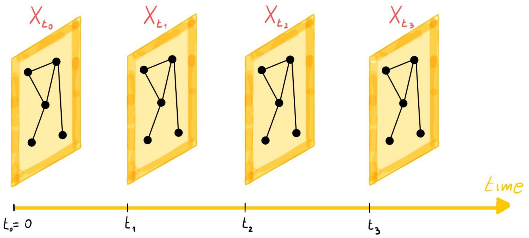
$$G = (V, E, X)$$



Static Graph

- No notion of time

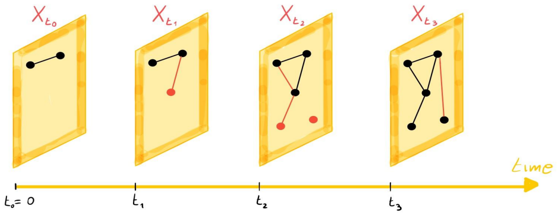
$$G_t = (V, E, X_t)$$



Spatio-Temporal Graph

- Topology is fixed, but features change over time
- (Usually) observed at regular intervals
- Examples: *traffic forecasting*, *covid-19 forecasting*

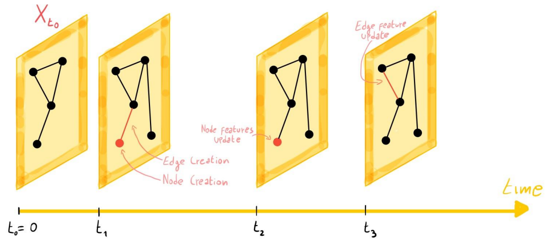
$$G_t = (V_t, E_t, X_t)$$



Discrete-Time Dynamic Graph (DTDGs)

- Both topology and features change over time
- However, graph is observed at regular intervals (no information about what happens in between)
- Examples: Any system which is observed at regular intervals

$$G(t) = \{x_{t_1}, x_{t_2}, \ldots\} \quad 0 \leq t_1 \leq t_2 \leq \cdots \leq t$$



Continuous-Time Dynamic Graph (CTDGs)

- Most general formulation
- Each change (‘*event*’) in the graph is observed individually with its timestamp
- Examples: Social networks, interaction networks, financial transaction networks

Less General → *More General*

CTDGs: Many Types of Events

	Node	**Edge**
Creation	User joins platform	User follows another user
Deletion	User leaves platform	User unfollows another user
Feature Change	User updates their bio	User changes retweet message

Why is Learning on Dynamic Graphs Different?

Model needs to:

- *Handle different types of events*
- *Use the time information of the events*
- *Efficiently and incrementally incorporate new events at test time*
- Different tasks: predict *when* something will happen

Using a static *GNN* would mean:

- *Loss of information*: Model would use the last snapshot of the graph, but not able to take into account how the graph evolved
- *Inefficiency: computation is repeated* each time we want to compute a node embedding
- *No way to do time prediction*

Model

Temporal Graph Model

$$G(t) \xrightarrow{\textbf{\textit{Encoder}}} Z(t) \xrightarrow{\textbf{\textit{Decoder}}} y(t)$$

Graph up to time *t*
(ordered sequence of events)

Temporal node embeddings

Node classifications at time *t*

Encoding a Temporal Graph

Assume our temporal graph consists only of edge creation events:

$$G(t) = \{(u_1, v_1, t_1, e_1), \ldots, (u_N, v_N, t_N, e_N)\} \qquad t_1 \leq \ldots \leq t_N \leq t$$

Idea 1:

- *Process events in order* using an *RNN*, with a *different hidden state per node*
- Final hidden states can be used as temporal node embeddings
- **Pros**:
    - Built in bias of sequentiality
- **Cons**:
    - Not using the graph of interactions directly
    - Suffer from the *memory staleness* problem

$$\mathbf{h}_u^{(i)} = \mathrm{RNN}\left(\mathbf{m}^{(i)}, \mathbf{h}_u^{(i-1)}\right)$$

$$\mathbf{m}^{(i)} = \mathbf{h}_v^{(i-1)} \,||\, \mathbf{e}^{(i)}$$

Kumar et al. 2019, Trivedi et al. 2018

Encoding a Temporal Graph

Idea 2:

