THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

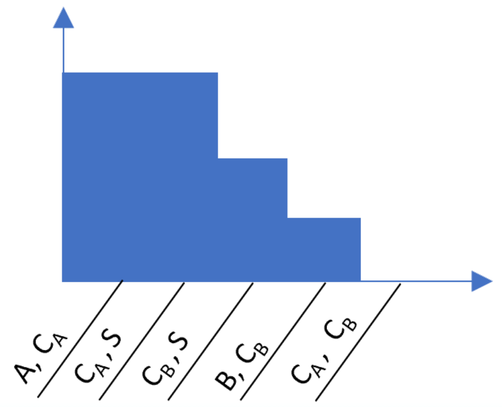
EULER: Detecting Network Lateral Movement via Scalable Temporal Graph Link Prediction

Isaiah J. King & H. Howie Huang

Network and Distributed Systems Security Symposium (NDSS), San Diego, 2022

Ways of Representing Networks

ts	src_bytes	dst_bytes	protocol	service	flag
1	231	432	tcp	http	SF
1	245	521	tcp	http	SF
		…			
5	255	1023	tcp	smtp	SF
5	126	10	tcp	smtp	REJ



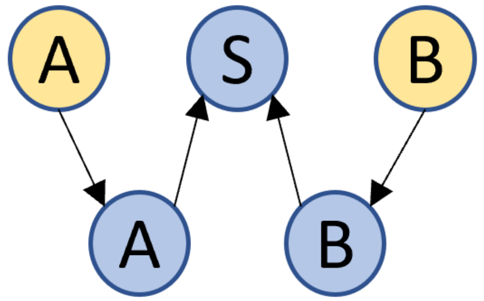
Traditionally 2 ways of abstracting network traffic:

1. Events-based
   - Anomalies defined by unusual features
   - Notably, relational data not considered (see KDD Cup data set)
2. Frequency-based
   - Anomalies defined by unusual counts of events
   - Difficult to correlate structural relationships beyond 1-hop

Ways of Representing Networks (cont.)

More recently, graph analytics have been applied:

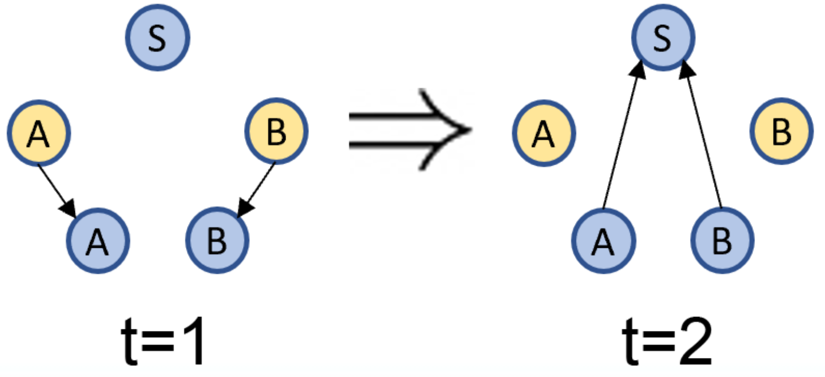
- Captures structural relationships
- Better suited for lateral movement detection
- Does not consider time, however



Solution: Temporal Graphs

Given a multiset of tuples $\mathrm{I} = < src, dst, t >$ of interactions on a network, a temporal graph with time window δ is the set

$G = \{G_0, \dots, G_T\}$ where
$G_t = \{V_t, E_t\}$ with the constraint that
$\forall\ (u, v) \in E_t,\ \exists < u, v, i > \in I\ \land\ t \leq i < (t + \delta)$



Temporal Link Prediction

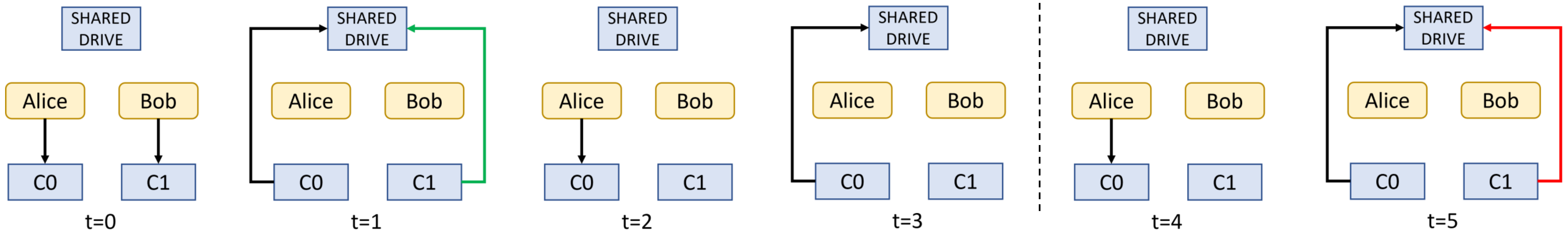
Given a temporal graph *G* the objective is to find a function

$f(\, u, v, t \mid \{G_0, \dots, G_t\}\,) \approx P[(u, v) \in E_{t+n}]$
Where $n \geq 0$

If $n = 0$, we call it **Dynamic Link Detection**

If $n > 0$, we call it **Dynamic Link Prediction**

Motivating Example



- When Alice and Bob authenticate with C0 and C1, they query the SD
- When Bob does not authenticate with C1, it does not query the SD

A simple probability distro is apparent:

$$P[\,(C1, SD) \in E_{t+1} \mid (B, C1) \in E_t\,] = 1$$
$$P[\,(C1, SD) \in E_{t+1} \mid (B, C1) \notin E_t\,] = 0$$

How do Other Data Structures Fare?

