

<https://doi.org/10.1038/s44335-026-00068-2>

Citation network datasets for benchmarking spiking graph neural networks on experimental neuromorphic hardware



Kevin Zhu^{1,4} ✉, Ian Mulet^{2,4} ✉, Prasanna Date³, Ashish Gautam³, Tyler Nitzsche², Shay Snyder¹, Shruti Kulkarni³, Seung-Hwan Lim³, Guojing Cong³, Cameron Nowzari¹, Catherine Schuman² ✉, Maryam Parsa¹ ✉ & Thomas Potok³

Spiking neural networks (SNNs) running on neuromorphic computers offer an energy-efficient alternative for AI tasks. Recently, spiking graph neural networks (S-GNNs) have been shown to produce encouraging results on benchmark citation network datasets such as Cora, CiteSeer, and PubMed for node classification tasks. These S-GNNs were run on SNN simulators only because they contain up to tens of thousands of neurons and up to millions of synapses, translating poorly to neuromorphic hardware. Therefore, in this paper, we create a suite of benchmark datasets from the CiteSeer dataset that can be accommodated on current neuromorphic hardware platforms. Our contribution consists of a collection of three datasets. First, we have an induced subgraph of CiteSeer, which we call MiniSeer, containing 2110 papers, 3604 binary features, and 6 topics. Second, MicroSeer is a very small dataset consisting of 84 papers, 1227 features, and 6 topics. Lastly, BiteSeer is a collection of 15 binary classification datasets. We present creation of these datasets along with accuracies, running times, and spike counts when simulated. We believe that our results in this paper will be used by the neuromorphic community to benchmark, test, and develop neuromorphic hardware and simulators.

With the rise of the success of AI, there has been a corresponding increase in power consumption required for training and inference. Neuromorphic computers offer one path forward as a hardware paradigm for low-power AI computations. The vast majority of research in porting AI computation to neuromorphic systems has been focused on convolutional neural networks for image processing^{1,2}; in more recent efforts, there has been some attempts to map attention models onto neuromorphic systems^{3,4} and to extend Graph Convolutional Networks (GCNs) with spiking elements^{5–7}. In contrast, there has been relatively little work in mapping graph neural networks directly onto neuromorphic systems, even though such neuromorphic systems have been shown to perform graph processing such as shortest path calculations very efficiently^{8–10}. Moreover, it has been shown that there are approaches for mapping graph neural network-like problems such as node classification onto neuromorphic systems that achieve comparable results to

conventional graph neural network approaches. As such, graph neural networks are a natural fit for evaluation and mapping on neuromorphic hardware systems.

In the field of graph neural networks, there are a variety of benchmarks that are widely used in the field and are typically used for initial evaluation of graph neural network approaches (similar to how MNIST is used for initial evaluation of convolutional neural network approaches). Datasets for node classification include three citation networks, Cora, CiteSeer, and PubMed, each of which has thousands of nodes. Large-scale neuromorphic systems, which have primarily been fully digital, fully CMOS implementations, such as Loihi¹¹ and SpiNNaker¹², can reach hundreds of thousands to millions of neurons in their implementations; these types of implementations can easily evaluate benchmarks at this scale. However, there is a tremendous amount of research in mixed analog digital neuromorphic implementations, as well

¹Department of ECE, George Mason University, Fairfax, VA, USA. ²Department of EECS, University of Tennessee, Knoxville, TN, USA. ³Oak Ridge National Laboratory, Oak Ridge, TN, USA. ⁴These authors contributed equally: Kevin Zhu, Ian Mulet. ✉e-mail: kzhu4@gmu.edu; imulet@vols.utk.edu; cschuman@utk.edu; mparsa@gmu.edu

as implementations leveraging emerging devices, including memristors or resistive RAM, ferroelectric devices, superconducting devices, and biomimetic devices^{13–20}. Physical realizations of neuromorphic implementations of these types are typically very small (i.e., on the order of 100s of neurons). For example, ROLLS, a mixed analog-digital implementation, has the ability to represent 128K analog synapses on-chip, but only 256 neurons²¹. For these emerging hardware implementations, even the MNIST-scale tasks of Cora, CiteSeer, and PubMed are too large to evaluate on chip.

In order to push forward research in realizing graph neural network algorithms on emerging neuromorphic hardware, it is critical that benchmarks be developed that can be evaluated on these smaller systems. Similar to approaches which use smaller versions of MNIST, such as digits, smaller-scale graph neural network benchmarking datasets can be developed from existing datasets.

In this work, we introduce a suite of datasets derived from the CiteSeer dataset²² with varying sizes so that their SNN representations can fit on various neuromorphic hardware. We explain how our datasets—MiniSeer, MicroSeer, and the BiteSeer collection—were generated from CiteSeer, and we provide metadata for these datasets. Additionally, we provide initial results using a spiking graph neural network implementation on these datasets, evaluated in the SuperNeuroMAT²³ simulator. We present the performance metrics such as validation and testing accuracies, running times, and spike counts for each of these datasets in the Results Section, providing an initial baseline of neuromorphic results for these benchmarks. In creating and releasing these datasets and the initial baseline results, we hope that these datasets will advance hardware-software co-design of spiking graph neural network-based AI models on emerging neuromorphic hardware platforms.

In graph learning, the task of node classification finds applications in various domains such as social network analysis^{24,25}, recommender systems²⁶, materials sciences^{27,28}, and fraud detection²⁹. The citation network datasets generated in this paper are examples of node classification problems as well³⁰. The ability to exploit the underlying graph structure allows link-based classification methods to capture complex dependencies and structural information that purely feature-based algorithms might miss. Modern methods for link-based classification leverage GNNs, or perhaps most popularly, Graph Convolutional Networks (GCNs) as used by Kipf & Welling³¹.

On the other hand, neuromorphic approaches have benefits such as energy efficiency and data efficiency³². In the literature, there have been some neuromorphic approaches developed for graph learning problems. Yin et al. present a framework called dynamic spiking graph neural networks, which propagates information in the first layer to the last layer to

avoid loss of information and also applies implicit differentiation to dynamic graph neural networks⁷. Yin et al. present a framework called continuous spiking graph neural networks, which requires significantly less computational resources as compared to traditional non-spiking continuous GNNs⁶. Xing et al. propose SpikeLLM, which is a spiking version of an LLM containing millions of neurons and billions of synapses³³. However, in all these approaches, only a part of the entire network was ported to the spiking domain.

Cong et al. previously devised an alternative spiking approach for GNNs^{34,35}. This approach leverages the spiking behavior not just in a particular layer of the network, but throughout the entire network. We refer to this as the neuromorphic graph learning approach, in which we create a spiking graph neural network, or S-GNN. The S-GNN is constructed by leveraging the topology of the underlying citation graph. Therefore, the S-GNN can be constructed in $\mathcal{O}(|V| + |E|)$ time, where V is the set of nodes and E is the set of edges in the citation network. In addition, a neuron is created for each topic and feature (if used) in the network. This method of creating a matching graph structure inside the spiking neural network (SNN) has the benefit of reducing training time. It requires some papers to have known labels in order to infer the label of the other papers. We show a visual example of this translation from citation network to S-GNN in Fig. 1. Using this method, Cong et al. obtained validation and testing accuracies on the Cora, CiteSeer, and PubMed datasets that were on par with the state-of-the-art accuracies using traditional GNN-based methods. However, these results were obtained in simulation and not on neuromorphic hardware. This is because the Cora, CiteSeer, and PubMed S-GNNs have up to tens of thousands of neurons and up to millions of synapses, especially if this neuromorphic graph learning approach is extended to use node feature information using additional neurons. S-GNN configurations of this scale cannot be accommodated on readily available neuromorphic hardware. Although some proprietary neuromorphic hardware or cloud platforms are able to accommodate S-GNNs of this scale, the physical neuromorphic hardware devices are often not open, easily obtainable, or otherwise readily accessible to the neuromorphic community at large.

