Evaluating SpikeNet on the Temporal Graph Benchmark

Tej Shah
tej.shah@rutgers.edu
B.S. Computer Science
B.S. Business Analytics

Advith Chegu
ac1771@rutgers.edu
M.S. Computer Science

Priyanshu Srivastava
priyanshu.s@rutgers.edu
M.S. Computer Science

Abstract

We evaluated SpikeNet, a new biologically inspired temporal graph representation learning technique, on the recent Temporal Graph Benchmark (TGB). For the Node Classification task on `tgbn-trade`, SpikeNet is the worst performing technique, but is similarly competitive to other technical temporal graph representation learning techniques, suggesting model performance may be closely related to the dataset size regime: for small data regimes, the simplest model is preferred. For the Node Classification task on `tgbn-genre`, SpikeNet is the worst-performing technique among all the techniques. All code is found on GitHub, linked here: https://github.com/tejpshah/tgb-spikenet.

Introduction

Graphs are powerful tools to represent complex relationships between entities. However, most existing work is focused on and is designed for static graphs. This is a significant limitation, since, most real-world graphs are dynamic, and change over time. For example, in a social network, people follow and unfollow other people. Or, in the case of finance, transactions continuously happen between buyers and sellers.

Practically, it is useful to model this spatio-temporal behavior on graphs for node classification and link prediction. Temporal graph representation learning can be used in downstream applications to predict friendships in a social network (link prediction) or detect anomalies in a financial transaction network (node classification). For example, link prediction is useful in social , citation, and biological networks (Zhang & Chen, 2018). Temporal graphs model are changing networks throughout time with timestamped edges. The graph G_t at time t includes vertices V_t, edges E_t, and optionally node features X_t and edge features M_t. A fixed chronological split is used for training, validation, and test datasets.

At NeurIPS, the temporal graph representation learning workshop has only been around for two years! Temporal graph representation learning as a subfield is growing more popular as researchers extend static models with temporal dimensions and update schemes. Techniques like TGN (Rossi et al., 2020) and JODIE (Kumar, Zhang, & Leskovec, 2019) use RNN architectures to update node states to capture graph dynamics despite the high computational overhead; moreover, methods like EvolveGCN (Pareja et al., 2020) use RNNs to regulate parameters at each timestep, while other approaches model graph events as stochastic processes.

Computational cost is a significant issue with temporal graphs. One class of energy-efficient neural networks is Spiking Neural Networks (SNNs). SNNs are brain-inspired, deliver discrete, asynchronous information through spike trains, and are inherently suited for temporal data. Since prior work in temporal graphs focus on non-temporal techniques, SNNs are a promising approach to feasibly benefit from the temporal structure found in dynamic graphs(Debanne & Inglebert, 2023). Our initial plan for this project was to investigate SpikeNet on Node Classification and Link Prediction tasks.

Background

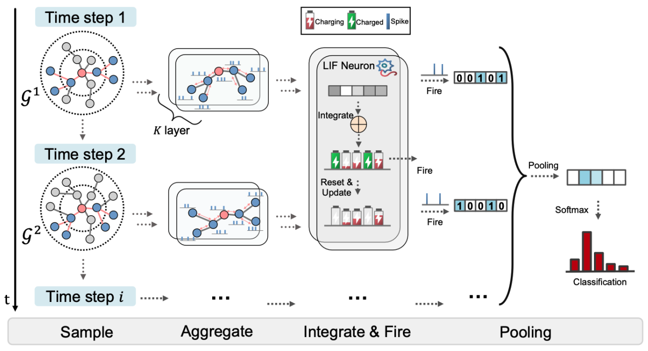


Figure 1: SpikeNet Architecture (Li et al., 2023)

Recently, Li et al. (Li et al., 2023) proposed SpikeNet, a brain-inspired approach to capture the spatio-temporal dynamics of graphs in a computationally efficient manner. At each time step, SpikeNet aggregates neighborhood information by sampling at macro and micro scales. After aggregation, these signals are passed to a leaky-integrate-and-fire (LIF) neuron, where spikes are emitted when the voltage exceeds the threshold. SpikeNet has an adaptive voltage threshold. Afterward, SpikeNet uses a pooling operation to compute final representations for each node, which can be used for downstream tasks (i.e. node classification, and link prediction). Empirically, SpikeNet outperforms relevant techniques for node classification on temporal graphs.

