Transductive Spiking Graph Neural Networks for Loihi

Shay Snyder*
ssnyde9@gmu.edu
George Mason University
Fairfax, Virginia, USA

Victoria Clerico*
mclerico@gmu.edu
George Mason University
Fairfax, Virginia, USA

Guojing Cong
congg@ornl.gov
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA

Shruti Kulkarni
kulkarnisr@ornl.gov
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA

Catherine Schuman
cschuman@utk.edu
University of Tennessee - Knoxville
Knoxville, Tennessee, USA

Sumedh R. Risbud
sumedh.risbud@intel.com
Intel Labs
Santa Clara, California, USA

Maryam Parsa
mparsa@gmu.edu
George Mason University
Fairfax, Virginia, USA

ABSTRACT

Graph neural networks have emerged as a specialized branch of deep learning, designed to address problems where pairwise relations between objects are crucial. Recent advancements utilize graph convolutional neural networks to extract features within graph structures. Despite promising results, these methods face challenges in real-world applications due to sparse features, resulting in inefficient resource utilization. Recent studies draw inspiration from the mammalian brain and employ spiking neural networks to model and learn graph structures. However, these approaches are limited to traditional Von Neumann-based computing systems, which still face hardware inefficiencies. In this study, we present a fully neuromorphic implementation of spiking graph neural networks designed for Loihi 2. We optimize network parameters using Lava Bayesian Optimization, a novel hyperparameter optimization system compatible with neuromorphic computing architectures. We showcase the performance benefits of combining neuromorphic Bayesian optimization with our approach for citation graph classification using fixed-precision spiking neurons. Our results demonstrate the capability of integer-precision, Loihi 2 compatible spiking neural networks in performing citation graph classification with comparable accuracy to existing floating point implementations.

CCS CONCEPTS

• **Computing methodologies** → **Bio-inspired approaches**; • **Theory of computation** → **Dynamic graph algorithms**; • **Hardware** → **Emerging technologies**.

*These authors contributed equally to this work

This work is licensed under a Creative Commons Attribution International 4.0 License.

GLSVLSI '24, June 12–14, 2024, Clearwater, FL, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0605-9/24/06
https://doi.org/10.1145/3649476.3660366

KEYWORDS

graph neural networks, spiking neural networks, transductive learning

ACM Reference Format:
Shay Snyder, Victoria Clerico, Guojing Cong, Shruti Kulkarni, Catherine Schuman, Sumedh R. Risbud, and Maryam Parsa. 2024. Transductive Spiking Graph Neural Networks for Loihi. In *Great Lakes Symposium on VLSI 2024 (GLSVLSI '24), June 12–14, 2024, Clearwater, FL, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3649476.3660366

1 INTRODUCTION

Inspired by its success in image processing, researchers have recently adapted the concept of convolution to analyze the relationships within graph structures. This technique is termed graph convolution [24]. Compared to traditional convolutional neural networks that extract features from local receptive fields [16], graph convolutional neural networks (GCNN) extract information from heterogeneous graph structures by understanding the relationship between interconnected nodes [24]. This learning approach has been successfully applied to a variety of real-world machine learning applications such as material design [22], text classification [23], citation graph classification [6], and pharmaceutical discovery [21].

Graph neural networks are particularly effective in applications with high-levels of sparsity by employing semi-supervised learning [9, 12, 14]. Popular benchmarks such as Sen et al. [19] achieved high accuracy despite having only 6% of the graph nodes labeled as training. While existing approaches can achieve high accuracy, they often struggle to efficiently model real-world graphs. These data structures are characterized by a mixture of highly interconnected regions and immensely sparse areas. Popular hardware accelerators, such as graphic processing units, struggle to efficiently utilize computational resources when processing irregularly clustered computational graphs since their design is optimized for dense workloads.

Neuromorphic computing and biologically inspired networks have emerged as prevalent research fields to address sparse data representations. Recent works demonstrate the performance of spiking neural networks for a variety of graph learning tasks such as the

minimum spanning tree [13] and shortest path problems [2, 18]. These learning approaches posses unique properties making them especially attractive for implementing graph learning algorithms through event-driven communication between individual computational units (spiking neurons). Moreover, research investment from companies such as Intel [8] and IBM [3] has lead to the development of ASIC processors specifically designed to perform these neuromorphic computations.