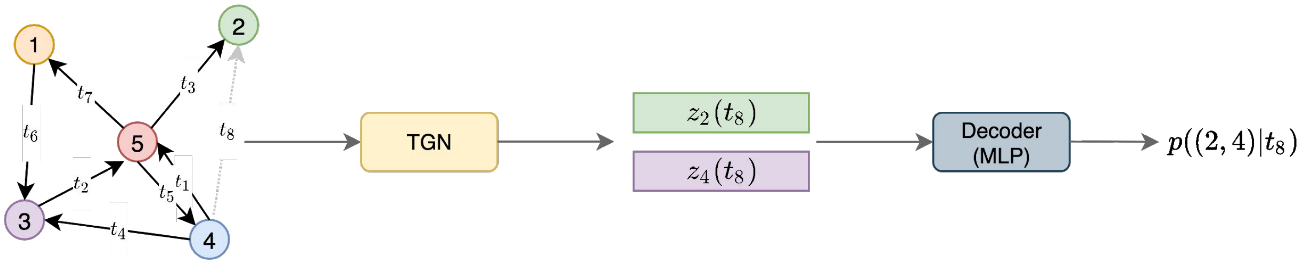
- Use a *GNN with attention* and use *timestamps as* edge *features*
- **Pros**:
  - *More efficient* as no need for sequential processing
  - Using the *graph explicitly* $\rightarrow$ *Mitigates staleness* problem
- **Cons**
  - Can *only handle edge addition events*
  - *Not suitable to online updates*

$$\mathbf{Z}(t) = \mathrm{GAT}(\mathbf{G}(t), \mathbf{E}(t), \mathbf{X}(t))$$

Xu et al. 2019

TGN: Temporal Graph Networks

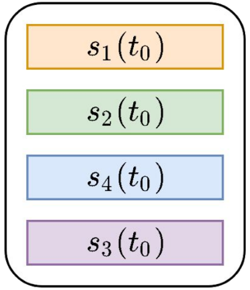
- Combines sequential processing of events with GNN
    - Handles general event types: each event generates a message which is then used to update nodes' representations
    - Uses GNN directly on graph of interaction, combining the computed hidden states with node features
- **General theoretical framework**, which consists of **5 different modules**
- Generalizes existing models such as *Jodie*[1], *TGAT*[2] and DyRep[3]



[1]Kumar et al. 2019, [2]Xu et al. 2019, [3]Trivedi et a. 2018

Modules: Memory

- Analogous to RNN hidden state, one for each node
- State (vector) for each node the model has seen so far
- **Compressed representation** of all past interactions of a node
- **Not a parameter** → updated also at test time
- Initialized at 0, it can handle new nodes (inductive)

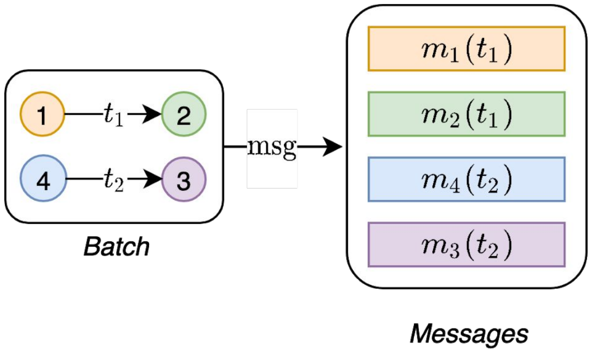


Memory

Modules: Message Function

- Each event generates a message
- **Messages** will be used to **update the memory**
- **Given an interaction** (u, v, t, e), **computes messages** for the source and the destination

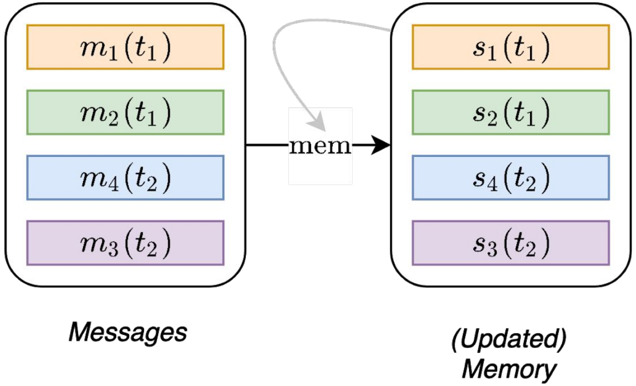
$$\mathbf{m}_u(t) = \mathrm{msg_s}\left(\mathbf{s}_u(t^-), \mathbf{s}_v(t^-), t, \mathbf{e}\right),$$
$$\mathbf{m}_v(t) = \mathrm{msg_d}\left(\mathbf{s}_v(t^-), \mathbf{s}_u(t^-), t, \mathbf{e}\right)$$