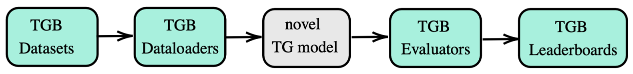


Figure 2: TGB Pipeline (Huang et al., 2023)

Huang et al. (Huang et al., 2023) introduced the Temporal Graph Benchmark (TGB) at NeurIPS 2023. TGB contains large-scale, diverse datasets for temporal graph representation learning, spanning both link prediction and node property prediction. Key to TGB is a pipeline with data loaders to plug-and-play different models, on a consistent set of evaluations (i.e. ranking-based metrics, multiple negative samples for link prediction). Extensive empirical experiments show variance in model performance across different datasets, and that, simple heuristics oftentimes outperform state-of-the-art models. Therefore, TGB provides a leaderboard to help researchers understand temporal graph modeling techniques and their performance tradeoffs on different datasets.

Experimental Design

The evaluation metric for the Node Prediction tasks is Normalized Discounted Cumulative Gain (NDCG), which takes into account the relative preference ordering for a node label.

`tgbn-trade`: This dataset is used for Node Prediction. Each node in this dataset represents a nation in the United Nations agriculture trading network from 1986 to 2016. Each edge represents the sum trade value from one nation to another. The task is to predict the proportion one country trades with every other country every year. In this dataset, there are 255 nodes, $468,245$ edges, and 32 time-steps.

`tgbn-genre`: This is another graph dataset used for Node Prediction. Each node represents a user or genre in a bipartite graph, and edges represent interactions between users and the genre of music they listen to. The task is to predict the proportion of music of the different genres that users listen to. It is important to note that this goes against the conventional task of node classification as there isn't a single label that we are trying to predict but rather a vector of labels for each node which is a much more difficult task. In this dataset, there are 1,505 nodes, 17,858,395 edges, and 133,758 timesteps.

TGBN Trade Results

We used Table 1 to train 540 unique SpikeNet models on the `tgbn-trade` dataset for a train-test-split of $70-15-15$, with a LIF Neuron, and 100 epochs each for training. Across all these combinations, we got the following distributions for the NDCG evaluation metric. Across the 540 models trained, the median NCDG performance on the test set was approximately 0.325 per Figure 3. We ran the best hyperparameter setting in grid search for $k = 5$ times, and then averaged the results, to get performance metrics for the leaderboard shown in Table 2. SpikeNet is the worst performing technique.

This performance appears to be a limitation of the `tgbn-trade` dataset since it has only 255 nodes and only 32 timesteps. It's possible that in this smaller data regime, more technical methods like SpikeNet do not receive enough spatio-temporal signals to learn interesting patterns. This trend seems to hold with similarly poor performance from other methods such as DyGFormer, TGN, and DyRep.

Hyperparameter	Values
Alpha	0.6, 0.8, 1.0
Batch Size	512, 1024
Dropout	0.1, 0.2, 0.3
Hidden Units	[128, 10], [256, 20], [128, 128], [10, 10], [256, 256]
p	0.6, 0.8, 1.0
Concat Flag	True, False