Event-based:

- If <C1, SD, 5> has similar features to <C1, SD, 3>, the anomaly is undetectable

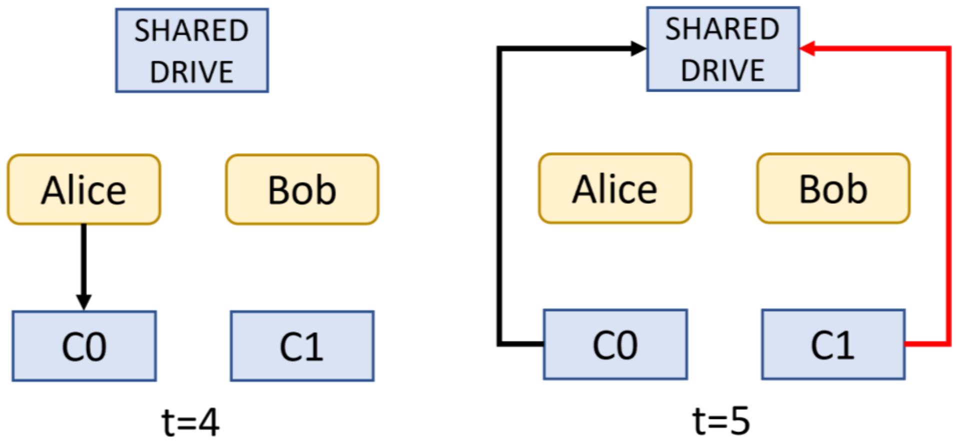
Frequency-based:

- The lack of an event with B at t=4 would have little bearing on an event between C1 and SD

Graph-based:

- P[<C1, SD, 3>] = P[<C1, SD, 5>]; time is not considered

Temporal graphs are the perfect solution to this problem, and those solved by prior works!

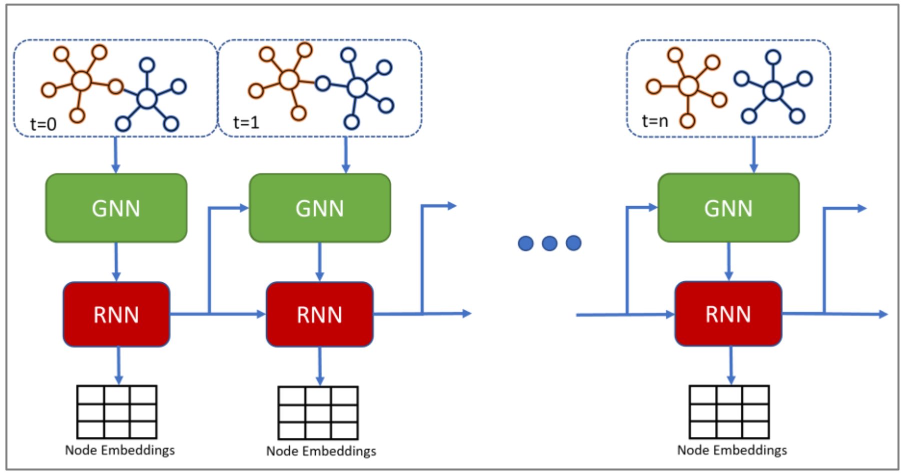


$$P[\,(C1, SD) \in E_{t+1} \mid (B, C1) \in E_t\,] = 1$$
$$P[\,(C1, SD) \in E_{t+1} \mid (B, C1) \notin E_t\,] = 0$$

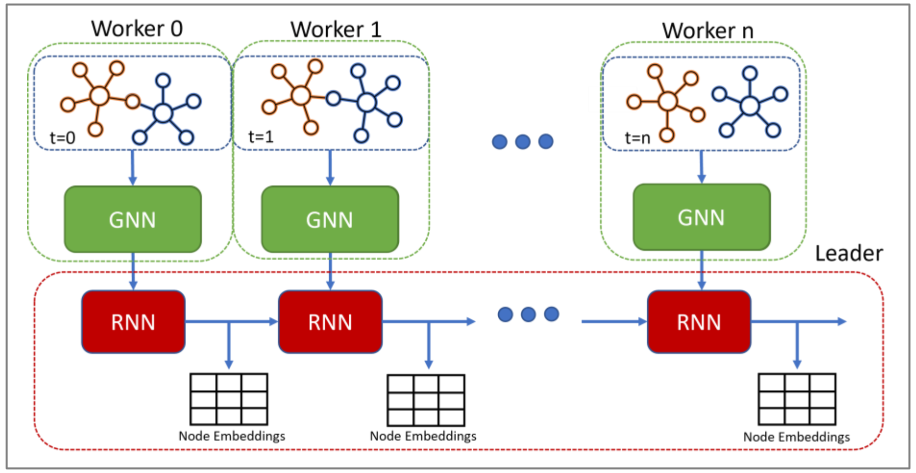
Temporal Link Prediction

- In the past, GNN output passed through a sequence encoder
- **Forces process to be sequential**
- Cannot scale to large graphs (i.e. network logs)

- We propose **uncoupling the RNN and GNN**
- GNN is most complex portion of the approach
- Amdahl's law—distribute the hard parts