Therefore, in this paper, we focus on creating a suite of small datasets from the CiteSeer dataset called MiniSeer, MicroSeer, and BiteSeer. These new datasets can be used to build, train, and test S-GNN models that are more amenable for usage with the current neuromorphic hardware platforms, particularly hardware-under-development. We expect that the neuromorphic community will use these datasets to benchmark their neuromorphic graph learning approaches running in either simulation or on digital, analog, or mixed-signal neuromorphic hardware.

Fig. 1 | Example of application of S-GNN approach to Citation Network. **A** Diagram of example citation network, and **(B)** its translation to S-GNN. Each paper in the citation network is represented as a node in graph **(A)** or a neuron in network **(B)**. Paper citations are represented as directed edges in the graph, for example Paper 6 is cited in Paper 7. However, the current Neuromorphic Graph Learning approach represents this as a bidirectional connection using a pair of synapses. In this example, Paper 6 is part of the test set, so it is connected to each of the three neurons representing topics/labels. We represent the true label of each node in graph **(A)** using colors. **NB:** In this example, Paper 6 is part of the test set so its label is meant to be inferred, but the datasets contain ground-truth labels for all papers.

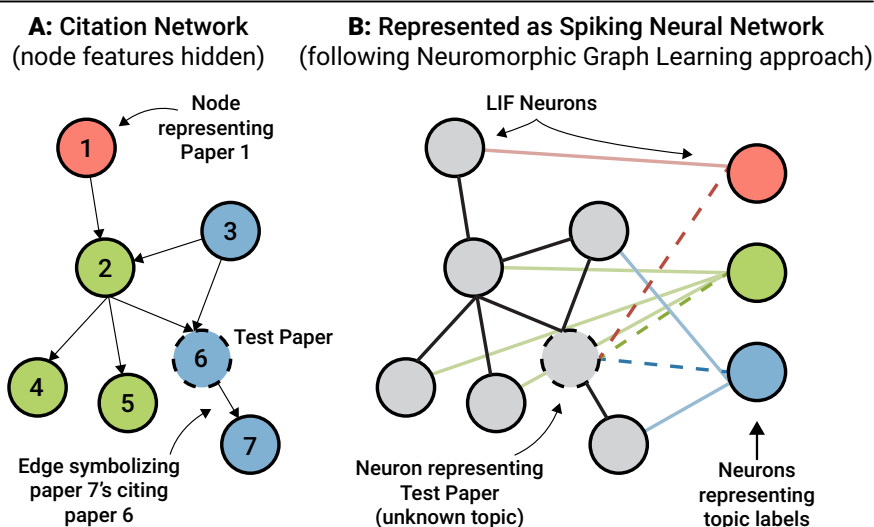


Table 1 | Accuracies for Validation and Testing Sets

	SuperNeuroMAT		SpikingGCN		
	Validation	Test	Training	Validation	Test
MicroSeer	83.33%	58.33%	97.50%	61.20%	61.00%
MiniSeer	55.00%	58.34%	96.50%	72.40%	71.50%
CiteSeer	42.50%	42.40%	96.67%	71.20%	70.90%

Results

Experimental setup

For all SNN experiments, we use SuperNeuro, a high-performance neuro-morphic simulator that supports both event-driven and matrix-based computation²³. Implementing an S-GNN requires explicit support for bidirectional synapses and dense graph connectivity. While libraries such as *snnTorch*³⁶ can, in principle, be extended to support such structures, they are primarily optimized for layer-based and feedforward SNN architectures, making dense graph connectivity and online STDP cumbersome and inefficient to implement at scale. Earlier work on spiking GNNs leveraged NEST but encountered significant computational overhead when running the large networks required by neuromorphic graph learning^{34,35,37}. Other SNN simulators are available, such as Nengo³⁸ and Brian³⁹, but previous work highlights that NEST provides the lowest computational overhead with larger graph networks⁴⁰. SuperNeuroMAT, the matrix computation backend of SuperNeuro, provides an efficient way to implement neuromorphic graph learning because all synaptic and membrane updates are performed as batched tensor operations. This results in predictable scaling behavior, reduced simulation time, and stable memory usage across different network sizes²³.

We compare our results against SpikingGCN, which is a spiking graph convolutional network approach developed by Zhu et al.³². In SpikingGCN, they use the convolutional aspects of graphs translated onto spike trains. All of the experiments presented were conducted on a workstation with an AMD Ryzen 7 7800X3D CPU, and the validation, training, and test were seeded with the same value.

Table 1 shows the relative performance of each dataset on the validation and testing sets for both our SuperNeuroMAT S-GNN implementation and the SpikingGCN implementation. Our algorithmic setup for the neuromorphic graph neural network is from ref. 34, which was then ported to the SuperNeuroMAT SNN simulator. This algorithm does not make use of features from the citation graphs and has no training phase, only a network construction phase. Instead of typical decoding strategies utilized in SNNs, we analyze the weights from the synapses connecting the test paper and each of the topics. The lowest weight of all the synapses is the prediction of the topic. We utilized the STDP implementation in SuperNeuroMat, with $\alpha_+ = [1.0, 0.5, 0.1, 0.01, 0.001]$ and a fixed α_- of [0.001] across 5 timesteps. Unlike in ref. 35, we present results here with only hand-tuning of hyperparameters; no other hyperparameter optimization techniques were used. All of the results use the same configuration for SuperNeuro, which can be found in Table 2.

As in ref. 34, once the network is constructed, each paper in the validation set is evaluated by sending a spike to the neuron associated with that paper. During simulation, spike timing dependent plasticity (STDP) causes weight updates on synapses within the network. After the simulation elapses, the topic with the strongest connection to that paper is the predicted class. We ensure the weights and initial network state are the same before each evaluation. To measure accuracy, we simply use the percent of correct class predictions out of the total number of papers, $TP/N \times 100$.