Existing applications of GCNNs extract hidden features within the network, where the convolution operation is performed over adjacent nodes and information from neighboring nodes is iteratively propagated and aggregated [5]. Similarly, other methods employ graph attention over the neighborhood of each node and learn graph embeddings [1]. While existing methods utilize a separate neural network for processing the graph, our approach builds the network structure directly based on graph anatomy. This transforms learning into a transductive approach where learning happens within the citation graph itself. Additionally, existing methods using spiking neural networks for transductive learning have been limited to simulated, floating point environments that are not natively compatible with neuromorphic hardware [4, 7]. This paper investigates the capability of fixed-precision spiking neural networks designed for Intel's neuromorphic processor, Loihi 2 [8].

We introduce a new, fixed-precision neuron model (extending from the traditional leaky-integrate-and-fire (LIF) known as **LIF Long Reset**. This neuron model allows the network to fully reset at predefined intervals by setting all currents and voltages to zero for multiple timesteps. In tandem with Lava Bayesian Optimization [20], we optimize network parameters to achieve performance approaching existing floating point methods while limiting intra-network calculations to integer precision. The key contributions of this work are summarized as follows:

- We provide a novel fixed precision spiking neural network model for transductive graph learning in Lava.
- We introduce the LIF Long Reset neuron model that allows the network to self-stabilize at pre-defined intervals while also contributing to the Lava software framework.
- Our integer-precision implementation in Lava exhibits similar performance to an existing floating point implementation in NEST [4].

2 RESEARCH METHODS

Citation network graphs consist of interconnected paper nodes based on their citations. Each node represents a distinct research paper, while edges signify citations between papers. Moreover, each paper is labeled with a specific paper topic. The citation graph from Cora [17] includes 2708 paper nodes, 5258 edges, and 7 topics: rule learning, neural networks, case-based, genetic algorithms, theory, reinforcement learning, and probabilistic methods. Many citation network datasets also include a series of features linked to each paper. Our current implementation does not include this additional information.

We randomly select 20 nodes per paper topic to serve as the labeled training neurons. The remaining paper nodes are proportionally split into validation and testing neurons. This is represented in our spiking neural network by creating three individual clusters of spiking neurons; 140 for training, 140 for validation, and 2428 for testing. Lastly, we initialize another cluster of neurons representing the topic nodes with 7 neurons.

The paper neurons are initialized with a voltage decay of 0 and a voltage threshold equal to 1. For each citation (edge) in the citation graph from paper **A** to paper **B**, we create a corresponding synapse with an initial weight of 100. These synapses from paper **A** to paper **B** can occur within the same neuron cluster or interconnect neurons from different clusters. The process is repeated in the reverse direction from paper **B** to paper **A**, encouraging recurrent communication. Each paper neuron within the training cluster is directly connected to the correct topic neuron with a weight of 1 along with an identical reverse connection.

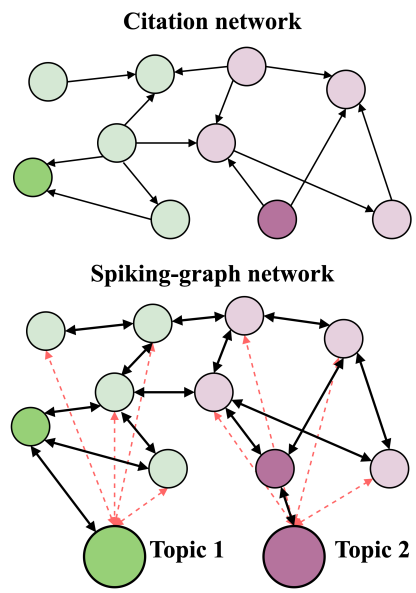

Figure 1: A simplified graph demonstrating the architecture of our spiking neural network for citation graph classification. Red dotted arrows represent the learnable synapses.