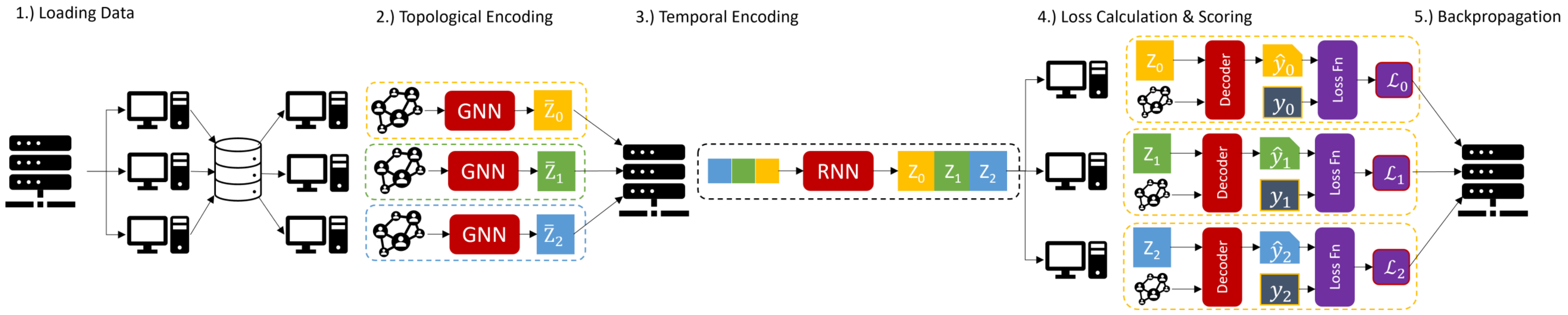


SoTA



Our Approach

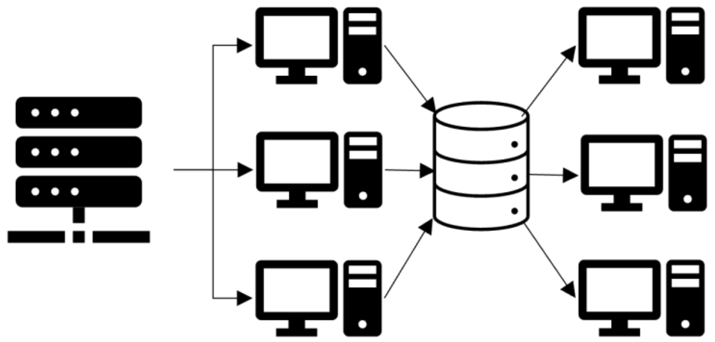
The EULER Framework

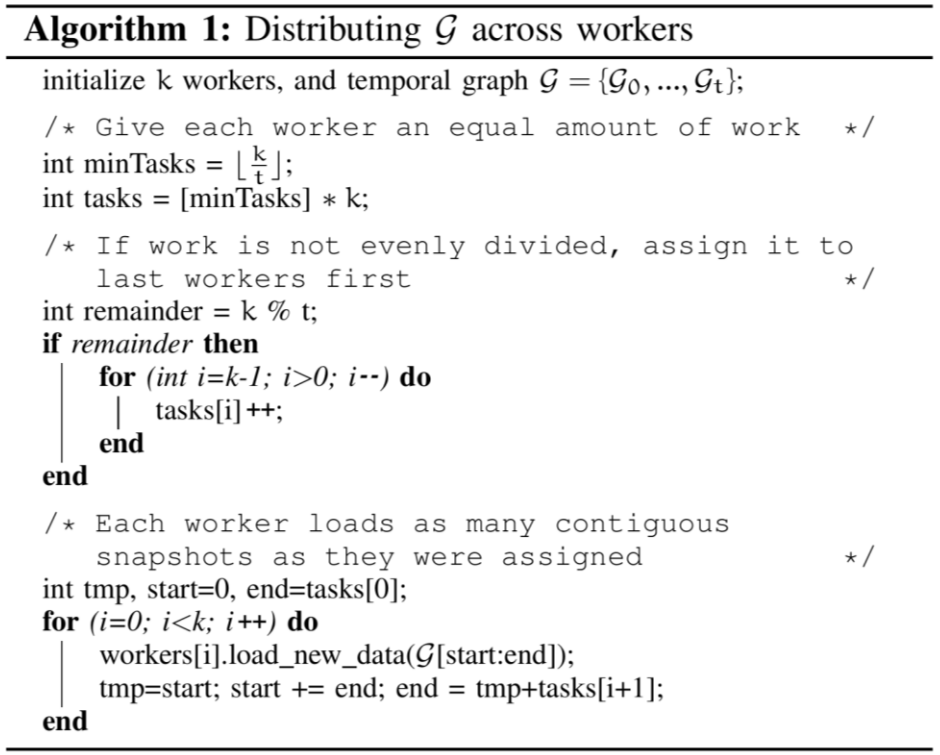


1.) Loading Data

- Leader machine spins up *k* workers
- Workers issued command to load $\left\lfloor \frac{k}{t} \right\rfloor$ disjoint, sequential snapshots
- Remaining snapshots assigned to workers holding oldest snapshots

- Read occurs in parallel
- Leader awaits workers to signal they have finished loading