Modules: Memory Updater

- **Updates memory** using new messages

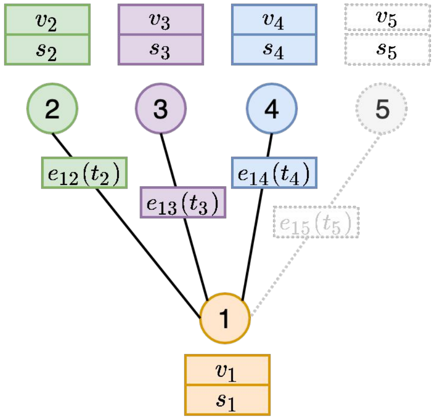
$$\mathbf{s}_u(t) = \mathrm{mem}\left(\mathbf{m}_u(t), \mathbf{s}_u(t^-)\right)$$



Modules: Graph Embedding

- A GNN which **computes** the **temporal embedding** of a node (which can be then used for prediction) using the graph and the memory
- **Solves** the **staleness problem** (memory becoming out of date)

$$\mathbf{Z}(t) = \mathrm{GAT}(\mathbf{G}(t), \mathbf{E}(t), \mathbf{X}(t), \mathbf{S}(t))$$



Link Prediction with TGN

Tasks

- *Dynamic Node Classification*
- ***Future Link Prediction***
- *Dynamic Graph Classification*

Future Link Prediction

- Data is *split chronologically*
  - Eg. if data spans 1 year → First 10 months train set, 11th month validation and 12th month test set
- *Model predicts events sequentially* (all previous events are used to predict the next one)
- We design an efficient training algorithm to speed up learning

Scalability

- **Memory** is **not a parameter** and we can just think of it as an additional feature vector for each node which we change over time
- **Only memory for nodes involved in a batch** is in GPU memory at any time
- Model is as scalable as GraphSage → **Can scale to very large graphs** (even if we don't show this in the paper)

Experiments

Experiments: Future Edge Prediction

	Wikipedia		Reddit		Twitter	
	Transductive	Inductive	Transductive	Inductive	Transductive	Inductive
GAE*	91.44 ± 0.1	†	93.23 ± 0.3	†	—	†
VAGE*	91.34 ± 0.3	†	92.92 ± 0.2	†	—	†
DeepWalk*	90.71 ± 0.6	†	83.10 ± 0.5	†	—	†
Node2Vec*	91.48 ± 0.3	†	84.58 ± 0.5	†	—	†
GAT*	**94.73** ± 0.2	91.27 ± 0.4	97.33 ± 0.2	95.37 ± 0.3	67.57 ± 0.4	62.32 ± 0.5
GraphSAGE*	93.56 ± 0.3	91.09 ± 0.3	97.65 ± 0.2	**96.27** ± 0.2	65.79 ± 0.6	60.13 ± 0.6
CTDNE	92.17 ± 0.5	†	91.41 ± 0.3	†	—	†
Jodie	94.62 ± 0.5	**93.11** ± 0.4	97.11 ± 0.3	94.36 ± 1.1	**85.20** ± 2.4	**79.83** ± 2.5
TGAT	**95.34** ± 0.1	**93.99** ± 0.3	**98.12** ± 0.2	**96.62** ± 0.3	70.02 ± 0.6	66.35 ± 0.8
DyRep	94.59 ± 0.2	92.05 ± 0.3	**97.98** ± 0.1	95.68 ± 0.2	**83.52** ± 3.0	**78.38** ± 4.0
TGN-attn	**98.46** ± 0.1	**97.81** ± 0.1	**98.70** ± 0.1	**97.55** ± 0.1	**94.52** ± 0.5	**91.37** ± 1.1