The testing and validation neurons have dense connectivity with all topic neurons and learnable synaptic weights. This is demonstrated in Figure 1. Using Hebbian learning rules [10], we employ bi-directional spike timing-dependent plasticity (STDP) to learn the correct paper-to-topic connection for each paper in the testing and validation sets.

STDP-based learning uses the temporal correlation between spikes registered in time t_{pre} from pre-synaptic neurons and spikes registered at t_{post} from post-synaptic neurons to perform localized learning. This mechanism strengthens the synaptic weight between neurons when the pre-synaptic neuron fires before the post-synaptic neuron and weakens the connection if the pre-synaptic neuron fires afterwords. These connections all begin with an initial weight of 0. The learning rule is configured with a learning rate of 2, $a_- = -1$, $a_+ = 1$, $\tau_- = 30$, and $\tau_+ = 30$[1]. These parameters are also summarized in Table 1.

[1]See Gerstner et al. [10] for more information on STDP parameters

To classify any given paper, we send a single spike at $t = 0$ to the corresponding paper neuron and allow it to propagate through the network for t_s time steps. At the end of this interval, we extract the predicted label by interpreting information from the trained graph. One strategy is to infer the learned topic as the topic neuron with the highest firing rate. Another approach is to choose the predicted label based on the learned synaptic weights between the specific paper and all topic neurons. To avoid encouraging increased synaptic activity, we employ the second method where the label is inferred via the corresponding topic to paper synaptic weights. With the topic (pre-synaptic) neurons firing after the paper (post-synaptic) neurons, the lowest weight (largest negative magnitude) is selected because STDP depresses the synaptic weight between neurons when the post-synaptic neuron fires before the pre-synaptic neuron.

To reset our dynamic spiking graph between paper evaluations, we created a new neuron based on the traditional leaky-integrate-and-fire (LIF) model, called the **LIF Long Reset** neuron. This neuron can intermittently stop firing at pre-specified reset intervals $\mathbf{T_r}$ for a given reset length. We constrain the reset interval T_r to be equal to the number of spike propagation steps such that $T_r = T_s$. This allows us to accurately decode the learned weights before the network is allowed to reach homeostasis. Our model also supports refractory dynamics so it will only fire once for any given paper evaluation. The performance implications and inter-neuron characteristics for this model are highlighted in Figure 2.

Developing and deploying spiking neural networks on Loihi 2 requires prototyping the model using Intel's neuromorphic framework Lava [15]. The fundamental blocks of Lava are *Processes*, which operate in parallel and asynchronously with each other. Lava's *Processes* have their own private memory and communicate with other processes through channel-based message passing. *Processes* can be used to implement a single neuron, a cluster of neurons, or an entire neural network at the complexity level of ResNet [15]. A *Process* is defined by *input ports*, *internal variables* and *output ports*. The *Processes* only define an interface with state variables and communication ports, therefore they do not provide a behavioral implementation. As such, the behavior is implemented in *Process Models*[2].

Our learning pipeline consists of one process for generating input spikes (*Spike Encoder*) and a subprocess model for the spiking graph neural network (*Spiking GNN*), see Figure 3. The *Spike Encoder* process is responsible for sending the single spike through the paper neuron under evaluation. The *Spiking GNN* subprocess model synchronously runs three types of processes: (1) *LIFLongReset* for the paper and topic neurons, (2) *Dense* for the fixed synaptic connections and (3) *Learning Dense* for the learnable synapses of the graph, see Figure 3. The learning rule for the learnable synapses is STDP-based (*STDPLoihi*) and is implemented as described in [10].

At the end of execution, classification accuracy is computed outside the process by taking the predicted labels for each paper and comparing with the ground truth. With this multi-process architecture in Lava, our spiking graph has full support for simulation with *fixed-precision, integer computations* as required by Loihi 2.