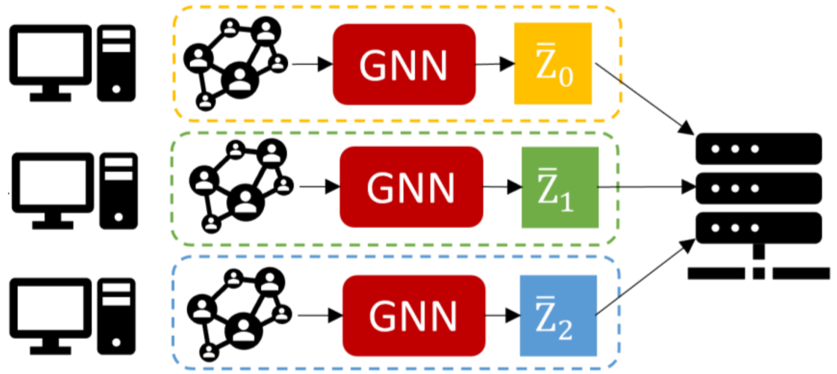


Algorithm 1: Distributing $\mathcal{G}$ across workers

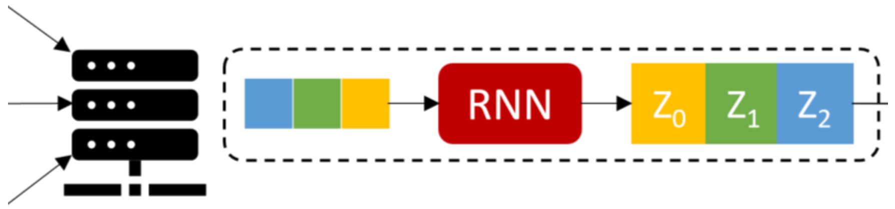
```
initialize k workers, and temporal graph $\mathcal{G} = \{\mathcal{G}_0, ..., \mathcal{G}_t\}$;
/* Give each worker an equal amount of work   */
int minTasks = $\lfloor \frac{k}{t} \rfloor$;
int tasks = [minTasks] * k;
/* If work is not evenly divided, assign it to
   last workers first                          */
int remainder = k % t;
if remainder then
    for (int i=k-1; i>0; i--) do
        tasks[i] ++;
    end
end
/* Each worker loads as many contiguous
   snapshots as they were assigned             */
int tmp, start=0, end=tasks[0];
for (i=0; i<k; i++) do
    workers[i].load_new_data($\mathcal{G}$[start:end]);
    tmp=start; start += end; end = tmp+tasks[i+1];
end
```

2.) Topological Encoding

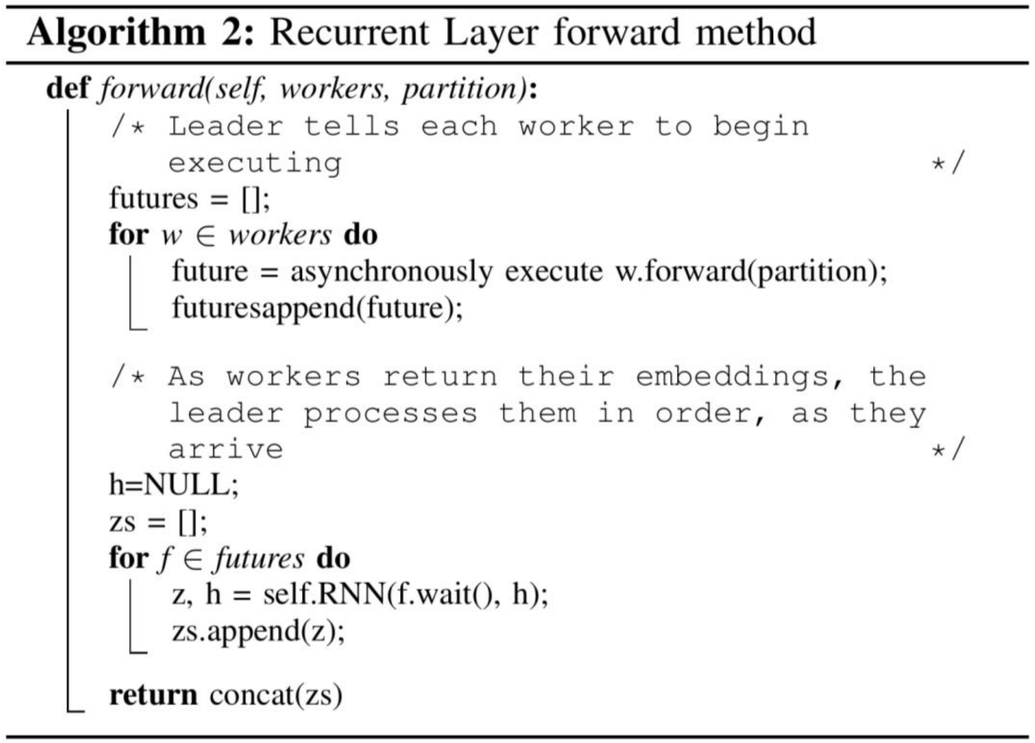
- Leader issues command to workers to generate topological embeddings
- Only argument is ENUM representing which partition (if any) of edges to process
- Workers process each snapshot they hold in parallel



3.) Temporal Encoding



- RNN must operate sequentially
- Some parallelism from the data imbalance
- Leader processes topological encodings as soon as they arrive
- Returns final **Z** embeddings



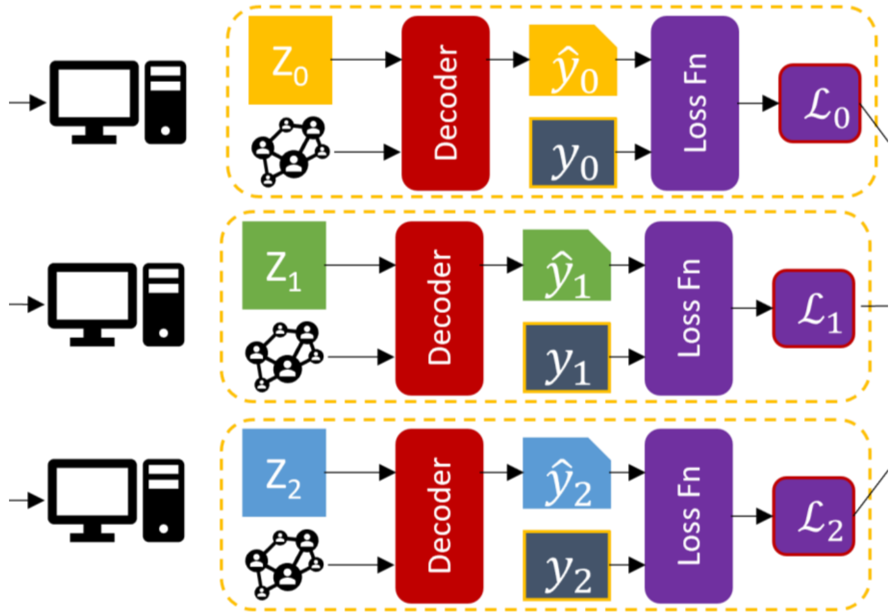
Algorithm 2: Recurrent Layer forward method

```
def forward(self, workers, partition):
    /* Leader tells each worker to begin
       executing                                   */
    futures = [];
    for w ∈ workers do
        future = asynchronously execute w.forward(partition);
        futuresappend(future);

    /* As workers return their embeddings, the
       leader processes them in order, as they
       arrive                                      */
    h=NULL;
    zs = [];
    for f ∈ futures do
        z, h = self.RNN(f.wait(), h);
        zs.append(z);

    return concat(zs)
```