Table 1: `tgbn-trade` Grid Search Hyperparameters

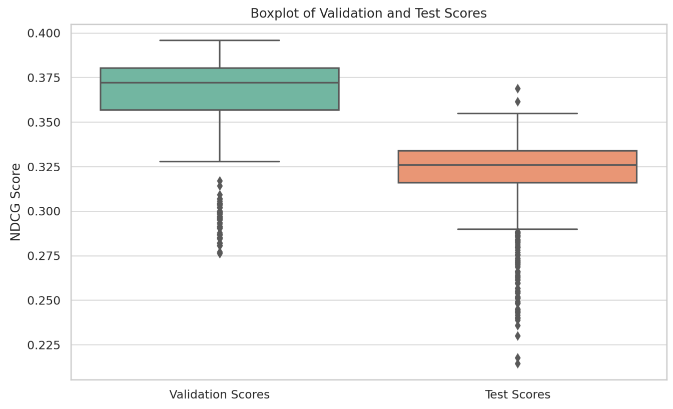


Figure 3: `tgbn-trade` Grid Search Performance

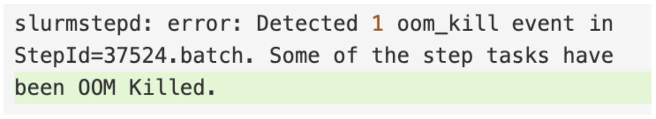
Method	Test NDCG@10	Val NDCG@10
Persistent Forecast	0.855	0.860
Moving Average	0.823	0.841
DyGFormer	0.388	0.408
TGN	0.374	0.395
DyRep	0.374	0.394
SpikeNet	0.334	0.380

Table 2: `tgbn-trade` NDCG@10 Leaderboard

TGBN Genre Results

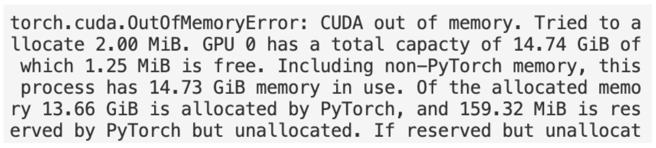
We were severely limited in this task due to the lack of computational resources we had at our disposal through SLURM. Due to a combination of poorly handled memory management in the SpikeNet architecture and the size of the `tgbn-genre` dataset, we were only able to load 1% of the total dataset into memory. You can see an example of the out of memory error faced by the SLURM job when trying to handle more than 1% of the dataset in Figure 4. Additionally, we had to reduce our batch size from a default size of 1024 edges to 16 since we faced memory issues on the GPUs as well. You can see an example of the error in Figure 5.

Our training process consisted of 57 epochs run over 3 days resulting in minimal learning by the model. This is mainly due to the fact that the size of the vector we need to predict is much larger than the vector for `tgbn-trade` which made model prediction more difficult. Due to the size of the dataset and memory issues we ran into, we were unable to perform a grid search or any type of hyper parameter tuning. Final results can be seen in Table 3. Though disappointing, we believe that with a rewritten dataloader for SpikeNet, this task would be more feasible and produce better results.



```
slurmstepd: error: Detected 1 oom_kill event in
StepId=37524.batch. Some of the step tasks have
been OOM Killed.
```

Figure 4: Memory Error



```
torch.cuda.OutOfMemoryError: CUDA out of memory. Tried to a
llocate 2.00 MiB. GPU 0 has a total capacity of 14.74 GiB of
 which 1.25 MiB is free. Including non-PyTorch memory, this
 process has 14.73 GiB memory in use. Of the allocated memo
ry 13.66 GiB is allocated by PyTorch, and 159.32 MiB is res
erved by PyTorch but unallocated. If reserved but unallocat
```

Figure 5: GPU Memory Error

Method	**Test NDCG@10**	**Val NDCG@10**
Persistent Forecast	0.509	0.499
Moving Average	0.367	0.403
DyGFormer	0.365	0.371
TGN	0.357	0.350
DyRep	0.351	0.357
SpikeNet	< 0.01	< 0.01

Table 3: `tgbn-genre` NDCG@10 Leaderboard

Discussion

At the start of the semester, we initially proposed to evaluate SpikeNet on TGBN for 1 link prediction task (`tgbl-review`) and 1 node prediction task (`tgbn-genre`). After further investigation, we realized that SpikeNet only worked with node classification in the original paper, so adding capabilities for link prediction would be non-trivial, and also, not faithful to the original implementation.