Table 1 shows that the neuromorphic graph learning approach, when implemented in SuperNeuroMAT, achieves higher accuracy on our new, smaller datasets than when using the original CiteSeer dataset. For MiniSeer, we expect to see a performance increase from removing the smaller connected components, as neuromorphic graph learning does well with a single large connected component. There is a 15.84%

Table 2 | Default configuration for SuperNeuro

Parameter	Value
Paper leak	0.0
Paper threshold	1.0
Training Refractory	1.0
Validation Refractory	1.0
Test Refractory	1000
Topic Leak	0.0
Topic Threshold	1.0
Graph Weight	100
Graph Delay	1.0
Train to Topic Weight	1.0
Train to Topic Delay	1.0
Validation to Topic Weight	0.001
Validation to Topic Delay	1.0
Test to Topic Weight	0.001
Test to Topic Delay	1.0
Simulation Time	10
α Positive	1.0, 0.5, 0.1, 0.01, 0.001
α Negative	0.0001
STDP Time-steps	5

increase in accuracy in the testing set, over the original CiteSeer dataset. MicroSeer also has an overall increase in both validation and testing accuracy, and we attribute this to the same reasoning that MiniSeer saw a boost in accuracy: a single “large” component. Although, there is a much higher accuracy for the validation versus the testing sets for MicroSeer. Simply, the way the train/validation/test split occurs has an effect on the validation and test accuracies. This discrepancy is due to the size of the MicroSeer dataset, and was to be expected, as the validation set is much smaller than the test set size. If the papers that were selected in the validation set perform usually well, there is a much higher than expected accuracy boost, likewise if papers are selected are inferred worse than expected, we have noted an increase in the test accuracy. We assume then there is a relationship between validation and test accuracies, and with the size of this dataset just a small number of papers is enough to increase either validation or test accuracies. Despite the size of MicroSeer, we still see improved performance over CiteSeer, showcasing how a fully connected structure improves performance, even at small scale.

SpikingGCN’s results were obtained in much of the same way as the SuperNeuroMAT results. This means that the results from SpikingGCN were without features or Bayesian Optimization. The hyperparameters were simply hand-tuned, and the datasets were utilized as-is.

With SpikingGCN, we can see a similar trend of MiniSeer performing better than CiteSeer in testing accuracies, but that trend does not continue with MicroSeer. SpikingGCN utilizes a more traditional approach regarding neuromorphic implementation, using the features of the graph to create a spike train for input into the network. This methodology difference is the reason why we see varied accuracies between the SuperNeuroMAT implementation and SpikingGCN. It is important to note that this is not a comparison of accuracies between the two implementations, but rather an illustration of the datasets being used in a working environment.

Table 3 demonstrates the results for each topic comparison in BiteSeer. We see high accuracies for both validation and testing sets, mostly from reducing the classification space down to binary. For this reason, the comparison in Table 3 between BiteSeer and the other datasets in Table 1 is not entirely direct, which is typical when switching from multiclass to binary classification. The split for each dataset comparison is a 10/10/80% for train, validation, and test. It is important to note that when creating our S-GNNs,

Table 3 | BiteSeer Results for SuperNeuroMAT

	Classes		Splits			Accuracy		Spikes	
	A	B	Train	Validation	Test	Validation	Test	Validation	Test
AI vs. Agents	52	406	20	20	418	80%	88%	8925	188,851
AI vs. DB	14	257	20	20	231	100%	94%	5032	53,680
AI vs. HCI	7	207	20	20	174	80%	88%	8925	188,851
AI vs. IR	20	447	20	20	427	85%	95%	10,120	216,275
AI vs. ML	13	150	20	20	123	95%	89%	3688	23,871
Agents vs. DB	388	294	68	68	546	93%	90%	28,187	272,933
Agents vs. HCI	405	262	66	66	535	77%	87%	38,596	326,391
Agents vs. IR	396	456	85	85	682	93%	95%	63,741	488,502
Agents vs. ML	393	172	56	56	453	88%	93%	20,773	208,233
DB vs. HCI	259	213	47	47	378	85%	94%	11,600	96,365
DB vs. IR	306	485	79	79	733	75%	87%	57,958	502,287
DB vs. ML	261	154	41	41	333	95%	96%	12,271	103,197
HCI vs. IR	235	452	68	68	551	88%	90%	32,127	346,643
HCI vs. ML	209	153	36	36	290	86%	89%	9284	83,960
IR vs. ML	465	214	67	67	545	75%	87%	49,285	423,467

Table 4 | BiteSeer dataset sizes

	A	B	Papers	Cites	Features	Neurons	Synapses
AI vs. Agents	52	406	458	708	2233	460	3064
AI vs. DB	14	257	271	433	1724	273	1840
AI vs. HCI	7	207	214	283	1581	216	1336
AI vs. IR	20	447	467	1019	2299	469	3718
AI vs. ML	13	150	163	259	1372	165	1104
Agents vs. DB	388	294	682	1073	2659	684	4600
Agents vs. HCI	405	262	667	980	2609	669	4360
Agents vs. IR	396	456	852	1613	2924	854	6292
Agents vs. ML	393	172	565	883	2513	567	3800
DB vs. HCI	259	213	472	694	2299	474	3086
DB vs. IR	306	485	791	1553	2783	793	5952
DB vs. ML	261	154	415	662	2209	417	2818
HCI vs. IR	235	452	687	1317	2722	689	5106
HCI vs. ML	209	153	362	527	2098	364	2356
IR vs. ML	465	214	679	1474	2641	681	5392

"Features" counts the number of features appearing at least once. Neuron and Synapse counts are for the neuromorphic SuperNeuroMAT implementation.

we use fixed-size splits for any classification with the AI topic. This is because few papers on the AI topic exist in each dataset, causing our dataset to allocate too many AI papers into a single split, which hinders accurate training, as shown in Table 4. This limitation can be addressed by creating a more robust splitting process to ensure even distribution of papers to individual splits; furthermore, even after fixing the splits, the accuracy performance remained comparable to other topic-vs-topic comparisons.

In lieu of measuring the actual energy usage of a specific hardware implementation, we report the spike counts from the simulations. Table 5 details the runtime and spike counts that we observed when simulating the S-GNN with SuperNeuroMAT. For BiteSeer, in Table 3 we report averaged timings and spike counts across all topic comparisons.

While the datasets introduced in this work are intentionally compact, they reflect the practical constraints of currently accessible neuromorphic

hardware. As summarized in Table 6, mixed-signal, digital, FPGA-based, and NVM-based neuromorphic platforms support different subsets of the proposed datasets depending on neuron and synapse capacity. In this context, MicroSeer and BiteSeer align well with the capabilities of a broad range of present-day neuromorphic substrates, enabling meaningful hardware benchmarking without aggressive time-multiplexing or off-chip memory.

Discussion

In this work, we present a suite of CiteSeer-derived datasets to help drive the evolution of the graph neural network community. We show our implementation of these datasets with two spiking neural network approaches, Neuromorphic Graph Learning and SpikingGCN, and we show the integrity of these datasets across different environments. We describe how these datasets may be used in the process of software, hardware, and algorithm co-

design, and we present baseline results to show the effect that shrinking the datasets has when used with the neuromorphic approach.

We identified a gap in available datasets suitable for running on real neuromorphic hardware, particularly those small enough to fit within the constraints of current systems. We thought it was prudent to create smaller, more connected datasets for the graph neural network community to fill this gap. These datasets will act as a bridge while neuromorphic hardware has time to scale up in the more general community. We understand that not every researcher has the hardware to run a very large network (i.e., one with thousands of neurons) on hardware.

We acknowledge that the results using S-GNN do not utilize possible optimizations such as Bayesian optimization (BO) or the use of features as described in other works on S-GNN³⁵. We, in good faith, want to be transparent with the results that we achieved with this dataset and set some baseline results for the larger community, even though higher accuracies are likely still achievable.

That said, it is inevitable that there will be some differences in the accuracy, runtime, or spike count when using these datasets, subject to the simulator/hardware being utilized, or the implementation or node-classification approach used. For example, both the SuperNeuroMAT simulator and the aforementioned NeuroCoreX platform support bidirectional all-to-all connectivity and online STDP-learning, but NeuroCoreX is subject to additional hardware constraints—limited bit precision of the weights, fixed-point representation of neuron internal states, and simplified implementations of the STDP learning rule^{41–44}.

This is also the reason we report spike counts, rather than estimated total energy usage for a specific platform. A direct measurement of energy would then only be comparable to measurements from other works on that platform. We expect our datasets to be implementable not only in simulators, but also on FPGAs, analog ASICs, and memristive crossbars. These platforms differ in static and dynamic power, leakage, and switching behavior. For example, in analog designs, energy per spike can be as low as a few femtojoules. In contrast, FPGAs may consume energy in the nanojoule range per spike. As our contribution is a set of datasets rather than any improvement in overall energy usage, we report spike count in Tables 3 and 5. This hardware-independent metric allows for extrapolating estimated energy usage for any target platform. Along with the neuron and

synapse counts for the neuromorphic graph learning implementation in Tables 7 and 8, we hope this aids in the selection of a benchmark dataset to fit the constraints for a particular platform.

We designed these datasets to be utilized on such size-limited dedicated neuromorphic hardware, but we first exemplify their usage in simulation with the neuromorphic graph learning and SpikingGCN approaches. It is a prudent step to show results for the community to visualize what is expected from these datasets, and how the transition to hardware can be the next step. We recognize that, like many machine-learning benchmarks, ours may eventually be superseded as neuromorphic hardware evolves. However, in the near term, we believe these benchmarks offer significant value to the growing community exploring neuromorphic systems. We hope that in publishing the methods and resources to parametrically re-size the datasets also aids in the testing and design of neuromorphic systems, particularly in regards to algorithm-software-hardware co-design. The flexibility of the benchmarks enables those developing these systems to scale the dataset as the network size constraints loosen. In the longer term, when the neuromorphic community has grown out of any need to use datasets smaller than the original CiteSeer, we envision these benchmarks being a useful in an educational context in the same way MNIST⁴⁵ is, as the smaller networks are easy to comprehend.

We hope that the detailed explanation of the datasets and the information provided about the results can help fuel utilization and reproducibility in the community and beyond.

Methods

We chose CiteSeer as a base dataset to modify, as CiteSeer's largest single component is the smallest out of that of Pubmed, Cora, and CiteSeer. In this section, we will describe how we derive smaller datasets from CiteSeer.

CiteSeer

The datasets we present in this paper are derivatives of the CiteSeer dataset²². CiteSeer is a citation network consisting of 3312 papers and 4715 citations. Each paper in the dataset is a node on the citation network. If paper A cites paper B in the dataset, then there exists an edge from node A to node B in the citation network. Each paper in the original dataset has an associated topic, which can be one of the following six topics: Artificial Intelligence, Agents, Databases, Human-Computer Interaction, Information Retrieval, or Machine Learning. These are abbreviated to AI, Agents, DB, HCI, IR, and ML, respectively. The dataset also includes 3703 binary features for each paper, representing if a paper includes (1) or does not include (0) a particular token. These tokens are similar to the bag-of-words representation constructed from all the papers in the dataset. However, the CiteSeer dataset itself does not include information on what word is represented by a particular element in the feature vector, only that certain papers share that word. The creators filtered out stem words and only included words used at least 10 times across all documents.

We define the graph for CiteSeer such that each paper is represented by a single vertex or node in the graph G , resulting in N nodes for N papers connected by the set of vertices $V = \{1, 2, \dots, N\}$. If paper i cites paper j , this is represented by the directed edge $(i, j) \in E$. This forms the graph $G(V, E)$. By

Table 5 | Runtime (Minutes:Seconds) and Spike Count for SuperNeuroMAT

	Runtime		Spike Count	
	Validation	Test	Validation	Test
MicroSeer	00:00.74	00:00.70	2368	2104
MiniSeer	00:05.27	00:24.65	154,863	2,499,494
BiteSeer	00:00.72	00:01.53	23,740	223,989
	± 00:00.16	± 00:00.56	± 20,479.67	± 163,824.43
CiteSeer	00:08.49	01:00.93	105,152	2,546,296

For BiteSeer, we report the mean and standard deviation across runs of every pair.

Table 6 | Representative small-scale neuromorphic hardware platforms and their neuron, synapse, and learning capabilities

Hardware Platform	Max Neurons (Synapses)	Weight Precision (STDP)	Suited Dataset
DYNAP-SE2 (Mixed-signal) ⁴⁸	~1024 (~65k)	6-bit mixed-signal (Enabled)	MicroSeer, MiniSeer
ODIN-256 (Digital ASIC) ⁴⁹	256 (~64k)	4-bit digital SRAM (Enabled)	MicroSeer
CMOL-SBS (CMOS+Memristor) ⁵⁰	64–128 (~4k)	Binary memristor (Enabled)	BiteSeer, MicroSeer
FPGA-based SNN (NeuroCoreX) ⁴¹	~100 (10k)	8-bit fixed-point (Enabled)	BiteSeer, MicroSeer
Daffodil (Memristor+FPGA in-the-loop) ¹⁴	~50–100 (~20k)	3-bit memristor (Enabled)	BiteSeer, MicroSeer

The supported dataset indicates which of the proposed benchmarks can be realistically mapped without excessive time-multiplexing or off-chip memory.

methodically removing nodes and their associated edges from this graph, we can create smaller datasets, each with a different purpose. In the following sections, we present the derived datasets that we created: MiniSeer, MicroSeer, and BiteSeer.

1. MiniSeer: The largest graph component of CiteSeer, which allows us to examine the effects of removing small components on link-based classification accuracy.
2. MicroSeer: An 84-paper dataset that allows for the implementation of S-GNN on neuron-constrained neuromorphic hardware.
3. BiteSeer: A collection of 15 binary (two-topic) classification datasets that allow for the implementation of S-GNN on synapse-constrained

Table 7 | Sizes of CiteSeer, MiniSeer, and MicroSeer

	Papers	Cites	Features	Neurons	Synapses
CiteSeer	3312	4732	3703	3318	47,616
MiniSeer	2110	3757	3603	2116	31,456
MicroSeer	84	86	1227	90	876

“Features” counts the number of features appearing at least once. We also report the Neuron and Synapse counts for the S-GNN.

Table 8 | Topic distribution of CiteSeer, MiniSeer, and MicroSeer

	AI	Agents	DB	HCI	IR	ML
CiteSeer	249	596	701	508	668	590
MiniSeer	115	463	388	304	532	308
MicroSeer	3	20	12	26	15	8

neuromorphic hardware, with more complex and varied network topologies compared to that of MicroSeer.

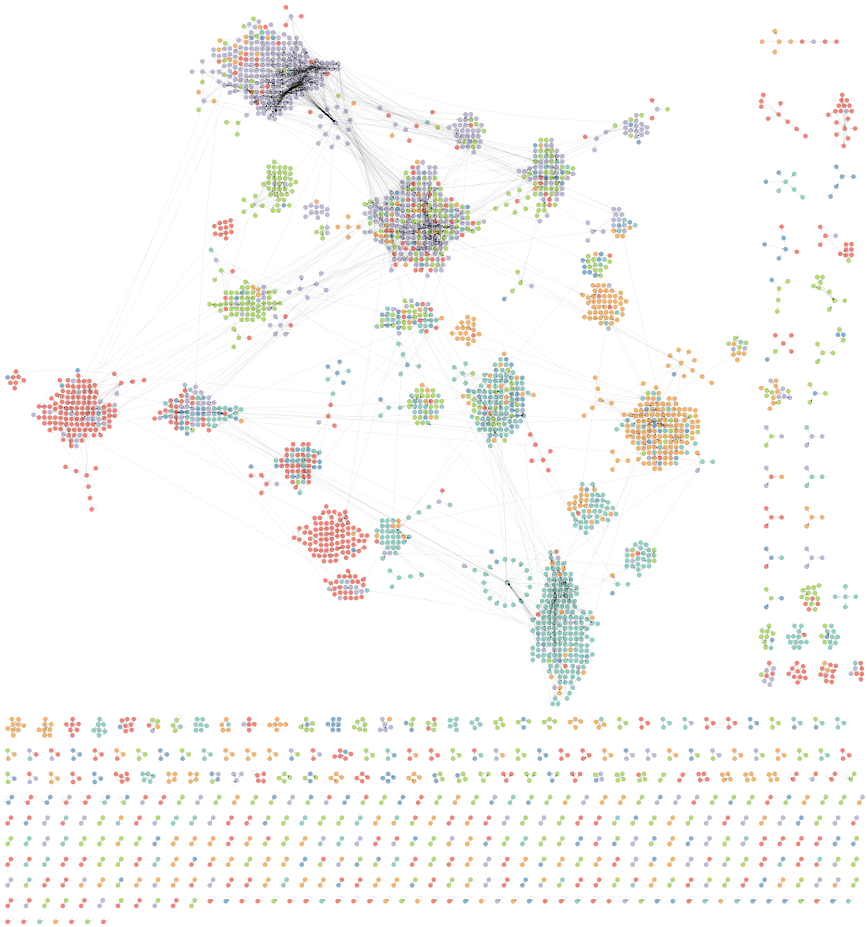
MiniSeer

MiniSeer is the largest graph component of CiteSeer, which allows us to examine the effects of removing small components on purely link-based classification, but more importantly serves as a base for our other datasets. It is formed by assuming the graph edges as undirected, and then taking the largest connected component of CiteSeer. That is, for the original CiteSeer citation graph G , MiniSeer is defined as follows: Consider that graph G contains j connected components. We denote the connected components of the graph as $\mathcal{C}(G) = \{S_1, S_2, \dots, S_j\}$, where each component $S_i, i \in 0, 1, \dots, j$ is fully connected. We create MiniSeer from the largest connected component, $G_{mini} = \arg \max_{S \in \mathcal{C}(G)} |S|$, where $|S|$ represents the number of nodes in component S . While we consider undirected connections between papers during component analysis, the resulting datasets still provide direction information for each edge.

This dataset was created to determine if the large number of disjoint components in the CiteSeer graph significantly affects our performance in classification, as small components have fewer connections between papers, and we theorize that this weaker association between papers of the same topic results in weaker classification accuracy. The effects of this can most easily be seen in Fig. 2, where MiniSeer is the largest connected component, shown in the top left of the figure. The smaller components of CiteSeer are shown along the right and bottom edges of the figure.

All datasets created in this paper—MiniSeer, MicroSeer, and BiteSeer—retain the full binary feature vector for each paper. We keep the entire feature vector so that it is still possible to cross-reference features from the larger dataset, even though this means many features are not

Fig. 2 | Graph of CiteSeer, generated using yEd with clustering by Louvain modularity.



well represented in the resulting dataset, i.e., we do not guarantee that each word appears in a document in the dataset. In fact, there are 3604 active features in the dataset; 99 of the binary features have no papers that use the associated word. The removal of the smaller components has affected the distribution of papers across topics, as can be seen in Table 8 and Fig. 3. We publish these derived datasets online along with the Python code used to generate them.

Despite the name, the primary feature of MiniSeer is not its smaller size (although it is smaller than CiteSeer). Since CiteSeer has many smaller components, we felt it would not make sense to split these 2–10 paper components up further, as link-based classification would be pointless on such small components. This is especially the case when random training and test splits are considered, as most of these smaller components would lack nodes from the training set. Thus, creating

MiniSeer allows us to examine how much of CiteSeer is impossible to classify using purely link-based classification methods for a given split. Moreover, it provides a base for the following even smaller datasets, which should certainly only contain a single graph component.

MicroSeer

MicroSeer is an 84-paper dataset that allows for the implementation of an S-GNN on neuron-constrained neuromorphic hardware. Considering the six topics, it allows a network of $84 + 6 = 90$ neurons to be constructed on hardware limited by neuron count. It is constructed by randomly removing nodes from the graph while maintaining connectivity of the undirected graph, as shown in Algorithm 1. We chose to preserve connectivity and have only one component, as we posit that such a small, fragmented dataset of many components would be difficult to infer classes from. Like with MiniSeer, we did not change the length of the feature vector, but there are now only 1227 features used. It should be noted that although the paper removal uses a uniform sample, the conditioning of connectivity destroys any guarantee of maintaining the distribution of topics across the dataset.

Algorithm 1. Creating MicroSeer with N papers

Input: *miniseer*

Output: *miniseer*

Initialization :

1: *microseer* $\leftarrow$ **copy**(*miniseer*)

2: *candidates* $\leftarrow$ **copy**(*microseer*)

3: **set_seed**(random_paper, 2025)

LOOP Process

4: **while** |*microseer*| > N **do**

5: *paper* $\leftarrow$ **random_paper**(*candidates*)¹

6: *test_graph* $\leftarrow$ **copy**(*microseer*) \ {*paper*}

7: **if is_connected**(*test_graph*) **then**

8: *microseer* $\leftarrow$ *test_graph*

9: *candidates* $\leftarrow$ **copy**(*microseer*)



Fig. 3 | Relative sizes of datasets.

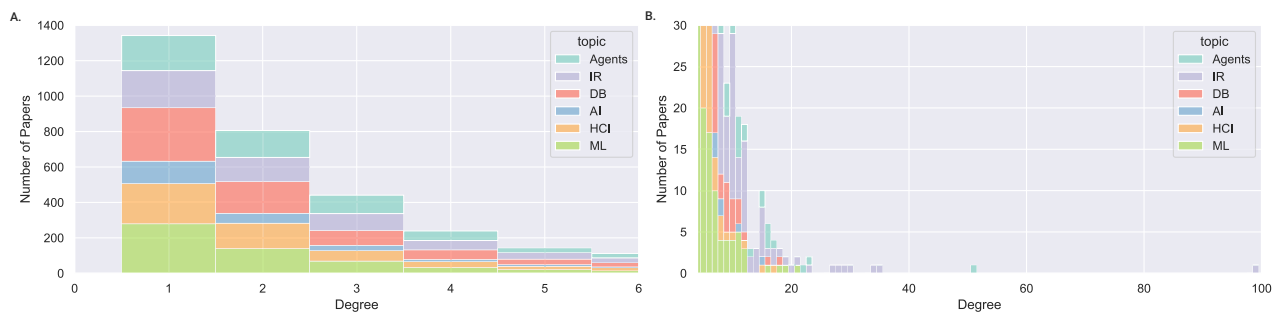
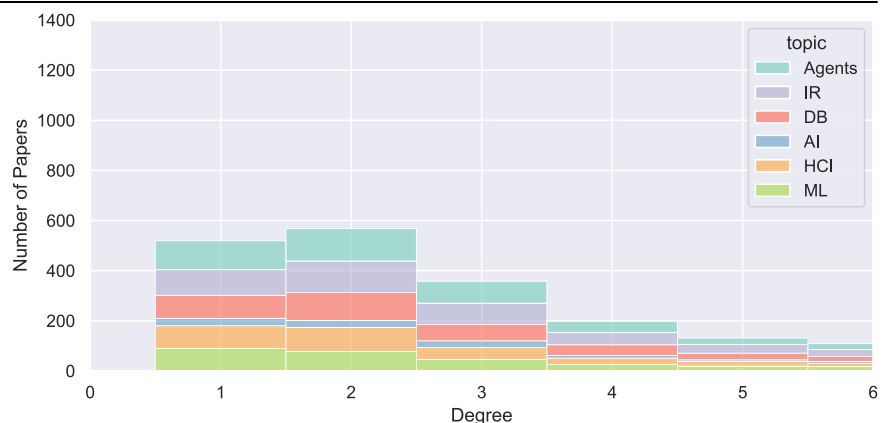


Fig. 4 | Histogram of degree of nodes for CiteSeer. In this figure, “degree” refers to the number of papers that cite or are cited by a certain number of papers. **A** Shows the paper count for papers up to a degree of six, **B** shows the count for degrees higher than six. Note the y axis is not shared between (**A**, **B**).

Fig. 5 | Histogram of papers per degree for MiniSeer. We do not show the histogram for degree > 6 as it appears similar to the histogram shown in Fig. 4B. Note the significant reduction in papers with degree < 3.



```

10: else
11:    $candidates \leftarrow candidates \setminus \{paper\}$ 
12: end if
13: end while
14: return microseer

```

Alternatives to this method of dataset construction were considered. We considered forcing the distribution of papers in MicroSeer to match that of MiniSeer by removing papers in a manner that minimally impacts the ratio of the number of papers in each topic, but we did not see a reason to impose this constraint, as we wanted to preserve the original class balance of CiteSeer. We also considered growing a network from a seed paper chosen from MiniSeer, which naturally guarantees that the result will be a single, connected component. However, this resulted in a heavily skewed dataset, where most papers were of a single topic. This was the case even if the seed paper was chosen to have citations with as many as five of the six topics, as no paper in CiteSeer cites papers from all six topics.

We envision MicroSeer as a basic testbed for validating performance on neuromorphic processors that are highly constrained in neuron and memory capacity. The 90-neuron network size target is specifically chosen to accommodate small-scale FPGA-based platforms with limited BRAM resources. In particular, MicroSeer serves as a suitable benchmark for platforms like NeuroCoreX, an open-source, easily configurable, and user-friendly hardware framework⁴⁶. Implemented on a low-cost Artix-7 FPGA, NeuroCoreX currently supports up to 100 neurons and 10,000 synapses.

We intend to implement the S-GNN model using the NeuroCoreX platform, and its current hardware limits align well with the dataset's scale. Although MicroSeer is derived by shrinking the MiniSeer dataset down to 84 papers, the same methodology can be applied to generate scaled-down versions of other datasets to match the constraints of any target neuromorphic system. While MicroSeer can certainly be used to evaluate algorithmic performance, our primary goal in creating it is to validate whether the performance of our neuromorphic graph learning approach transfers reliably across different hardware implementations. Once the model is

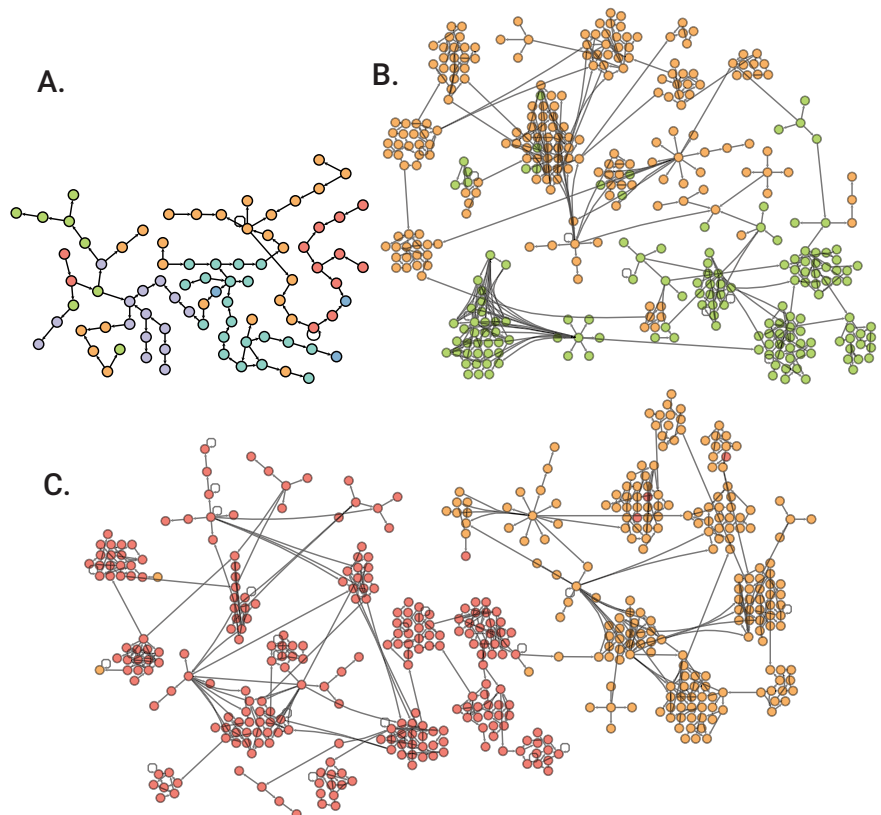
tuned through co-design on the NeuroCoreX platform using MicroSeer, future work will focus on validating its performance on larger datasets using more advanced neuromorphic platforms capable of supporting greater numbers of neurons.

BiteSeer

BiteSeer is a collection of 15 binary (two topic) classification datasets which allow for the implementation of S-GNN on synapse-constrained neuromorphic hardware, with more complex and varied network topologies compared to that of MicroSeer. The original six topics of MiniSeer provide for $\binom{6}{2} = 6!/(2!(6-2)!) = 15$ binary classification datasets, such as “AI vs. ML”, “ML vs. HCI”, etc. Each BiteSeer dataset is created by removing all papers that do not belong to either of the two topics, and then picking the single largest component. This provides no guarantee that the ratio between the two topics is even, or that the ratio matches that of CiteSeer or MiniSeer, as can be seen in Figs. 4 and 5. However, we chose to impose connectedness because the alternative of simply taking only papers in two topics would result in each dataset having many singleton components, as well as many small components with only two or three papers. The accuracies shown in Table 3 show that BiteSeer datasets have a higher baseline accuracy than other datasets. So, although these datasets *can* be used to compare different SNN-based approaches, we do not recommend them to be used for this purpose, especially if the network size is not limited by hardware. Thus, the BiteSeer datasets should not be used as a proxy for the full 6-class problem, such as a One vs. All classifier, but rather as a tool for benchmarking hardware with a constraint on the number of neurons or synapses, for example, memristive arrays¹³.

While any processor capable of containing a BiteSeer dataset would likely also be able to contain MicroSeer, the diminutive size of the network and the methods used to construct it have naturally led to a very simple graph structure. As a result, almost every edge is a separating edge, and most nodes have a degree of one, that is, a node with only one other associated paper. By visually inspecting the graph shown in Fig. 6, we can see that only a single

Fig. 6 | Visualizations of the generated citation networks. **A** MicroSeer graph, **B** BiteSeer DB vs. HCI graph, **C** BiteSeer HCI vs. ML graph.



edge between nodes can be removed without splitting the graph into two components. Conversely, BiteSeer datasets have more complex structures, such as larger Louvain communities⁴⁷, longer chains, and higher degrees of connections per node throughout the graph. This allows for testing of somewhat more complex graph structures on synapse-constrained neuromorphic hardware compared to MicroSeer. As shown in Table 4, BiteSeer datasets can be used with the neuromorphic graph learning approach on hardware with 6300 synapses or fewer, whereas MiniSeer would require over 30,000 synapses, as shown in Table 7. This sizing allows BiteSeer to be run using platforms such as the Daffodil platform, containing a ReRAM crossbar array of 20,000 devices¹⁴.

Furthermore, consider that analog and mixed-signal platforms introduce a notion of device variability, both in terms of chip-to-chip variations as well as between devices on the same chip. The 2016 ReRAM work notes a 74% yield on their wafer, for example¹³. It is even possible that some applications may require tighter constraints on the quality of devices, further lowering the usable device count. We predict that other new hardware platforms will encounter similar constraints which may necessitate even smaller datasets than the ones we present. Of course, we hope that this hardware variability improves. However, we cannot predict how neuromorphic hardware will evolve over time. The hardware variability and expectation of improvement for neuromorphic platforms are a primary motivating factor for the breadth of datasets that we show here, as well as the reason why we release the code and approaches used to generate these datasets to the public. We envision that smaller datasets would not be useful for benchmarking algorithms. Instead, hardware designers and researchers would use our work here as a guideline to parametrically create datasets for testing hardware at multiple points in development.

Data availability

Datasets and Source Code available: <https://github.com/ORNLCiteBench>.

Received: 7 October 2025; Accepted: 9 April 2026;

Published online: 01 June 2026

References

- Wu, J., Wang, Y., Li, Z., Lu, L. & Li, Q. A Review of Computing with Spiking Neural Networks. *Computers, Mater. Contin.* **78**, 2909 (2024).
- Yamazaki, K., Vo-Ho, V.-K., Bulsara, D. & Le, N. Spiking Neural Networks and Their Applications: A Review. *Brain Sci.* **12**, 863 (2022).
- Lee, D., Li, Y., Kim, Y., Xiao, S. & Panda, P. Spiking Transformer with Spatial-Temporal Attention. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 13948–13958 (2025).
- Sun, Y. et al. SpikeGraphormer: A High-Performance Graph Transformer with Spiking Graph Attention <http://arxiv.org/abs/2403.15480> (2024).
- Zhu, Z. et al. Spiking Graph Convolutional Networks <http://arxiv.org/abs/2205.02767> (2022).
- Yin, N. et al. Continuous spiking graph neural networks. *arXiv preprint arXiv:2404.01897* (2024).
- Yin, N. et al. Dynamic Spiking Graph Neural Networks. *Proc. AAAI Conf. Artif. Intell.* **38**, 16495–16503 (2024).
- Schuman, C. D. et al. Shortest Path and Neighborhood Subgraph Extraction on a Spiking Memristive Neuromorphic Implementation. In *Proceedings of the 7th Annual Neuro-inspired Computational Elements Workshop, NICE '19*, 1–6 <https://doi.org/10.1145/3320288.3320290> (Association for Computing Machinery, New York, NY, USA, 2019).
- Xiao, M., Zhu, Y., He, D. & Lin, Z. Temporal Spiking Neural Networks with Synaptic Delay for Graph Reasoning <http://arxiv.org/abs/2405.16851> (2024).
- Storesund, S., Aars, K. V., Dietrich, R. & Waniek, N. Predictive Spike Timing Enables Distributed Shortest Path Computation in Spiking Neural Networks <http://arxiv.org/abs/2509.10077> (2025).
- Davies, M. et al. Loihi: A Neuromorphic Manycore Processor with On-Chip Learning. *IEEE Micro* **38**, 82–99 (2018).
- Gonzalez, H. A. et al. SpiNNaker2: A Large-Scale Neuromorphic System for Event-Based and Asynchronous Machine Learning <http://arxiv.org/abs/2401.04491> (2024).
- Adam, G. C. et al. 3-D memristor crossbars for analog and neuromorphic computing applications. *IEEE Trans. Electron Devices* **64**, 312–318 (2016).
- Hoskins, B. et al. A System for Validating Resistive Neural Network Prototypes. In *International Conference on Neuromorphic Systems 2021, ICONS 2021*, 1–5. <https://doi.org/10.1145/3477145.3477260> (Association for Computing Machinery, New York, NY, USA, 2021).
- Mikolajick, T., Park, M. H., Begon-Lours, L. & Slesazek, S. From Ferroelectric Material Optimization to Neuromorphic Devices. *Adv. Mater.* **35**, 2206042 (2023).
- Sengupta, A. & Roy, K. Neuromorphic computing enabled by physics of electron spins: Prospects and perspectives. *Appl. Phys. Express* **11**, 030101 (2018).
- Liu, Z. et al. SUSHI: Ultra-High-Speed and Ultra-Low-Power Neuromorphic Chip Using Superconducting Single-Flux-Quantum Circuits. In *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO '23*, 614–627. <https://doi.org/10.1145/3613424.3623787> (Association for Computing Machinery, New York, NY, USA, 2023).
- Buckley, S. et al. Design of Superconducting Optoelectronic Networks for Neuromorphic Computing. In *2018 IEEE International Conference on Rebooting Computing (ICRC)*, 1–7. <https://ieeexplore.ieee.org/abstract/document/8638595> (2018).
- Luo, J., Chen, C., Huang, Q. & Huang, R. A Biomimetic Tunnel FET-Based Spiking Neuron for Energy-Efficient Neuromorphic Computing With Reduced Hardware Cost. *IEEE Trans. Electron Devices* **69**, 882–886 (2022).
- Li, S., Gao, L., Liu, C., Guo, H. & Yu, J. Biomimetic Neuromorphic Sensory System via Electrolyte Gated Transistors. *Sensors* **24**, 4915 (2024).
- Qiao, N. et al. A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128K synapses. *Front. Neurosci.* **9**, <https://doi.org/10.3389/fnins.2015.00141> (2015).
- Lu, Q. & Getoor, L. Link-based classification. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning, ICML'03*, 496–503 (AAAI Press, Washington, DC, USA, 2003).
- Date, P. et al. SuperNeuro: A Fast and Scalable Simulator for Neuromorphic Computing <http://arxiv.org/abs/2305.02510> (2023).
- Bian, T. et al. Rumor Detection on Social Media with Bi-Directional Graph Convolutional Networks. *Proc. AAAI Conf. Artif. Intell.* **34**, 549–556 (2020).
- Michail, D., Kanakaris, N. & Varlamis, I. Detection of fake news campaigns using graph convolutional networks. *Int. J. Inf. Manag. Data Insights* **2**, 100104 (2022).
- Wu, S., Sun, F., Zhang, W., Xie, X. & Cui, B. Graph Neural Networks in Recommender Systems: A Survey. *ACM Comput. Surv.* **55**, 97:1–97:37 (2022).
- Gobin, D. et al. Exploration of Novel Neuromorphic Methodologies for Materials Applications <http://arxiv.org/abs/2405.04478> (2024).
- Merchant, A. et al. Scaling deep learning for materials discovery. *Nature* **624**, 80–85 (2023).
- Motie, S. & Raahemi, B. Financial fraud detection using graph neural networks: A systematic review. *Expert Syst. Appl.* **240**, 122156 (2024).
- Suomela, S. *Evaluation of Node Classification Methods in Citation Networks*. Ph.D. thesis, University of Helsinki <https://helda.helsinki.fi/items/41bdfed8-67e2-49bf-9f9d-ae1e336d438e> (2021).
- Kipf, T. N. & Welling, M. Semi-Supervised Classification with Graph Convolutional Networks <http://arxiv.org/abs/1609.02907> (2017).

32. Zhu, Z. et al. Spiking Graph Convolutional Networks. In Raedt, L. D. (ed.) *Thirty-First International Joint Conference on Artificial Intelligence*, vol. 3, 2434–2440. <https://www.ijcai.org/proceedings/2022/338> (2022).
33. Xing, X. et al. Spikellm: Scaling up spiking neural network to large language models via saliency-based spiking. In *The Thirteenth International Conference on Learning Representations (ICLR)* <https://dblp.org/rec/conf/iclr/XingGLCX0DZLZ25> (2025).
34. Cong, G. et al. Semi-Supervised Graph Structure Learning on Neuromorphic Computers. In *Proceedings of the International Conference on Neuromorphic Systems 2022, ICONS '22*, 1–4. <https://doi.org/10.1145/3546790.3546821> (Association for Computing Machinery, New York, NY, USA, 2022).
35. Cong, G. et al. Hyperparameter Optimization and Feature Inclusion in Graph Neural Networks for Spiking Implementation. In *2023 International Conference on Machine Learning and Applications (ICMLA)*, 1541–1546. <https://ieeexplore.ieee.org/abstract/document/10459955> (2023).
36. Eshraghian, J. K. et al. Training spiking neural networks using lessons from deep learning. *Proc. IEEE* **111**, 1016–1054 (2023).
37. Gewaltig, M.-O. & Diesmann, M. Nest (neural simulation tool). *Scholarpedia* **2**, 1430 (2007).
38. Bekolay, T. et al. Nengo: a Python tool for building large-scale functional brain models. *Front. neuroinformatics* **7**, 48 (2014).
39. Stimberg, M., Brette, R. & Goodman, D. F. Brian 2, an intuitive and efficient neural simulator. *eLife* **8**, e47314 (2019).
40. Kulkarni, S. R., Parsa, M., Mitchell, J. P. & Schuman, C. D. Benchmarking the performance of neuromorphic and spiking neural network simulators. *Neurocomputing* **447**, 145–160 (2021).
41. Gautam, A. et al. Neurocorex: An open-source FPGA-based spiking neural network emulator with on-chip learning. In *Neuro Inspired Computational Elements (NICE)* (2026).
42. Gautam, A. & Kohno, T. Adaptive STDP-based on-chip spike pattern detection. *Frontiers in Neuroscience* **17**, <https://www.frontiersin.org/https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2023.1203956/full> (2023).
43. Gautam, A., Kohno, T., Date, P., Patton, R. & Potok, T. A suppression-based STDP rule resilient to jitter noise in spike patterns for neuromorphic computing. In *2024 International Conference on Neuromorphic Systems (ICONs)*, 209–216 (IEEE, 2024).
44. Gautam, A. & Kohno, T. An adaptive stdp learning rule for neuromorphic systems. *Front. Neurosci.* **15**, 741116 (2021).
45. Deng, L. The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]. *IEEE Signal Process. Mag.* **29**, 141–142 (2012).
46. Gautam, A., Date, P., Kulkarni, S., Patton, R. & Potok, T. NeuroCoreX: Brain-Inspired Computing from Code to Circuit. Oak Ridge National Laboratory (ORNL), Oak Ridge, TN (United States) <https://www.osti.gov/doi/10.26434/chemrxiv-2025-15317> (2025).
47. Blondel, V. D., Guillaume, J.-L., Lambiotte, R. & Lefebvre, E. Fast unfolding of communities in large networks. *J. Stat. Mech.: Theory Exp.* **2008**, P10008 (2008).
48. Richter, O. et al. Dynap-se2: a scalable multi-core dynamic neuromorphic asynchronous spiking neural network processor. *Neuromorphic Comput. Eng.* **4**, 014003 (2024).
49. Frenkel, C., Lefebvre, M., Legat, J.-D. & Bol, D. A 0.086-mm² 12.7-pJ/SOP 64k-Synapse 256-Neuron Online-Learning Digital Spiking Neuromorphic Processor in 28-nm CMOS. *IEEE Trans. Biomed. Circuits Syst.* **13**, 145–158 (2019).
50. Camuñas-Mesa, L. A., Vianello, E., Reita, C., Serrano-Gotarredona, T. & Linares-Barranco, B. A CMOL-like memristor-CMOS neuromorphic chip-core demonstrating stochastic binary STDP. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **12**, 898–912 (2022).

Acknowledgements

The research was funded in part by the U.S. Department of Energy through awards DE-SC0025349, DE-SC0025654, and DE-AC05-00OR22725.

This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the US Department of Energy. This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility. The United States Government retains, and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (<http://energy.gov/downloads/doe-public-access-plan>).

Author contributions

K.Z. primarily contributed to the Datasets and Discussion sections, revised the manuscript, and generated the derived datasets and figures. I.M. primarily contributed to the Results and Discussion sections and generated all results for SuperNeuroMAT. P.D. primarily contributed to the Introduction, Related Work, Discussion, and Conclusion sections. A.G. contributed to the Discussion section regarding the potential hardware platforms and NeuroCoreX. T.N. contributed to the benchmarking and tuning for results for SuperNeuroMAT. S.S. contributed to the benchmarking and tuning for results for SuperNeuroMAT. S.K. assisted in revising and advising of Neuromorphic Graph Learning approach. S.H.L. generated all results shown for the SpikingGCN approach. G.C. heavily advised the work performed and focus of the manuscript. C.N. advised the writing process for K.Z. C.S. primarily contributed to the Introduction, as well as the advising of I.M. M.P. advised the overall process for K.Z. T.P. advised the project at large. All authors reviewed and approve of the manuscript.

Competing interests

Author Catherine Schuman serves as a member of the editorial board for NPJ Unconventional Computing. Catherine Schuman was not involved in the journal's review of, or decisions related to, this manuscript. Author Catherine Schuman declares no financial competing interests. All other authors declare no financial or non-financial competing interests.

Additional information

Correspondence and requests for materials should be addressed to Kevin Zhu, Ian Mulet, Catherine Schuman or Maryam Parsa.

Reprints and permissions information is available at <http://www.nature.com/reprints>

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

© The Author(s) 2026