4.) Scoring & Loss

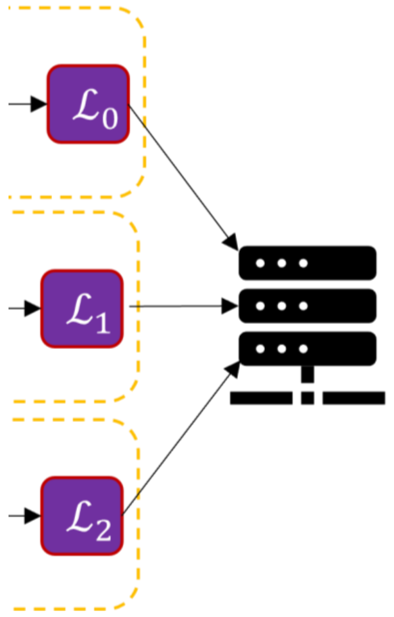
- Leader determines which slices of **Z** to send to which worker
  - During link detection, $\mathrm{g}(\mathbf{Z}_t) \approx \mathbf{A}_{\mathrm{t}}$
  - During link prediction, $\mathrm{g}(\mathbf{Z}_t) \approx \mathbf{A}_{t+n}$
  - Workers assume $\mathbf{Z}[\mathrm{i}]$ predicts the snapshot stored at index i, so leader must offset when necessary

- Workers decode all embeds
  - If training, workers return loss
  - Else, workers return scores, and labels (if available) for evaluation



5.) Backpropagation & Evaluation

- Leader awaits workers' output
- If training
  - Loss is calculated via DDP bucketing algorithm
  - First in RNN, then independently by each GNN
  - GNNs then average their grads via MPI collective communication
- If evaluating
  - Leader calculates AUC/AP if labels present
  - Leader merge sorts workers' scores and returns top-k anomalous edges



Loss Function

- Workers sample $|E_{t+n}|$ random negative edges
- New random sample each epoch
- Workers score negative and positive edges, P & N
- Loss is BCE of P and N

$$
\begin{aligned}
\mathcal{L}_t &= -\log(\Pr(\mathbf{A}_{t+n} \mid \mathbf{Z}_t)) \\
&\approx \frac{-1}{|P_{t+n}|} \sum_{p \in P_{t+n}} \log(p) + \frac{-1}{|N_{t+n}|} \sum_{n \in N_{t+1}} \log(1-n)
\end{aligned}
\tag{5}
$$

Benchmarking Methods

- (SI-)VGRNN
  - GCN on GRNN
  - GRNN output used as GCN input next snapshot
  - Currently #1 ranked Temporal LP model on PapersWithCode.com
- EGCN
  - RNN aims to find *parameters* of GCN
  - Very unique method, excellent at low info LP (guessing 10+ snapshots in the future)
- DynGraph2Vec (DynAE, DynRNN, DynAERNN)
  - MLP on RNN (no message passing or spectral convs)
  - Uses adj matrix as input & output vectors (not scalable)

