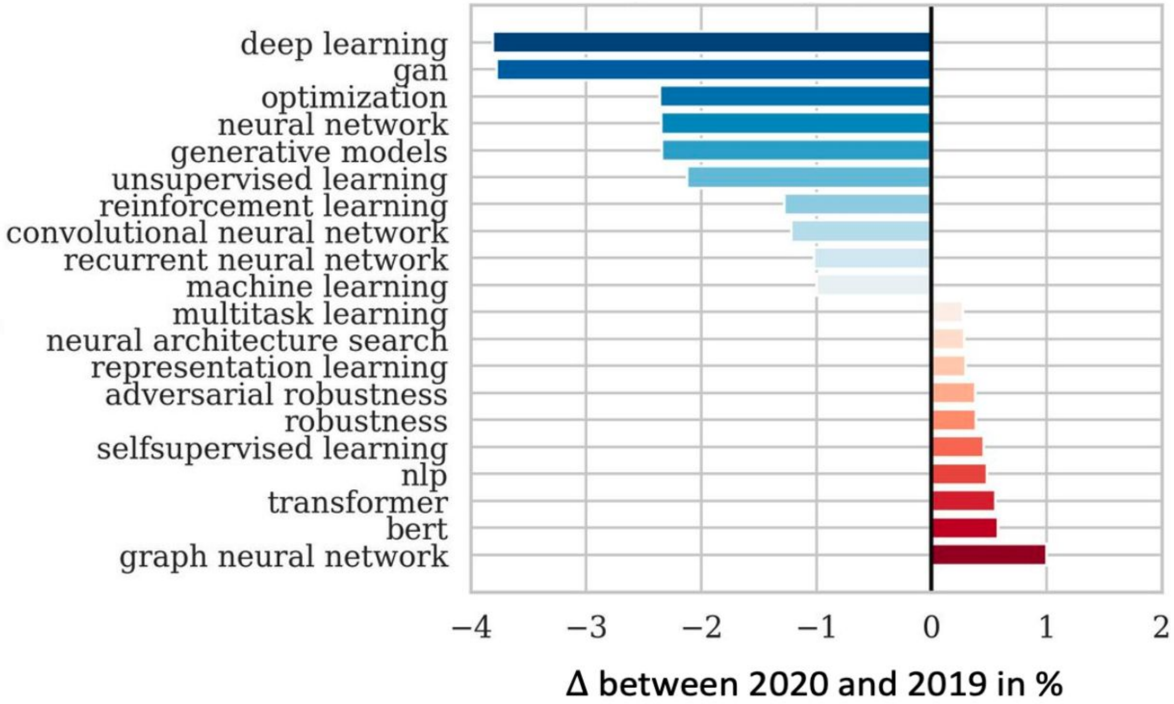
TGN: Temporal Graph Networks for Dynamic Graphs

Emanuele Rossi, Twitter

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

Background

Graph Neural Networks are a Hot Topic in ML!

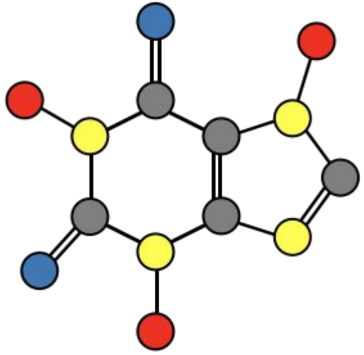


ICLR 2020 submissions keyword statistics

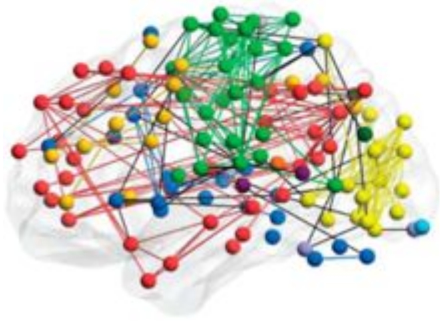
Plot: Pau Rodríguez López

Graphs are everywhere

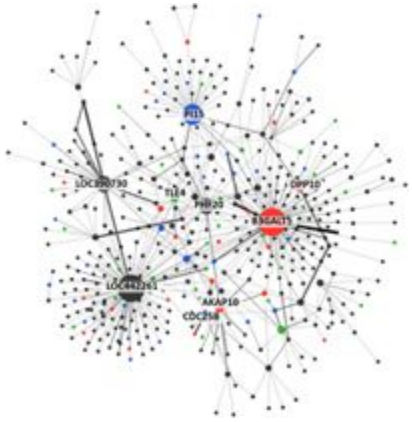
Social Networks



Molecules

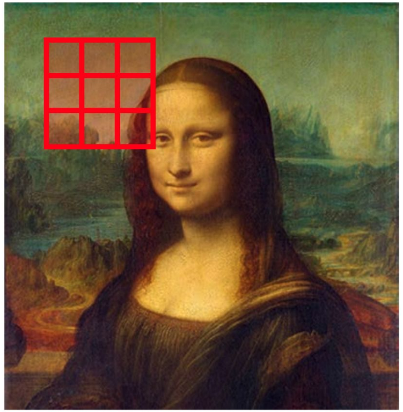


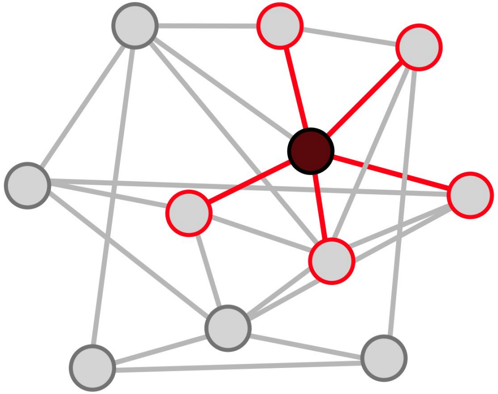
Functional Networks



Interaction Networks

From Images to Graphs



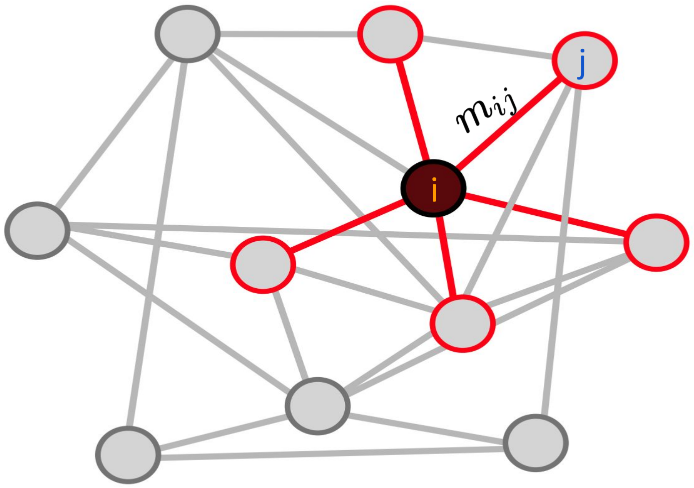


- Constant number of neighbors
- Fixed ordering of neighbors

- Different number of neighbors
- No ordering of neighbors

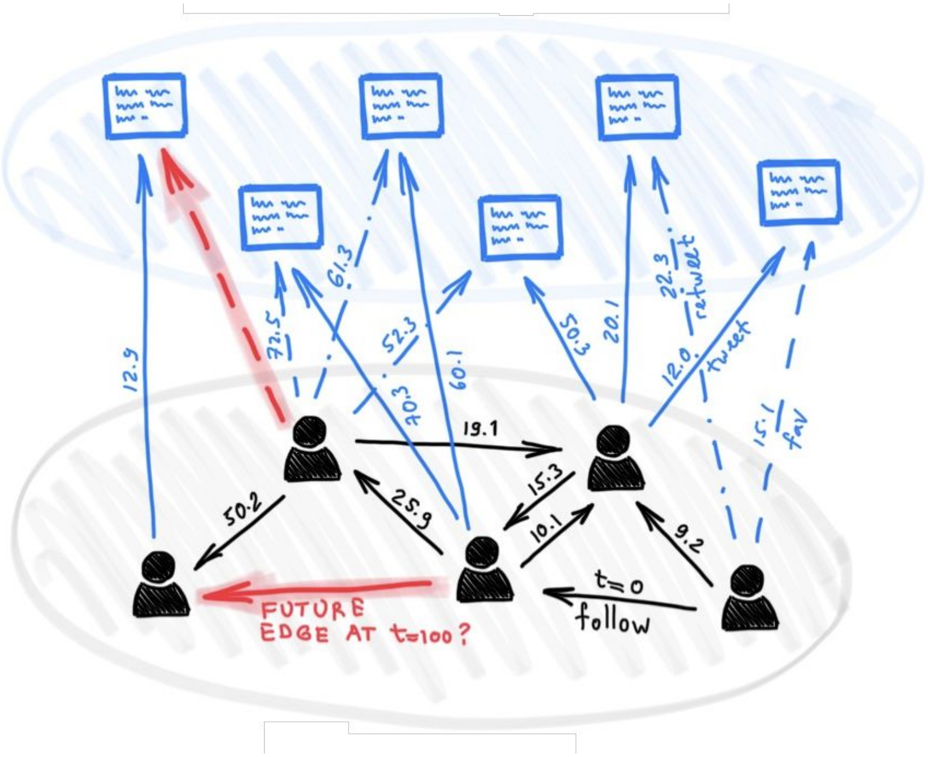
Graph Neural Networks

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

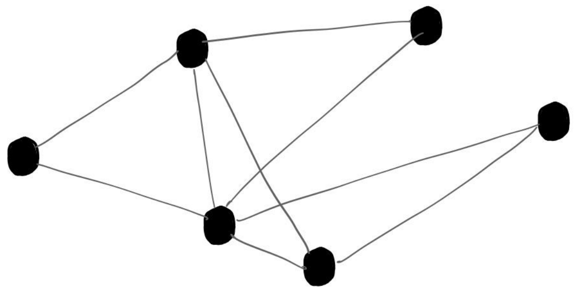


Gilmer et al. 2017

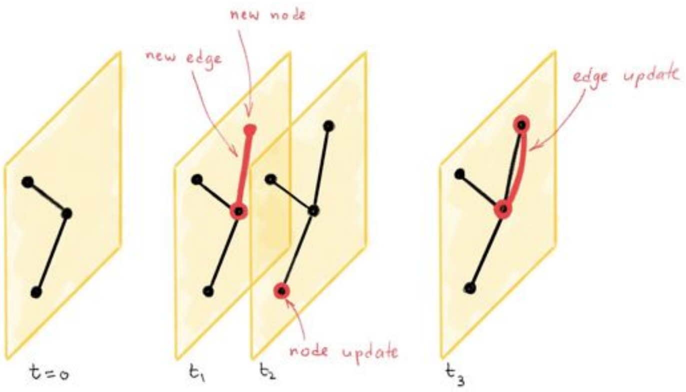
Problem: Many Graphs are Dynamic



Static vs Dynamic Graphs



Static Graph



Dynamic Graph

Why is Learning on Dynamic Graphs Different?

Model needs to:

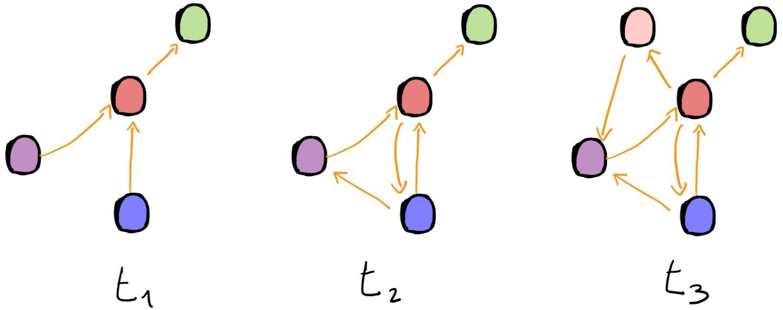
- *Support addition* / *deletion* of node and edges, as well as *feature changes*
- Make *predictions* (eg. classify a node) at *any point in time*

Using a static model (on a dynamic graph) would mean:

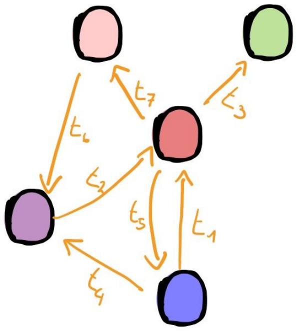
- *Inefficiency: computation is repeated* each time we want to make a prediction
- *Loss of information*: Model is able to handle the latest snapshot of the graph, but not how the graph evolved

Dynamic Graphs

- *Discrete-time dynamic graphs*: sequence of snapshots
- *Continuous-time dynamic graphs*: sequence of timed-events



Discrete-time dynamic graphs

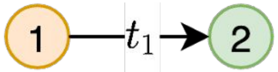


Continuous-time dynamic graphs

Learning on Dynamic Graphs

$$t_1 \leq t_2 \leq t_3 \leq t_4 \leq t_5 \leq t_6 \leq t_7$$

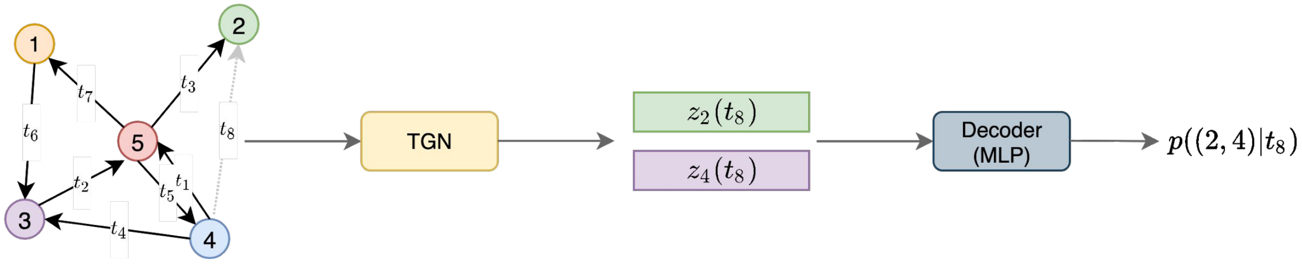
- Data is a **sequence of ordered timed events** (eg. edge addition)
- An epoch goes through the events in chronological order
- Model is **trained self-supervised**, predicting future edges using all information from previous edges



Model

TGN: Temporal Graph Networks

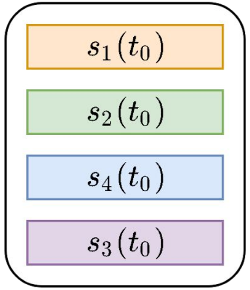
- Model for dynamic graphs is an encoder-decoder pair
- TGN is an encoder model which is able to generate **temporal node embeddings** $z_i(t) = f(i,t)$ for any node *i* and time *t*. Decoder is task-dependent, eg. MLP from two node embeddings to edge probability
- **General theoretical framework**, 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

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

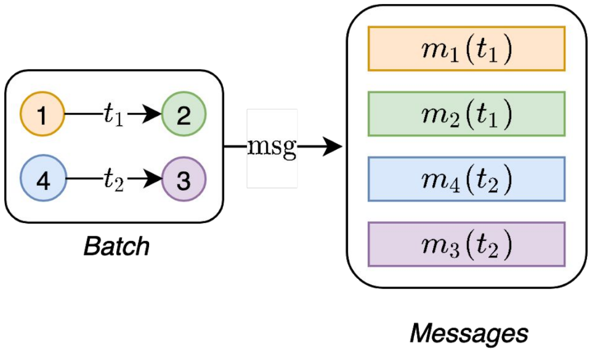


Memory

Modules: Message Function

- **Given an interaction** (i, j), **computes messages** for the source and the destination
- **Messages** will be used to **update the memory**

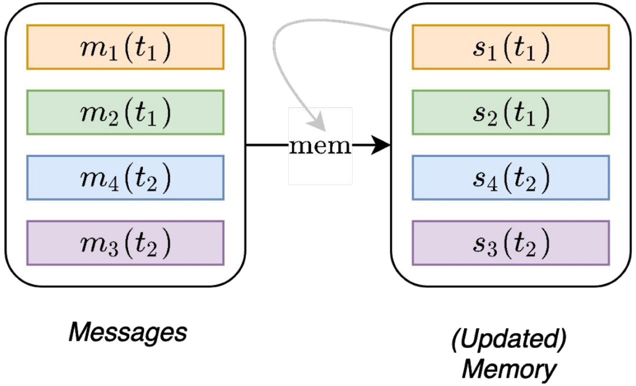
$$\mathbf{m}_i(t) = \text{msg}\left(\mathbf{s}_i(t^-), \mathbf{s}_j(t^-), t, \mathbf{e}_{ij}(t)\right),$$
$$\mathbf{m}_j(t) = \text{msg}\left(\mathbf{s}_j(t^-), \mathbf{s}_i(t^-), t, \mathbf{e}_{ij}(t)\right)$$



Modules: Memory Updater

- **Updates memory** using new messages

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

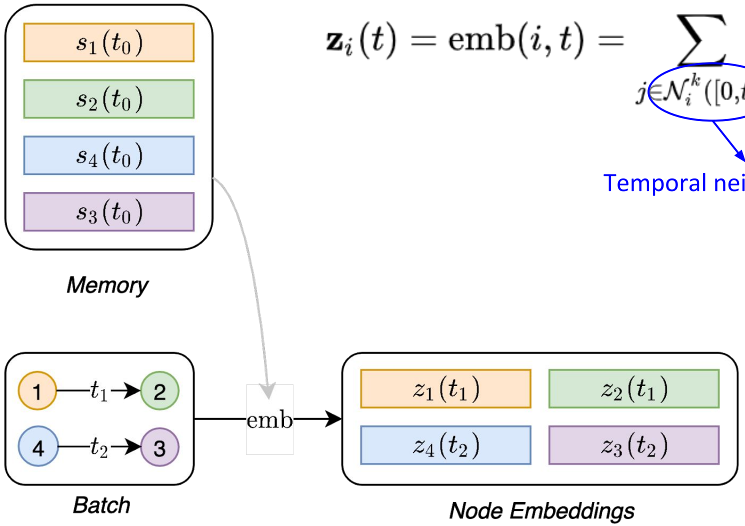


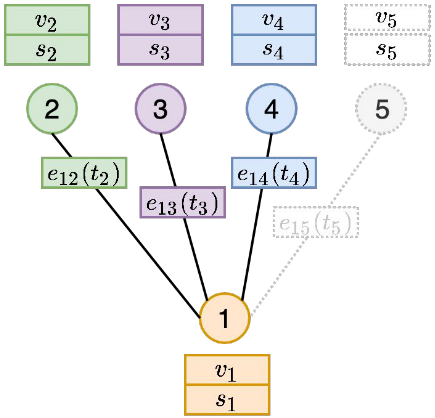
Modules: (Graph) Embedding

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

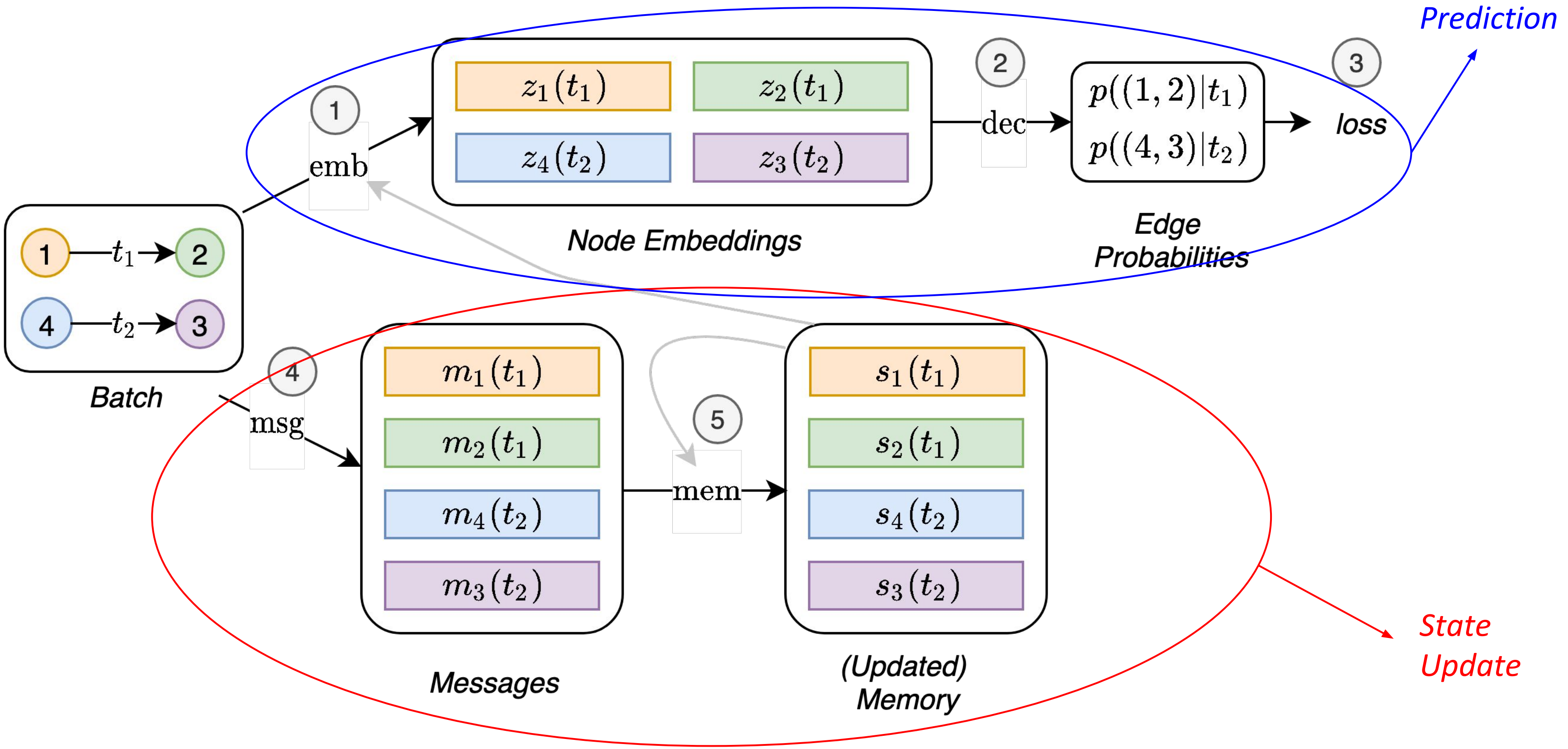
$$\mathbf{z}_i(t) = \mathrm{emb}(i,t) = \sum_{j\in\mathcal{N}_i^k([0,t])} h\left(\mathbf{s}_i(t), \mathbf{s}_j(t), \mathbf{e}_{ij}, \mathbf{v}_i(t), \mathbf{v}_j(t)\right),$$

Temporal neighborhood





TGN: Overview



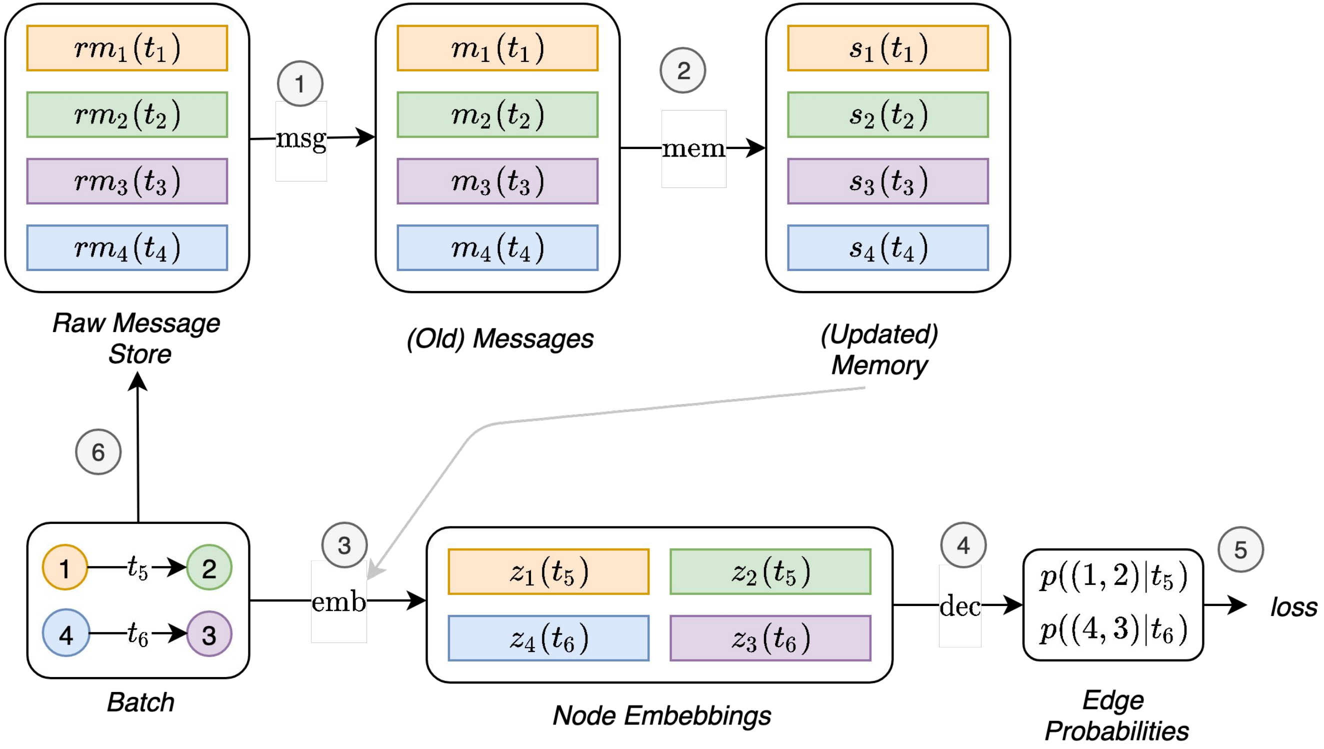
Learning TGN

- **Problem 1:** DTDGs can be seen as a sequence for each node, but the sequences are inter-dependent
  - We cannot use standard BPTT
  - Interactions need to be processed
- **Solution**: Process interactions according to a global chronological order

Learning TGN

- **Problem 2**: Memory-related modules do not directly influence the loss and therefore do not receive a gradient
    - The memory must be updated before predicting an interaction
    - However, updating the memory with the same interaction we then predict causes a leakage
- **Solution**:
    - Update memory with **messages from previous batches**

Learning TGN - Diagram



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 $\rightarrow$ **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*	$\mathbf{94.73} \pm 0.2$	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	$\mathbf{96.27} \pm 0.2$	65.79 ± 0.6	60.13 ± 0.6
CTDNE	92.17 ± 0.5	†	91.41 ± 0.3	†	—	†
Jodie	94.62 ± 0.5	$\mathbf{93.11} \pm 0.4$	97.11 ± 0.3	94.36 ± 1.1	$\mathbf{85.20} \pm 2.4$	$\mathbf{79.83} \pm 2.5$
TGAT	$\mathbf{95.34} \pm 0.1$	$\mathbf{93.99} \pm 0.3$	$\mathbf{98.12} \pm 0.2$	$\mathbf{96.62} \pm 0.3$	70.02 ± 0.6	66.35 ± 0.8
DyRep	94.59 ± 0.2	92.05 ± 0.3	$\mathbf{97.98} \pm 0.1$	95.68 ± 0.2	$\mathbf{83.52} \pm 3.0$	$\mathbf{78.38} \pm 4.0$
TGN-attn	$\mathbf{98.46} \pm 0.1$	$\mathbf{97.81} \pm 0.1$	$\mathbf{98.70} \pm 0.1$	$\mathbf{97.55} \pm 0.1$	$\mathbf{94.52} \pm 0.5$	$\mathbf{91.37} \pm 1.1$

Experiments: Dynamic Node Classification

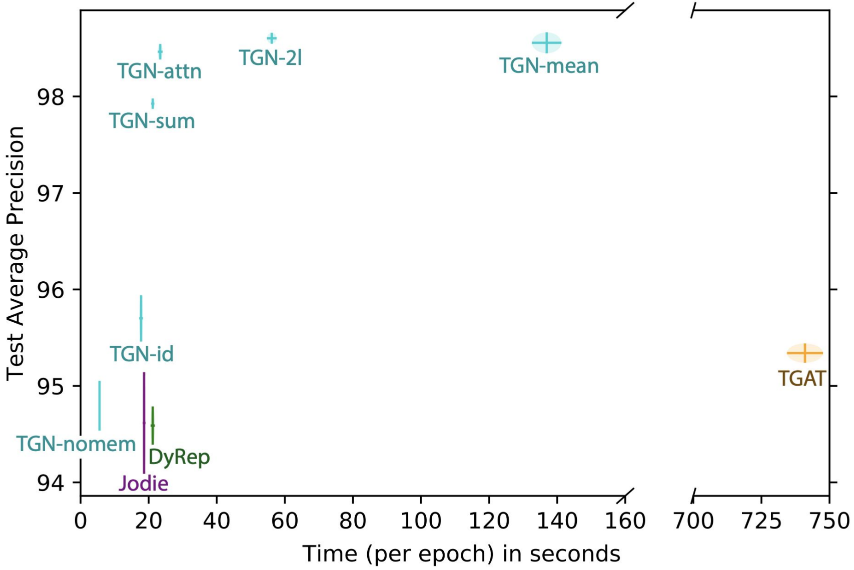
	Wikipedia	Reddit
GAE*	74.85 ± 0.6	58.39 ± 0.5
VAGE*	73.67 ± 0.8	57.98 ± 0.6
GAT*	82.34 ± 0.8	$\mathbf{64.52} \pm 0.5$
GraphSAGE*	82.42 ± 0.7	61.24 ± 0.6
CTDNE	75.89 ± 0.5	59.43 ± 0.6
JODIE	$\mathbf{84.84} \pm 1.2$	61.83 ± 2.7
TGAT	83.69 ± 0.7	$\mathbf{65.56} \pm 0.7$
DyRep	$\mathbf{84.59} \pm 2.2$	62.91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

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**
- Last message aggregator, while discarding some information, performs extremely well while being very fast (*TGN-attn* vs *TGN-mean*)
- Using the memory makes it enough to have 1 graph attention layer



Future Work

- **Benchmark datasets** for dynamic graphs (see OGB)
- **Time in ML**: Improve how we use timestamp information in ML
- **Method Extensions:** *Global* (graph-wise) *memory*, *continuous models* (eg. neural ODEs) to model the memory evolution
- **Training Algorithm**: Coming up with an even *more efficient training algorithm* for dynamic graphs
- **Scalability**: Propose methods which scale better (possibly combining with literature on graph sampling, but not trivial)
- **Applications**: Twitter, biology (molecular pathways, cancer evolution), finance (transaction networks) and more?

Conclusion

- **Dynamics graphs** are very common, but have received **little attention so far**
- We propose **TGN**, which **generalizes existing models** and 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