Experiments: Dynamic Node Classification

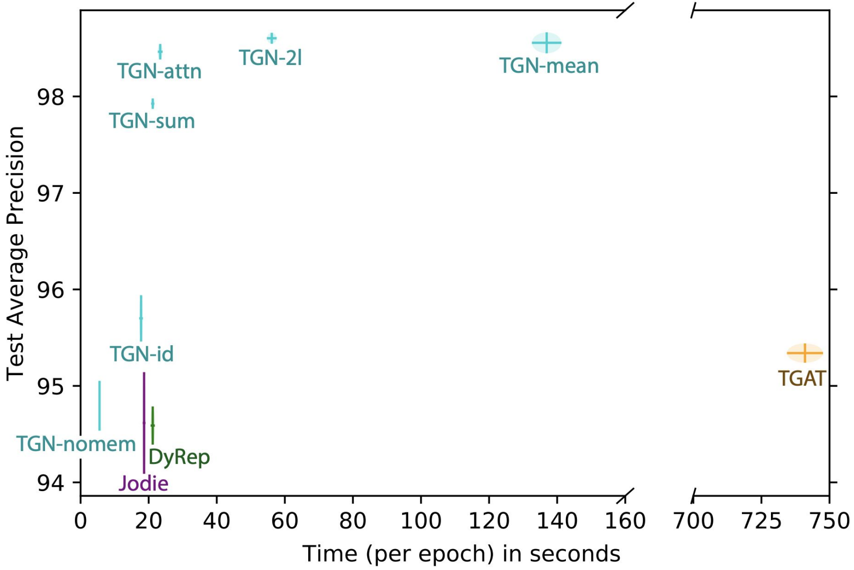
	Wikipedia	Reddit
GAE*	74.85 ± 0.6	58.39 ± 0.5
VAGE*	73.67 ± 0.8	57.98 ± 0.6
GAT*	82.34 ± 0.8	**64.52** ± 0.5
GraphSAGE*	82.42 ± 0.7	61.24 ± 0.6
CTDNE	75.89 ± 0.5	59.43 ± 0.6
JODIE	**84.84** ± 1.2	61.83 ± 2.7
TGAT	83.69 ± 0.7	**65.56** ± 0.7
DyRep	**84.59** ± 2.2	62.91 ± 2.4
TGN-attn	**87.81** ± 0.3	**67.06** ± 0.9

Ablation Study

(Future edge prediction)

	Mem.	Mem. Updater	Embedding	Mess. Agg.	Mess. Func.
Jodie	node	RNN	time	—[†]	id
TGAT	—	—	attn (2l, 20n)*	—	—
DyRep	node	RNN	id	—[‡]	attn[‖]
TGN-attn	node	GRU	attn (1l, 10n)	last	id
TGN-2l	node	GRU	attn (2l, 10n)	last	id
TGN-no-mem	—	—	attn (1l, 10n)	—	—
TGN-time	node	GRU	time	last	id
TGN-id	node	GRU	id	last	id
TGN-sum	node	GRU	sum (1l, 10n)	last	id
TGN-mean	node	GRU	attn (1l, 10n)	mean	id

- **Faster** and **more accurate** than other approaches
- **Memory** (*TGN-att* vs *TGN-no-mem*) leads to a **vast improvement** in performance
- **Embedding** module is also extremely **important** (*TGN-attn* vs *TGN-id*) and **graph attention performs best**
- Using the memory makes it enough to have 1 graph attention layer



Predicting when events will happen

- Qualitatively different question from other tasks
- A decoder which makes use of *Temporal Point Processes* is needed [3]

Applications:

- When will two users interact again?
- *How many retweets* will a given tweet have *in the next 30 or 60* minutes?

[3] Trivedi et al. 2018

Future Work

- **Benchmark datasets** for dynamic graphs (see OGB)
- **Method Extensions:** *Global* (graph-wise) *memory*, *continuous models* (eg. neural ODEs) to model the memory evolution
- **Scalability**: Propose methods which scale better (possibly combining with literature on graph sampling, but not trivial)
- **Applications**: Social Networks (eg. recommender systems, virality prediction), biology (eg. molecular pathways, cancer evolution), finance (eg. fraud detection) and more?

Conclusion

- **Dynamics graphs** are very common, but have received **little attention so far**
- We propose **TGN**, a **general encoder** for dynamic graphs which achieves **SOTA results** on a variety of benchmarks
- We design an **efficient algorithm for training** the memory-related modules
- The ablation study shows the **importance of the different modules**

Questions?

@emaros96