Results

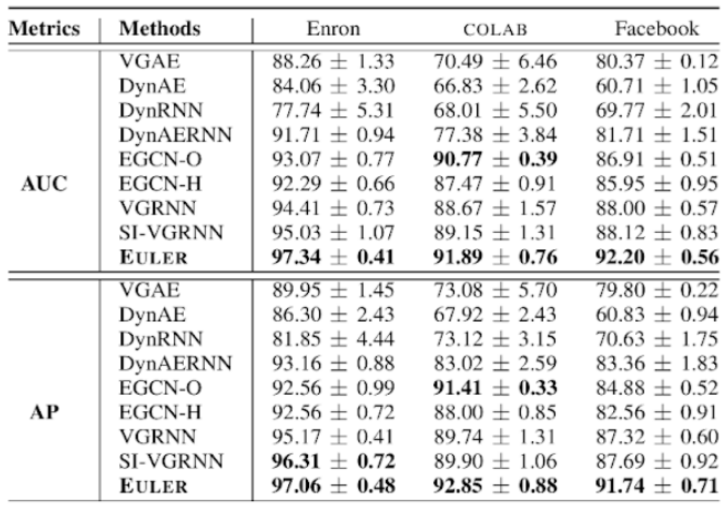


TABLE II: Comparison of EULER to related work on dynamic link detection

Metrics	Methods	Enron	COLAB	Facebook
AUC	VGAE	88.26 ± 1.33	70.49 ± 6.46	80.37 ± 0.12
	DynAE	84.06 ± 3.30	66.83 ± 2.62	60.71 ± 1.05
	DynRNN	77.74 ± 5.31	68.01 ± 5.50	69.77 ± 2.01
	DynAERNN	91.71 ± 0.94	77.38 ± 3.84	81.71 ± 1.51
	EGCN-O	93.07 ± 0.77	**90.77 ± 0.39**	86.91 ± 0.51
	EGCN-H	92.29 ± 0.66	87.47 ± 0.91	85.95 ± 0.95
	VGRNN	94.41 ± 0.73	88.67 ± 1.57	88.00 ± 0.57
	SI-VGRNN	95.03 ± 1.07	89.15 ± 1.31	88.12 ± 0.83
	EULER	**97.34 ± 0.41**	**91.89 ± 0.76**	**92.20 ± 0.56**
AP	VGAE	89.95 ± 1.45	73.08 ± 5.70	79.80 ± 0.22
	DynAE	86.30 ± 2.43	67.92 ± 2.43	60.83 ± 0.94
	DynRNN	81.85 ± 4.44	73.12 ± 3.15	70.63 ± 1.75
	DynAERNN	93.16 ± 0.88	83.02 ± 2.59	83.36 ± 1.83
	EGCN-O	92.56 ± 0.99	**91.41 ± 0.33**	84.88 ± 0.52
	EGCN-H	92.56 ± 0.72	88.00 ± 0.85	82.56 ± 0.91
	VGRNN	95.17 ± 0.41	89.74 ± 1.31	87.32 ± 0.60
	SI-VGRNN	**96.31 ± 0.72**	89.90 ± 1.06	87.69 ± 0.92
	EULER	**97.06 ± 0.48**	**92.85 ± 0.88**	**91.74 ± 0.71**

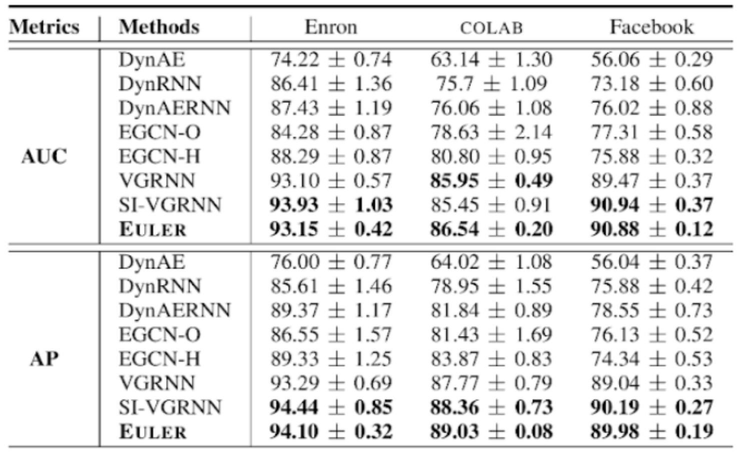


TABLE III: Comparison of EULER to related work on dynamic link prediction

Metrics	Methods	Enron	COLAB	Facebook
AUC	DynAE	74.22 ± 0.74	63.14 ± 1.30	56.06 ± 0.29
	DynRNN	86.41 ± 1.36	75.7 ± 1.09	73.18 ± 0.60
	DynAERNN	87.43 ± 1.19	76.06 ± 1.08	76.02 ± 0.88
	EGCN-O	84.28 ± 0.87	78.63 ± 2.14	77.31 ± 0.58
	EGCN-H	88.29 ± 0.87	80.80 ± 0.95	75.88 ± 0.32
	VGRNN	93.10 ± 0.57	**85.95 ± 0.49**	89.47 ± 0.37
	SI-VGRNN	**93.93 ± 1.03**	85.45 ± 0.91	**90.94 ± 0.37**
	EULER	**93.15 ± 0.42**	**86.54 ± 0.20**	**90.88 ± 0.12**
AP	DynAE	76.00 ± 0.77	64.02 ± 1.08	56.04 ± 0.37
	DynRNN	85.61 ± 1.46	78.95 ± 1.55	75.88 ± 0.42
	DynAERNN	89.37 ± 1.17	81.84 ± 0.89	78.55 ± 0.73
	EGCN-O	86.55 ± 1.57	81.43 ± 1.69	76.13 ± 0.52
	EGCN-H	89.33 ± 1.25	83.87 ± 0.83	74.34 ± 0.53
	VGRNN	93.29 ± 0.69	87.77 ± 0.79	89.04 ± 0.33
	SI-VGRNN	**94.44 ± 0.85**	**88.36 ± 0.73**	**90.19 ± 0.27**
	EULER	**94.10 ± 0.32**	**89.03 ± 0.08**	**89.98 ± 0.19**