Hence, we worked with `tgbn-trade`, a node classification task instead of `tgbl-review`. We were able to fully run model training and evaluation on `tgbn-trade` since it was sufficiently small to fit into memory. The `tgbn-genre` dataset, the next largest dataset, was too large to fit into memory, so even with maximum compute resources from SLURM, we were only able to work with 1% of the data.

Furthermore, working with TGBN is not as easy as advertised; it was not really a "plug-and-play" approach, and it does take non-trivial time to configure your specific models to integrate with the TGBN pipeline. That being said, TGBN does have some additional tools (i.e. more efficient data batching) that we didn't have a chance to try due to time constraints. Possibly, those tools may be able to remedy our data loading issues in `tgbn-genre`.

Though we were unable to get SpikeNet in its original form factor working with Link Prediction datasets in TGB, we argue that evaluating SpikeNet on link prediction datasets is promising. In a link prediction task, links change much more than the nodes themselves; therefore, having good spatio-temporal representations is necessary. If we can show SpikeNet to have good spatio-temporal representations, we'd correspondingly have good link prediction performance. To implement this, we would still compute node embeddings for each node as we did for node classification. Then, we'd use the dot-product between node embeddings, and an adaptive threshold, to determine if a link exists between two nodes. In the future, we plan to write modules enabling SpikeNet to be compatible with the Link Prediction task.

Conclusion

On `tgbn-trade`, SpikeNet is the worst-performing method, but is similarly competitive to other non-biologically inspired temporal graph representation learning methods. This suggests that we need to evaluate SpikeNet on a larger data regime to see where this technique is more performant versus simpler techniques (i.e. moving average, persistent forecast).

On `tgbn-genre`, SpikeNet is the worst-performing method by far, compared to all other non-biologically inspired methods. This result may be because we were only able to work with a 1% subset of the dataset, given our maximum computational resources from SLURM.

Overall, on the two datasets we evaluated, SpikeNet is the worst-performing compared to standard temporal graph methods for node classification. These results suggest that, SpikeNet, at best, is no better than existing non-biologically inspired approaches for learning on dynamic graphs.

References

Debanne, D., & Inglebert, Y. (2023). Spike timing-dependent plasticity and memory. *Current Opinion in Neurobiology*, *80*, 102707.

Huang, S., Poursafaei, F., Danovitch, J., Fey, M., Hu, W., Rossi, E., ... Rabbany, R. (2023). Temporal graph benchmark for machine learning on temporal graphs. *Advances in Neural Information Processing Systems*.

Kumar, S., Zhang, X., & Leskovec, J. (2019, July). Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th acm sigkdd international conference on knowledge discovery amp; data mining.* ACM. Retrieved from `http://dx.doi.org/10.1145/3292500.3330895` doi: 10.1145/3292500.3330895

Li, J., Yu, Z., Zhu, Z., Chen, L., Yu, Q., Zheng, Z., ... Meng, C. (2023). Scaling up dynamic graph representation learning via spiking neural networks. In *AAAI* (pp. 8588–8596). AAAI Press.

Pareja, A., Domeniconi, G., Chen, J., Ma, T., Suzumura, T., Kanezashi, H., ... Leiserson, C. (2020, Apr.). Evolvegcn: Evolving graph convolutional networks for dynamic graphs. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(04), 5363-5370. Retrieved from `https://ojs.aaai.org/index.php/AAAI/article/view/5984` doi: 10.1609/aaai.v34i04.5984

Rossi, E., Chamberlain, B., Frasca, F., Eynard, D., Monti, F., & Bronstein, M. (2020). *Temporal graph networks for deep learning on dynamic graphs.*

Xu, D., Ruan, C., Korpeoglu, E., Kumar, S., & Achan, K. (2020). Inductive representation learning on temporal graphs. In *International conference on learning representations (iclr).*

Zhang, M., & Chen, Y. (2018). Link prediction based on graph neural networks. *Advances in neural information processing systems*, *31*.