[2]See http://lava-nc.org for details about Lava concepts like *Process* and *Process Models*

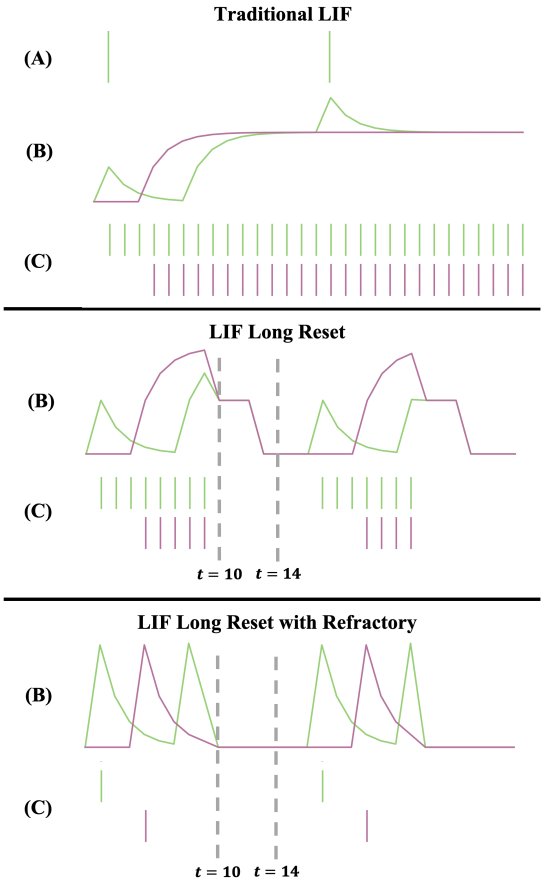

Figure 2: The inter-neuron dynamics of our LIF Long Reset neuron model compared to a traditional LIF neuron through two neurons recurrently connected with fully-connected layers. (A) Input spikes from the ring buffer (B) Synaptic current (C) Post synaptic spike rasters from both neurons. Each neuron was initialized with a current decay of 0.5. Each LIF Long Reset neuron variant was configured with a reset interval of 10 time steps with a reset period of 4 time steps.

Each of our spiking neuron clusters and instances of STDP have numerous parameters such as current decay, voltage decay, learning rate, and synaptic delays. With our architecture consisting of multiple clusters of neurons and multiple instances of STDP, it is extremely difficult to manually specify each individual parameter. To automate this process, we integrate our event-driven process architecture with Lava Bayesian Optimization (Lava BO) [20]. This framework provides a native, event-based framework for optimizing neuromorphic systems or other event-driven technologies developed in Lava.

We develop a wrapper process, known as the *Spiking GNN Runtime*, to integrate Lava BO with our spiking graph neural network. As shown in Figure 4, this process receives a vector of hyperparameters from Lava BO, configures the GNN accordingly, and evaluates its performance across the entire validation set. Once the accuracy is determined, both the parameters and the resulting accuracy value