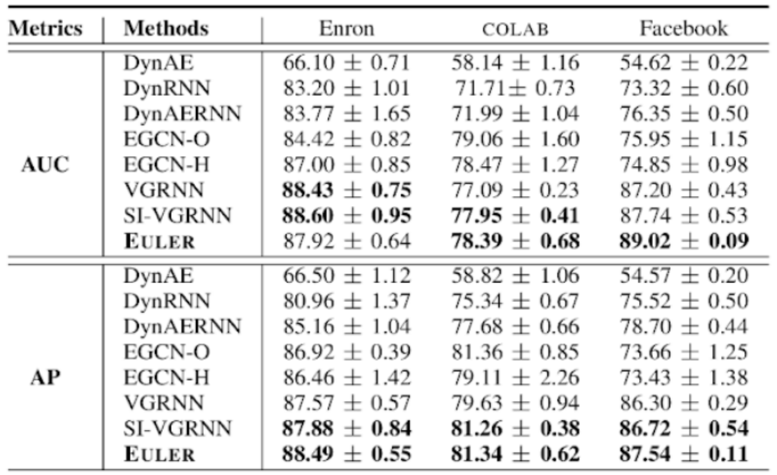


TABLE IV: Comparison of EULER to related work on dynamic new link prediction

Metrics	Methods	Enron	COLAB	Facebook
AUC	DynAE	66.10 ± 0.71	58.14 ± 1.16	54.62 ± 0.22
	DynRNN	83.20 ± 1.01	71.71± 0.73	73.32 ± 0.60
	DynAERNN	83.77 ± 1.65	71.99 ± 1.04	76.35 ± 0.50
	EGCN-O	84.42 ± 0.82	79.06 ± 1.60	75.95 ± 1.15
	EGCN-H	87.00 ± 0.85	78.47 ± 1.27	74.85 ± 0.98
	VGRNN	**88.43 ± 0.75**	77.09 ± 0.23	87.20 ± 0.43
	SI-VGRNN	**88.60 ± 0.95**	**77.95 ± 0.41**	87.74 ± 0.53
	EULER	87.92 ± 0.64	**78.39 ± 0.68**	**89.02 ± 0.09**
AP	DynAE	66.50 ± 1.12	58.82 ± 1.06	54.57 ± 0.20
	DynRNN	80.96 ± 1.37	75.34 ± 0.67	75.52 ± 0.50
	DynAERNN	85.16 ± 1.04	77.68 ± 0.66	78.70 ± 0.44
	EGCN-O	86.92 ± 0.39	81.36 ± 0.85	73.66 ± 1.25
	EGCN-H	86.46 ± 1.42	79.11 ± 2.26	73.43 ± 1.38
	VGRNN	87.57 ± 0.57	79.63 ± 0.94	86.30 ± 0.29
	SI-VGRNN	**87.88 ± 0.84**	**81.26 ± 0.38**	**86.72 ± 0.54**
	EULER	**88.49 ± 0.55**	**81.34 ± 0.62**	**87.54 ± 0.11**

- EULER out-performs prior work on all detection tests
  - Though only with *statistical significance* on FB and Enron AUC
- Prior works are not statistically significantly better than EULER on any prediction tests
- EULER is better with significance on new FB test, and equivalent elsewhere

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Discussion

- The benefits of using RNN output as GNN input is minimal
- The RNN and GNN *components* contribute to these models working

The LANL Dataset

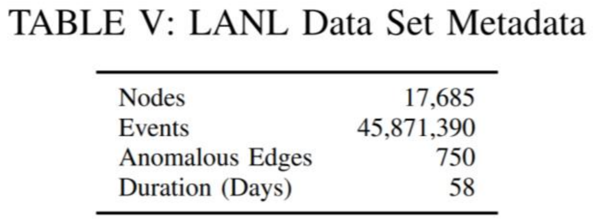


TABLE V: LANL Data Set Metadata

Nodes	17,685
Events	45,871,390
Anomalous Edges	750
Duration (Days)	58

- 58 Days of log files in a real-world system
- Attack campaigns sporadically
- Redlog identifies 750 authorization events “involved in compromise”
- Nodes: Users, Computers, System
- Edges: Authorizations, weighted according to frequency:

$$W((u,v) \in \mathcal{E}) = \sigma\left(\frac{C(u,v) - \mu_{\mathcal{E}}}{\Sigma_{\mathcal{E}}}\right)$$

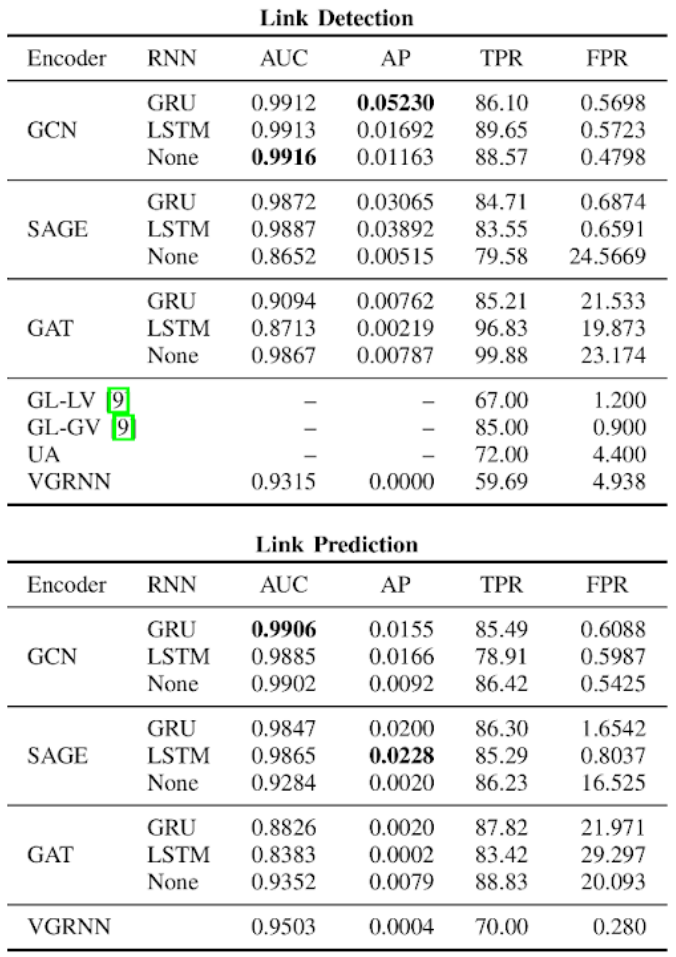
- Features: 1-hot ID, and 1-hot vector of node’s role
- Threshold for classification as anomaly: $\underset{\tau}{\mathrm{argmin}} \quad \|(1-\lambda)\mathrm{TPR}(\tau) - \lambda \mathrm{FPR}(\tau)\|$

Results

- Link Detection:
  - Best precision was GCN-GRU
  - Surprisingly, ablation study had best AUC (with GRU). RNN may not be necessary
  - SAGE also performed well

- Link Prediction
  - SAGE had best precision this time
  - AUC not as good as GCN

- Overall
  - Regression metrics are better than all prior works
  - Higher TPR and lower FPR on classification metrics than prior works