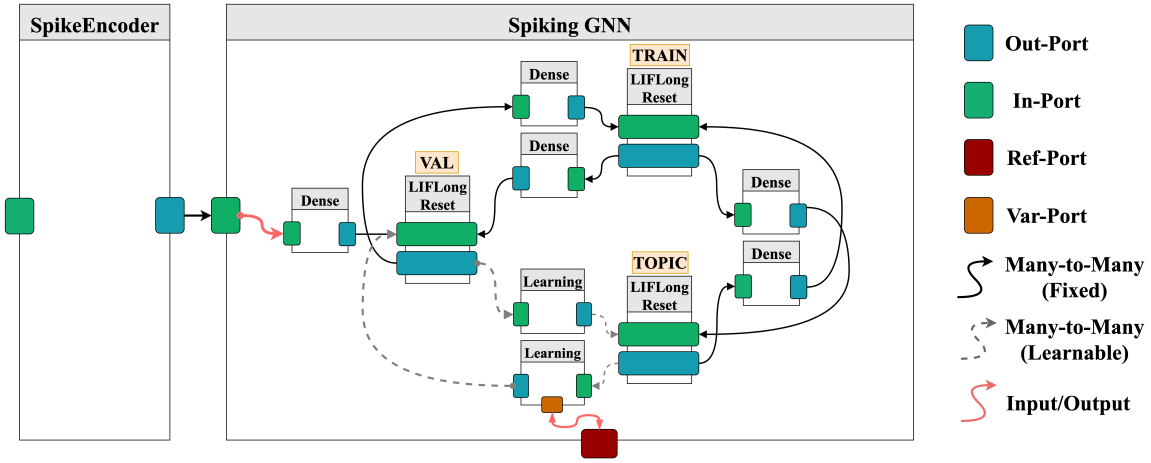

Figure 3: The multi-process and subprocess model architecture representing our Spiking GNN approach for citation network classification in Lava. For simplicity, connections within the validation and testing neurons clusters along with the entire testing cluster have been omitted. *Ref and Var Ports* allow for read and write operations between interconnected processes.

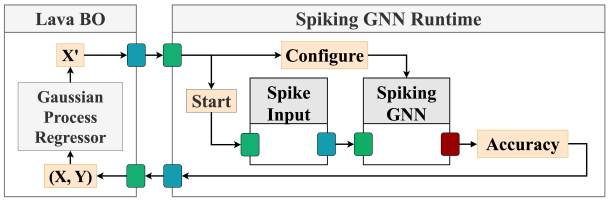

Figure 4: The runtime control flow provided by the Spiking GNN Runtime process whenever it receives new parameters from LavaBO [20]. For more details on the specifics of the Spiking GNN, see Figure 3.

are returned to Lava BO [20]. This iterative process continues until a predefined maximum number of iterations is reached.

3 RESULTS & DISCUSSION

With our fixed-precision spiking graph neural in Lava, we perform citation network classification with the Cora dataset [17]. We compare our approach against a floating-point implementation developed in NEST [4, 11]. This comparison highlights the performance differences between fixed-precision, Loihi 2 [8] compatible architectures compared against traditional floating point implementations that lack native support for neuromorphic hardware. Without this support, networks developed in NEST cannot be easily validated and designed for edge applications where neuromorphic computing systems show great promise over traditional artificial neural networks.

On the other hand, Lava is designed as a prototyping tool for deployment to heterogeneous, event-based architectures such as Loihi 2. The Lava runtime environment executes multiple asynchronous processes with channel-based communication, providing a fine-grained approach to building asynchronous systems. This event-based communication is ideal for scenarios where independent processes need to exchange data without blocking. This framework is developed in Python, which hurdles in fully exploiting the potential for concurrent processing. Python's interpreter restricts multiple processes from utilizing different CPU cores simultaneously. However, this parallelism and multi-processing are key in the deployment of neuromorphic cores.

Table 1: Individual parameters for our neurons, synapses, STDP learning rules, and simulation environment.

Parameter	Value
Paper to Paper w	100
Paper to Topic w	1
Val. to Topic τ_-	30
Topic Leak	Off
Paper Leak	Off
Val. / Test. Refractory	On
Train/Topic Refractory	Off
Time Steps	20

To evaluate the performance of Lava against NEST, we simulate both spiking graph implementations using the same parameters shown in Table 1. All of paper and topic neurons have leak disabled. The initial paper-to-paper and train-to-topic weights are 100 and 1, respectively. The NEST implementation assumes $\tau_+ = \tau_- = 30$ so we configured our Lava implementation to have the same specification. The initial weights for the learnable synapses (test and validation to topic) vary from one implementation to another. In NEST, the initial weight is 0.0001 with floating-point integer precision. Since our implementation in Lava operates using fixed-point integer precision, we initialized the weights to 0 for our experiments to provide a fair comparison between the two simulators. Following the implementation proposed in [6], the training and topic neurons are configured to have no refractory period while

the testing and validation neurons have a refractory period greater than the number of simulation steps. By imposing a high refractory period on validation and testing neurons, we create temporal boundaries that limit their firing frequency where leakage does not affect the behaviour. Meanwhile, the topic and training neurons operate without leakage or refractory periods to encourage continuous, reinforcing spiking activity within the learned synaptic connections.

Similar to our method, the spiking graph in NEST infers the predicted topic as the topic neuron with the lowest reverse synaptic connection between the given paper and topics.

Table 2: Comparing the validation classification accuracy and time per simulation step of NEST and Lava across varying time steps and delay periods.

Simulator	**Lava** [15]		**NEST** [11]	
Delay Period	0	1	0	1
$t = 8$	55.57%	28.57%	10.71%	70.71%
$t = 14$	**63.57**%	57.85%	12.14%	**68.57**%
$t = 20$	36.43%	**59.28**%	12.14%	**59.28**%
$\Delta t(t_i)$	9.4ms	83.5ms	1.4ms	1.6ms

Unless otherwise noted, all experiments are conducted with an Intel(R) Xeon(R) W-2295 CPU and 128GB of system memory. Table 2 shows a quantitative comparison between Lava and NEST. The ideal configuration for our system, achieving 63.57% accuracy on the validation set, used 14 simulation steps per paper with a delay of 0. The original configuration from [6] achieved 68.57% accuracy with the same number of time steps and a delay period of 1. Reducing the delay period below 1 results in drastic performance reductions within the NEST implementation. Based on these results, our Lava implementation appears to exhibit greater resilience and robustness when delay values are modified compared to NEST.

NEST requires 1.6 milliseconds per simulation time step compared to Lava with 9.4 milliseconds. Despite NEST having a lower per-timestep latency, the entire graph has to be reinitialized for every paper, so that the network is reset to the initial inactive state. With this operation taking 9.73 seconds and being repeated for each of the 140 neurons in the validation set, it drastically increases the overall runtime.

In our Lava implementation, we reset the system by allowing it to discharge for 7 timesteps (i.e. the reset period) and re-initialize learnable weights to their initial values. Because of our custom neuron model, our network is only initialized once with a latency of 0.16 seconds and the reset process per paper takes 123.6 milliseconds.

It should also be noted that NEST has been heavily optimized for von-Neumann processors, whereas Lava is designed to be an abstract tool for prototyping biologically inspired systems before deployment to true neuromorphic hardware.

We use Bayesian optimization, implemented in Lava Bayesian Optimization (Lava BO) [20], to learn the optimal hyperparameter combination that maximizes classification accuracy with our spiking graph neural network. Our search space, as shown in Table 3, consists 2450 unique parameter combinations across four individual parameters: paper to paper weight, topic to paper weight, τ_+ and τ_-, along with the number of simulation time steps. We provide Lava BO with 5 initial random points and allow it 10 iterations, for a total of 15 optimization iterations, to learn from this prior knowledge and intelligently select which individual parameter combinations could have provide the greatest increases in accuracy. This experiment concluded with a final parameter configuration providing performance increases of 3.58% over the original NEST parameters for a total classification accuracy of 62.86%. While this performance is not better than the optimized floating-point implementation, these results show that fixed precision spiking neural networks can be optimized with Bayesian optimization implemented in Lava BO [20].

Table 3: The parameter search space for optimizing our fixed-point spiking graph neural network with Lava BO [20]. The overall space contains 2450 parameters.

Parameter	***Options***	***Best***
Paper to Paper Weight	{1, 50, 100, 300, 500, 700, 1000}	300
Train to Topic Weight	{1, 50, 100, 300, 500, 700, 1000}	1
Val. to Topic τ_+ & τ_-	{20, 25, 30, 35, 40}	30
Simulation Steps	{11, 12, ..., 20}	12

4 CONCLUSION

In this work, we introduce a Loihi 2 compatible, fixed-precision spiking graph neural network for citation graph classification in Lava. Comparing against previous results from [4, 6], our method achieves comparable accuracy while limiting inter-network dynamics to integer precision values. While the per-timestep simulation latency of our network is 5.31x slower compared with the highly optimized CPU implementation in Nest [11], our implementation provides native support for deployment to event-driven neuromorphic processors such as Loihi 2. Our spiking graph doesn't need to be fully reinitialized on a paper-by-paper basis, saving 9.73 seconds per paper evaluation. Moreover, we create a novel runtime process wrapper compatible with Lava Bayesian Optimization (Lava BO) [20]. Using Lava BO, we were able to increase accuracy by 3.58% over the original configuration from [6] to a final value of 62.86%. While these results are not better than the optimized floating point implementation, they highlight the capability of fixed-precision, Loihi-compatible spiking neural networks to be optimized with Bayesian optimization.

In future works, we will deploy this architecture on physical Loihi 2 hardware. Furthermore, we will use the improved runtime efficiency of our spiking spiking graph to improve hyperparameter optimization speed by running multiple instances of the spiking graph communicating with a single instance of Lava BO. Lastly, we will experiment with different datasets and compare the latency, throughput, and power efficiency versus existing spiking neural network and graph convolution neural network implementations.

ACKNOWLEDGMENTS

The work in this paper is supported by a gift from Intel Corporation.

REFERENCES

[1] Sami Abu-El-Haija, Bryan Perozzi, Rami Al-Rfou, and Alex Alemi. 2017. Watch Your Step: Learning Graph Embeddings Through Attention. *CoRR* abs/1710.09599 (2017). arXiv:1710.09599 http://arxiv.org/abs/1710.09599

[2] James Bradley Aimone, Yang Ho, Ojas D. Parekh, Cynthia A. Phillips, Ali Pınar, William M. Severa, and Yipu Wang. 2020. Provable Neuromorphic Advantages for Computing Shortest Paths. *Proceedings of the 32nd ACM Symposium on Parallelism in Algorithms and Architectures* (2020). https://api.semanticscholar.org/CorpusID:220434243

[3] Filipp Akopyan, Jun Sawada, Andrew Cassidy, Rodrigo Alvarez-Icaza, John Arthur, Paul Merolla, Nabil Imam, Yutaka Nakamura, Pallab Datta, Gi-Joon Nam, Brian Taba, Michael Beakes, Bernard Brezzo, Jente B. Kuang, Rajit Manohar, William P. Risk, Bryan Jackson, and Dharmendra S. Modha. 2015. TrueNorth: Design and Tool Flow of a 65 mW 1 Million Neuron Programmable Neurosynaptic Chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34, 10 (2015), 1537–1557. https://doi.org/10.1109/TCAD.2015.2474396

[4] Guojing Cong, Shruti Kulkarni, Seung–Hwan Lim, Prasanna Date, Shay Snyder, Maryam Parsa, Dominic Kennedy, and Catherine Schuman. 2023. 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://doi.org/10.1109/ICMLA58977.2023.00232

[5] Guojing Cong and Seung-Hwan Lim. 2021. Versatile feature learning with graph convolutions and graph structures. In *2021 International Conference on Data Mining Workshops (ICDMW)*. 669–677. https://doi.org/10.1109/ICDMW53433.2021.00088

[6] Guojing Cong, Seung-Hwan Lim, Shruti Kulkarni, Prasanna Date, Thomas Potok, Shay Snyder, Maryam Parsa, and Catherine Schuman. 2022. Semi-supervised graph structure learning on neuromorphic computers. In *Proceedings of the International Conference on Neuromorphic Systems 2022*. 1–4.

[7] Guojing Cong, Seung-Hwan Lim, Shruti Kulkarni, Prasanna Date, Thomas Potok, Shay Snyder, Maryam Parsa, and Catherine Schuman. 2022. Semi-Supervised Graph Structure Learning on Neuromorphic Computers. In *Proceedings of the International Conference on Neuromorphic Systems 2022* (Knoxville, TN, USA) *(ICONS '22)*. Association for Computing Machinery, New York, NY, USA, Article 28, 4 pages. https://doi.org/10.1145/3546790.3546821

[8] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, Yuyun Liao, Chit-Kwan Lin, Andrew Lines, Ruokun Liu, Deepak Mathaikutty, Steven McCoy, Arnab Paul, Jonathan Tse, Guruguhanathan Venkataramanan, Yi-Hsin Weng, Andreas Wild, Yoonseok Yang, and Hong Wang. 2018. Loihi: A Neuromorphic Manycore Processor with On-Chip Learning. *IEEE Micro* 38, 1 (2018), 82–99. https://doi.org/10.1109/MM.2018.112130359

[9] Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, and Jie Tang. 2020. Graph Random Neural Network. *CoRR* abs/2005.11079 (2020). arXiv:2005.11079 https://arxiv.org/abs/2005.11079

[10] Wulfram Gerstner, Wulfram Gerstner, Richard Kempter, J. Leo van Hemmen, Hermann Wagner, and Hermann Wagner. 1996. A neuronal learning rule for sub-millisecond temporal coding. *Nature* 383 (1996), 76–78. https://api.semanticscholar.org/CorpusID:4319500

[11] Marc-Oliver Gewaltig and Markus Diesmann. 2007. NEST (NEural Simulation Tool). *Scholarpedia* 2, 4 (2007), 1430.

[12] Maoguo Gong, Hui Zhou, A. K. Qin, Wenfeng Liu, and Zhongying Zhao. 2023. Self-Paced Co-Training of Graph Neural Networks for Semi-Supervised Node Classification. *IEEE Transactions on Neural Networks and Learning Systems* 34, 11 (2023), 9234–9247. https://doi.org/10.1109/TNNLS.2022.3157688

[13] Bill Kay, Prasanna Date, and Catherine D. Schuman. 2020. Neuromorphic Graph Algorithms: Extracting Longest Shortest Paths and Minimum Spanning Trees. *Proceedings of the 2020 Annual Neuro-Inspired Computational Elements Workshop* (2020). https://api.semanticscholar.org/CorpusID:218593937

[14] Thomas N. Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. *CoRR* abs/1609.02907 (2016). arXiv:1609.02907 http://arxiv.org/abs/1609.02907

[15] Intel Labs. 2024. Lava Software Framework. https://lava-nc.org Accessed: 2024-3-16.

[16] Yann LeCun, Yoshua Bengio, et al. 1995. Convolutional networks for images, speech, and time series. (1995).

[17] Andrew Mccallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the Construction of Internet Portals with Machine Learning. *Information Retrieval* 3(2) (11 2000). https://doi.org/10.1023/A:1009953814988

[18] Catherine D. Schuman, Kathleen Hamilton, Tiffany Mintz, Md Musabbir Adnan, Bon Woong Ku, Sung-Kyu Lim, and Garrett S. Rose. 2019. Shortest Path and Neighborhood Subgraph Extraction on a Spiking Memristive Neuromorphic Implementation. In *Proceedings of the 7th Annual Neuro-Inspired Computational Elements Workshop* (Albany, NY, USA) *(NICE '19)*. Association for Computing Machinery, New York, NY, USA, Article 3, 6 pages. https://doi.org/10.1145/3320288.3320290

[19] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective Classification in Network Data. *AI Magazine* 29, 3 (Sep. 2008), 93. https://doi.org/10.1609/aimag.v29i3.2157

[20] Shay Snyder, Sumedh R. Risbud, and Maryam Parsa. 2023. Neuromorphic Bayesian Optimization in Lava. In *Proceedings of the 2023 International Conference on Neuromorphic Systems* (Santa Fe, NM USA) *(ICONS '23)*. Association for Computing Machinery, New York, NY, USA, Article 9, 5 pages. https://doi.org/10.1145/3589737.3605998

[21] Mengying Sun, Sendong Zhao, Coryandar Gilvary, Olivier Elemento, Jiayu Zhou, and Fei Wang. 2019. Graph convolutional networks for computational drug development and discovery. *Briefings in Bioinformatics* 21, 3 (06 2019), 919–935. https://doi.org/10.1093/bib/bbz042 arXiv:https://academic.oup.com/bib/article-pdf/21/3/919/33227266/bbz042.pdf

[22] Tian Xie and Jeffrey C. Grossman. 2018. Crystal Graph Convolutional Neural Networks for an Accurate and Interpretable Prediction of Material Properties. *Physical Review Letters* 120, 14 (April 2018). https://doi.org/10.1103/physrevlett.120.145301

[23] Liang Yao, Chengsheng Mao, and Yuan Luo. 2018. Graph Convolutional Networks for Text Classification. arXiv:1809.05679 [cs.CL]

[24] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. 2019. Graph convolutional networks: a comprehensive review. *Computational Social Networks* 6, 1 (2019), 1–23.