Link Detection

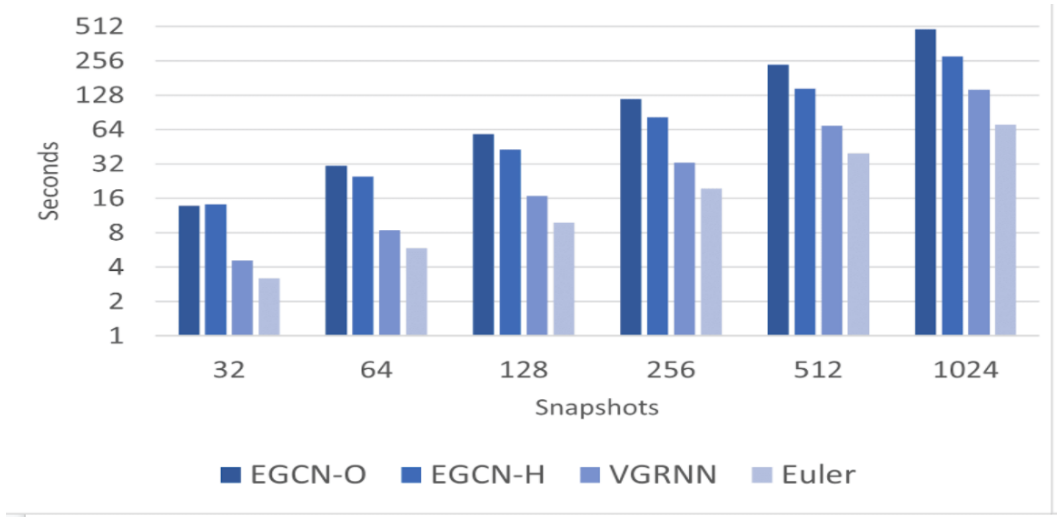
Encoder	RNN	AUC	AP	TPR	FPR
GCN	GRU	0.9912	**0.05230**	86.10	0.5698
	LSTM	0.9913	0.01692	89.65	0.5723
	None	**0.9916**	0.01163	88.57	0.4798
SAGE	GRU	0.9872	0.03065	84.71	0.6874
	LSTM	0.9887	0.03892	83.55	0.6591
	None	0.8652	0.00515	79.58	24.5669
GAT	GRU	0.9094	0.00762	85.21	21.533
	LSTM	0.8713	0.00219	96.83	19.873
	None	0.9867	0.00787	99.88	23.174
GL-LV [9]		–	–	67.00	1.200
GL-GV [9]		–	–	85.00	0.900
UA		–	–	72.00	4.400
VGRNN		0.9315	0.0000	59.69	4.938

Link Prediction

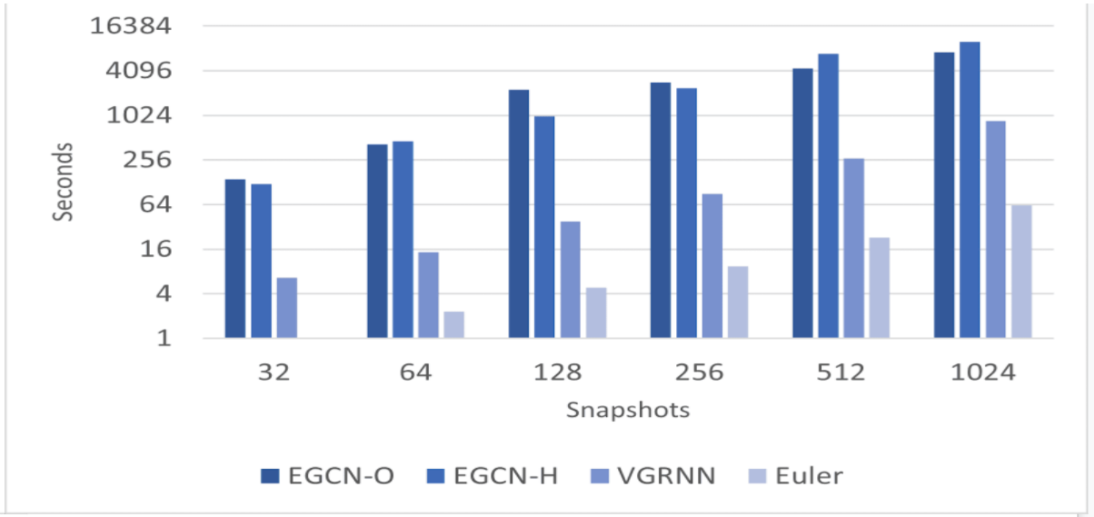
Encoder	RNN	AUC	AP	TPR	FPR
GCN	GRU	**0.9906**	0.0155	85.49	0.6088
	LSTM	0.9885	0.0166	78.91	0.5987
	None	0.9902	0.0092	86.42	0.5425
SAGE	GRU	0.9847	0.0200	86.30	1.6542
	LSTM	0.9865	**0.0228**	85.29	0.8037
	None	0.9284	0.0020	86.23	16.525
GAT	GRU	0.8826	0.0020	87.82	21.971
	LSTM	0.8383	0.0002	83.42	29.297
	None	0.9352	0.0079	88.83	20.093
VGRNN		0.9503	0.0004	70.00	0.280

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Performance Comparison



Forward Time



Backward Time

Euler uses 16 workers; prior works use 16 inter-op threads

•Euler is consistently faster than prior works
•Forward time is about 2x faster
•Backward time is 16x better (showing near-perfect scaling)

Conclusion

Euler accomplished the following:

- Consistently as powerful or better than prior work
- Parallelized temporal link prediction
- First use of graph temporal link prediction for IDS
- Achieved best scores on LANL, but could be improved

Future work

- Uses other than pure anomaly detection for Euler embeddings
- Test other decoding functions
- Distributed GNN in general, find other places to optimize

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Questions?

THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

Thank you

THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC