ICLR
International Conference On Learning Representations

A Graph is Worth 1-bit Spikes: When Graph Contrastive Learning Meets Spiking Neural Networks

Jintang Li, Huizhe Zhang, Liang Chen, Zibin Zheng, Sun Yat-sen University
Ruofan Wu, Baokun Wang, Changhua Meng, Ant Group
Zulun Zhu, Nanyang Technological University

lijt55@mail2.sysu.edu.cn

https://github.com/EdisonLeeeee/SpikeGCL

中山大學 SUN YAT-SEN UNIVERSITY

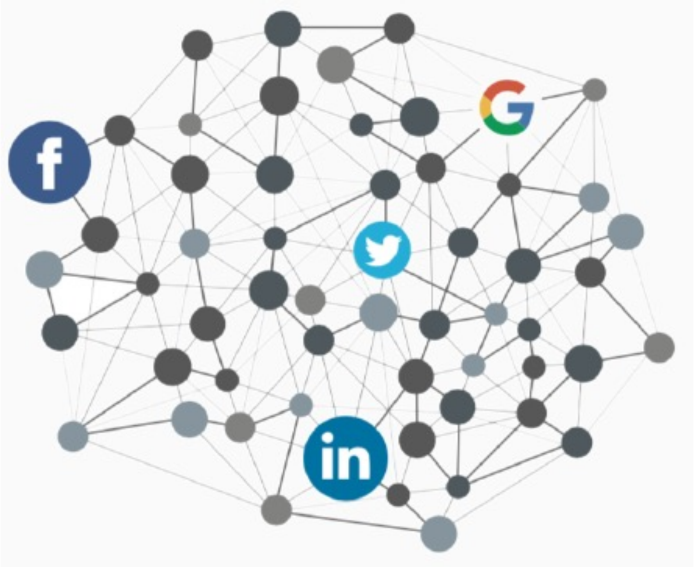
ANT GROUP

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

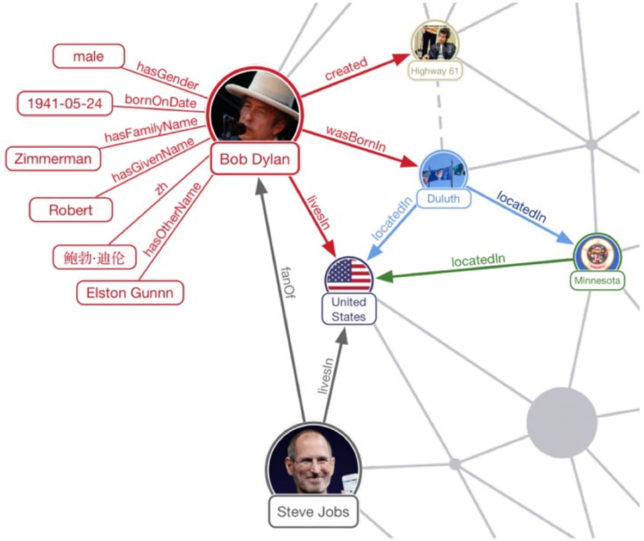
TABLE OF CONTENTS

BACKGROUND Graphs & Graph Neural Networks

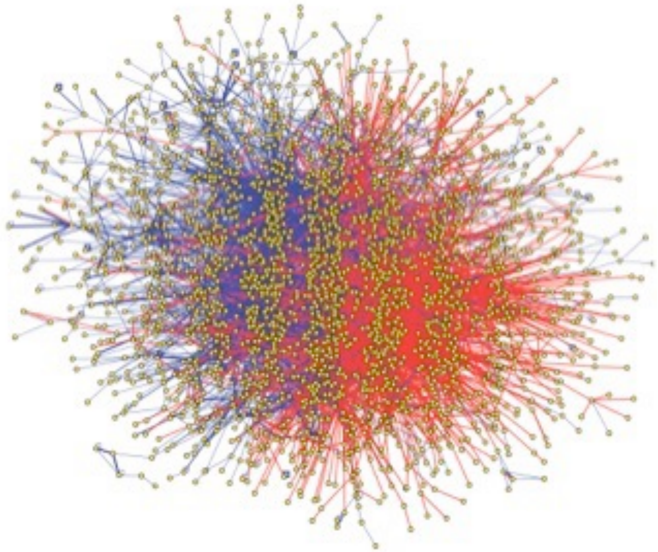
Networks are everywhere - Graphs are natural way to model such networks



Social Networks



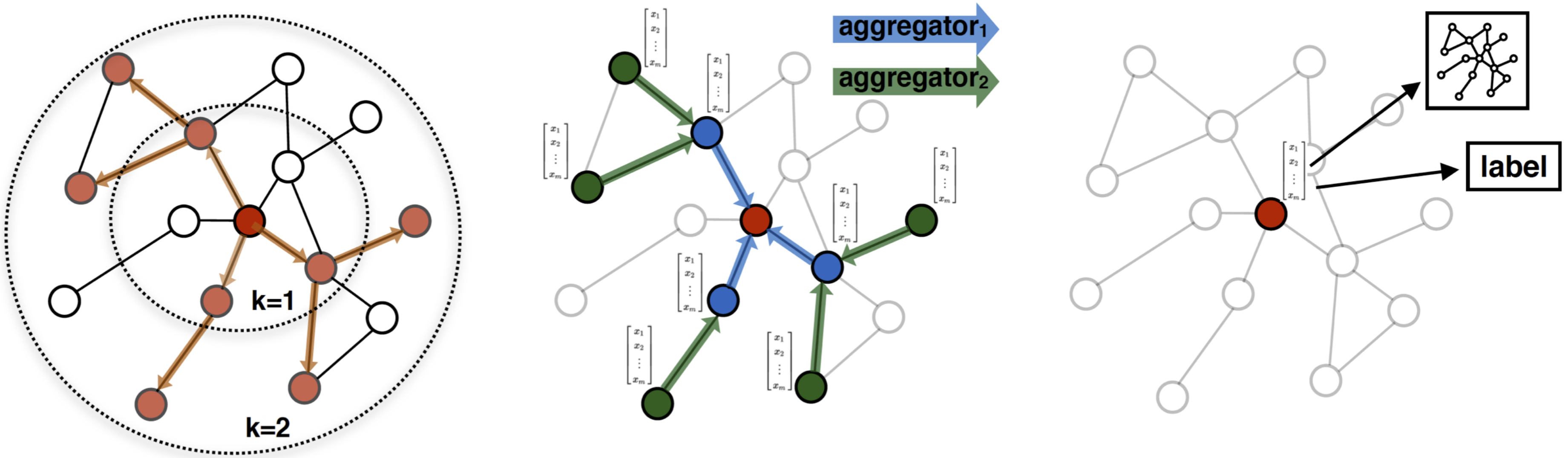
Knowledge Graph



Protein-Protein
Interaction Networks

Graph Neural Networks (GNNs)

- ❑ Message and Aggregation Scheme
- ❑ Compress a set of vectors into a single vector (i.e., node representation)

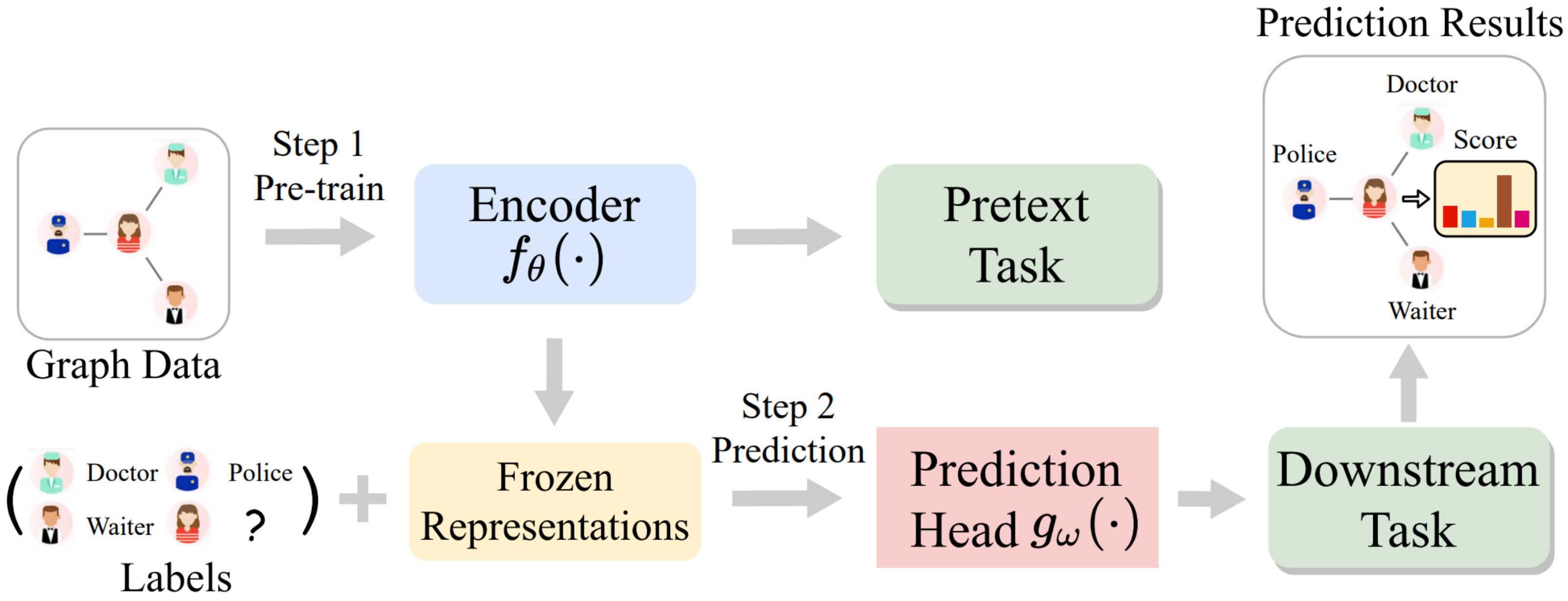


1. Sample neighborhood
2. Aggregate feature information from neighbors
3. Predict graph context and label using aggregated information

Hamilton, Will, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." Advances in neural information processing systems 30 (2017)

Graph Self-supervised Learning

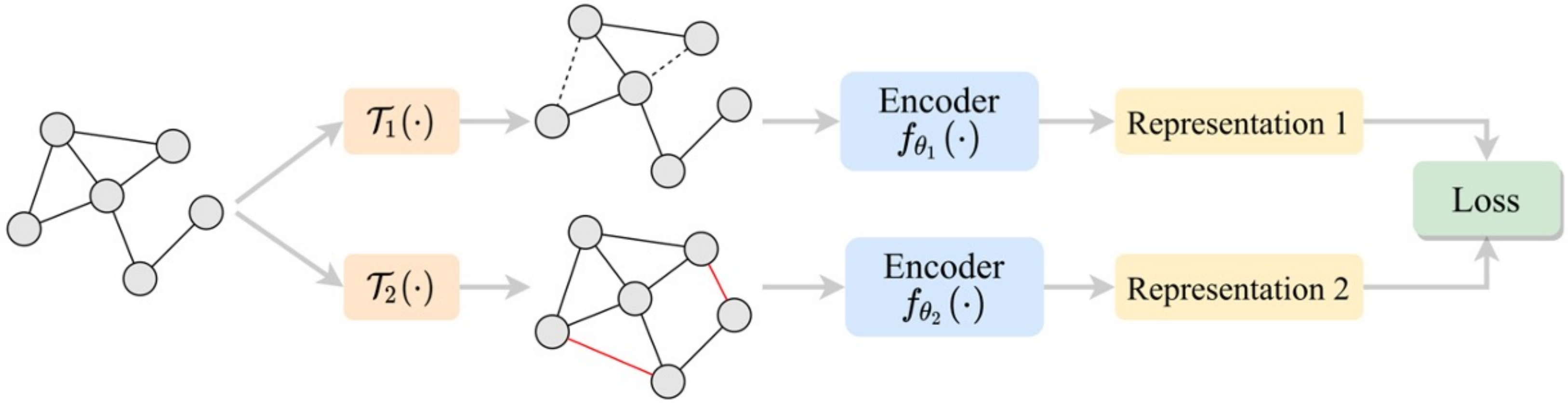
- ❑ GNN → training with self-defined pretext task → encoder f_θ
- ❑ Encoder f_θ → representations → generalize to other downstream tasks



Lirong, Wu et al. "Self-supervised Learning on Graphs: Contrastive, Generative, or Predictive." IEEE Trans. on Knowl. and Data Eng. 35, 4 (April 2023), 4216–4235.

Graph Contrastive Learning

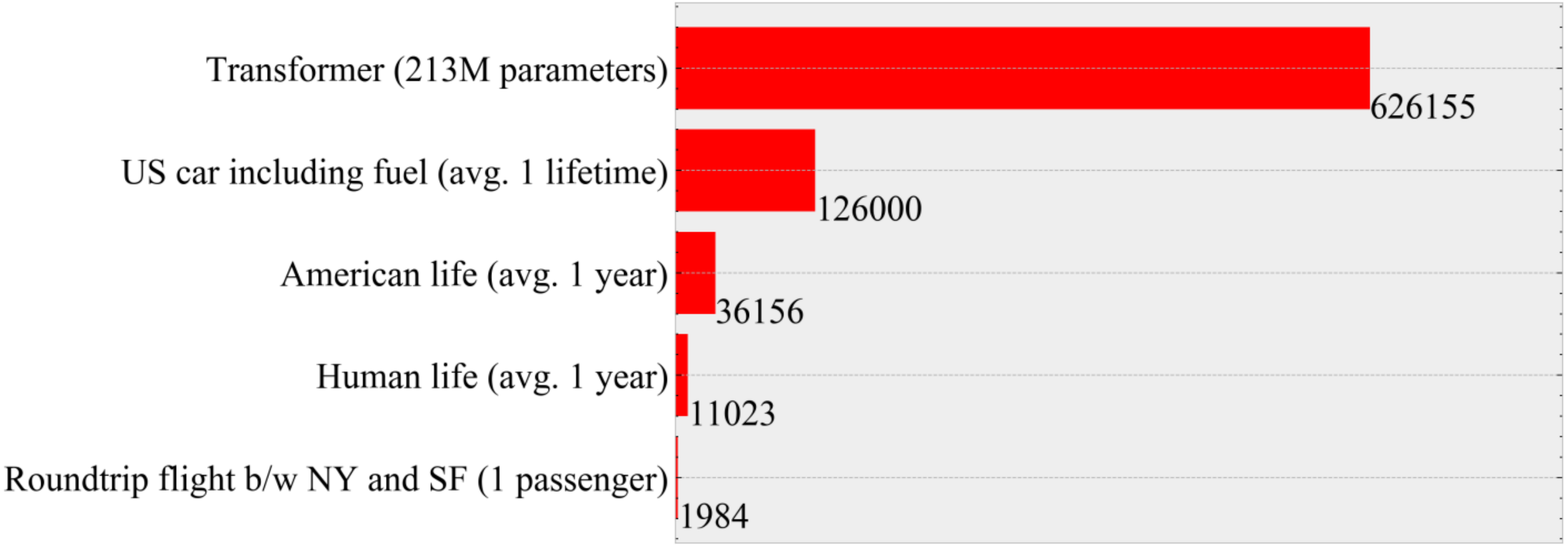
- ❑ Augmentation: graph data augmentation for different graph views
- ❑ Contrasting: maximize the agreement of two graph views
- ❑ Augmentation → Encoder f_θ → representations → Contrasting



Lirong, Wu et al. "Self-supervised Learning on Graphs: Contrastive, Generative, or Predictive." IEEE Trans. on Knowl. and Data Eng. 35, 4 (April 2023), 4216–4235.

Green AI: How Far are We?

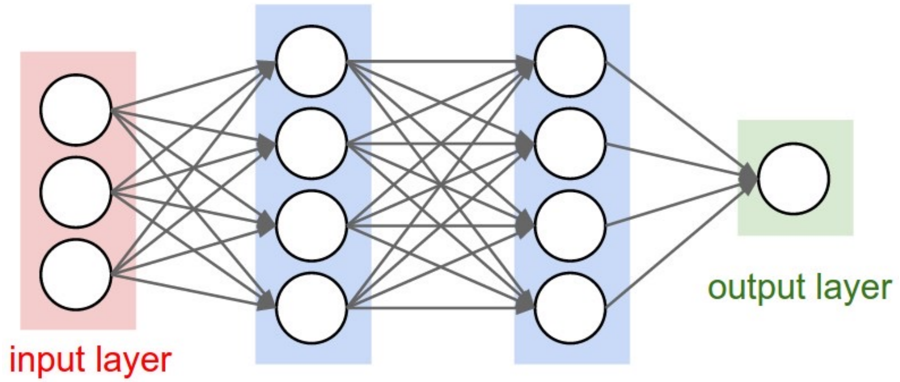
- ❑ CO2 emissions: A Transformers with 213M parameters ≈ 60x Human life
- ❑ Next-generation efficient neural networks -> spiking neural networks (SNN)



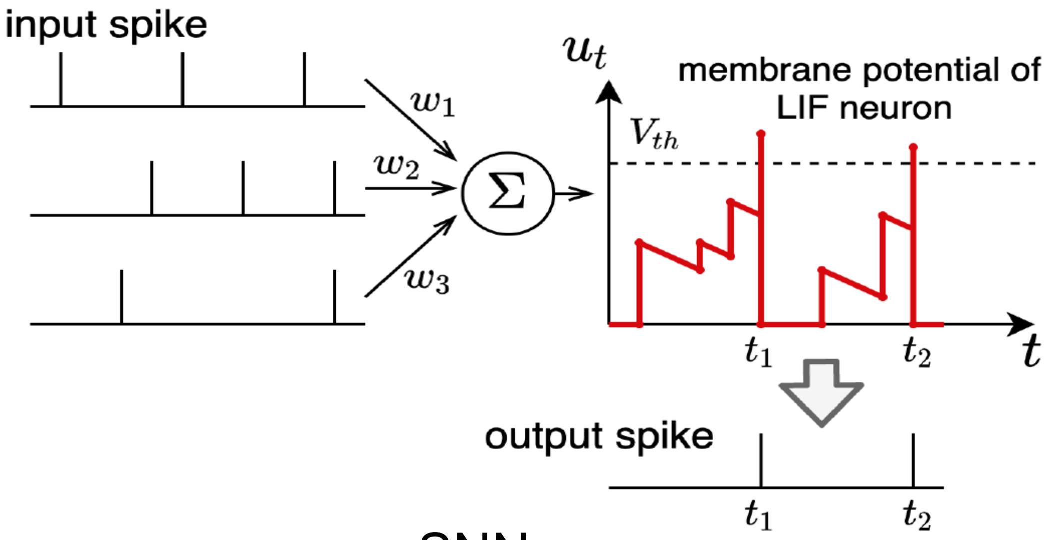
Strubell E, Ganesh A, McCallum A. Energy and policy considerations for deep learning in NLP[J]. arXiv preprint arXiv:1906.02243, 2019.

Spiking Neural Networks

- ❑ ANN (Artificial Neural Network) → Information is typically represented as continuous values in the activations of neurons and the weights of connections 0.4
- ❑ SNN (Spiking Neural Network) → Information is represented as discrete spikes of activity over time. The timing and frequency of spikes encode information in an SNN 0, 0, 1, 1, 0 → 0.4



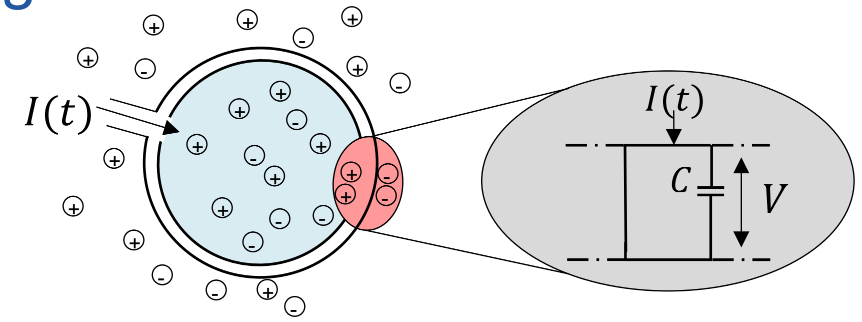
ANN



SNN

Spiking Neurons as Capacitor

- ❑ Spiking neurons act as a capacitor that stores charge over time
- ❑ It assumes ideally that there was no static power leakage



charge capacitance

$$Q = VC$$

voltage

$$\frac{dQ}{dt} = \frac{dV}{dt}C$$

current

$$I(t) = \frac{dV}{dt}C$$

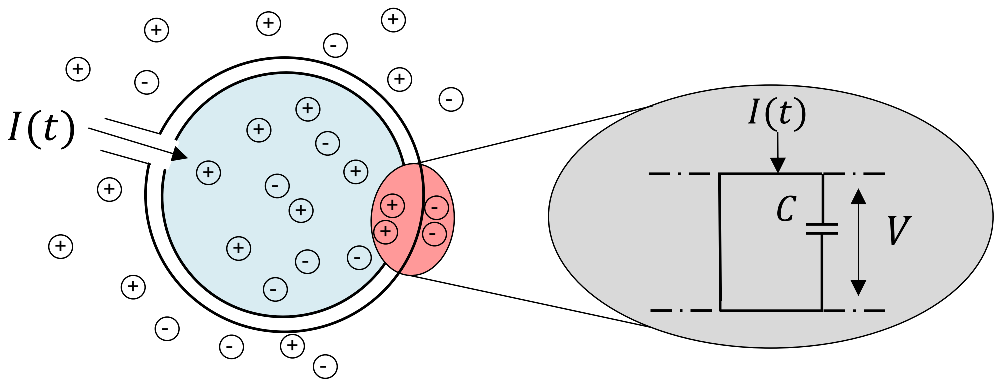
Spiking Neurons

- ❑ Integrate-and-fire (IF): the most simple and common spiking neuron

$$I(t) = \frac{dV}{dt}C$$

Pre-synaptic input

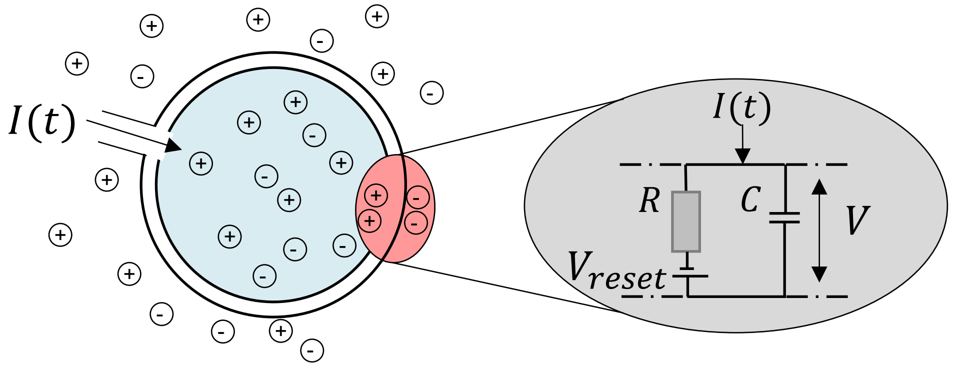
$$\frac{dV}{dt} = \boxed{I(t)}$$



Integrate-and-fire (IF) Neuron

Spiking Neurons

- ❑ Integrate-and-fire (IF): the most simple and common spiking neuron
- ❑ Leaky Integrate-and-fire (LIF): an additional leaky term for which membrane potential charges and discharges exponentially
- ❑ Parametric Leaky Integrate-and-fire (PLIF): the leaky term is trainable



Membrane voltage

Pre-synaptic input

$$\boxed{\tau_m}\frac{dV}{dt} = -(\boxed{V} - \boxed{V_{reset}}) + \boxed{I(t)}$$

Leaky term

Reset level

Leaky integrate-and-fire (LIF) Neuron

How SNNs Work?

- Fundamental characteristics: Integrate, Fire, and Reset
- Different spiking neurons have different integrate behaviors
- The reset mechanism can also be different

1. Integrate

$$V^t = V^{t-1} + I^t$$ IF

$$V^t = V^{t-1} + \frac{1}{\tau_m}(I^t - (V^{t-1} - V_{\text{reset}}))$$ LIF or PLIF

2. Fire (output)

$$S^t = \begin{cases} 1, & V^t \geq \boxed{V_{\text{th}}} \\ 0, & V^t < V_{\text{th}} \end{cases}$$ → threshold

3. Reset

$$V^t = \begin{cases} V^t, & S^t = 0 \\ V_{\text{reset}}, & S^t = 1 \end{cases}$$ Reset to zero

$$V^t = \begin{cases} V^t, & S^t = 0 \\ V^t - V_{\text{th}}, & S^t = 1 \end{cases}$$ Reset by subtraction

How SNNs Work?

- ❑ An simple illustration of the charging behavior in each spiking neuron

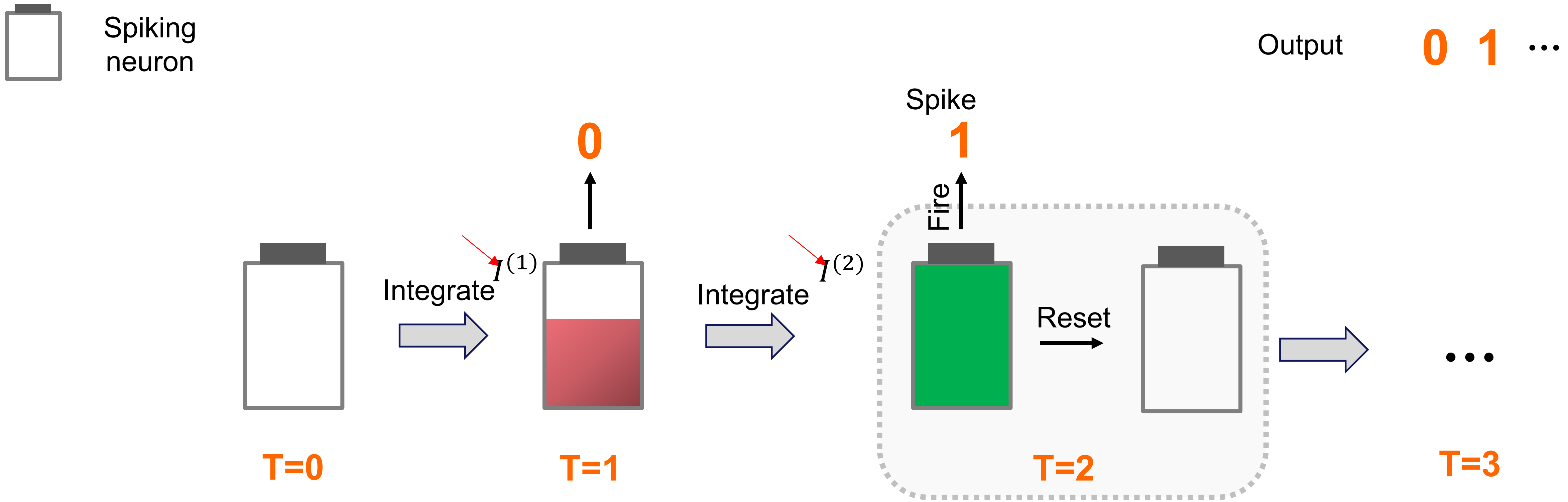
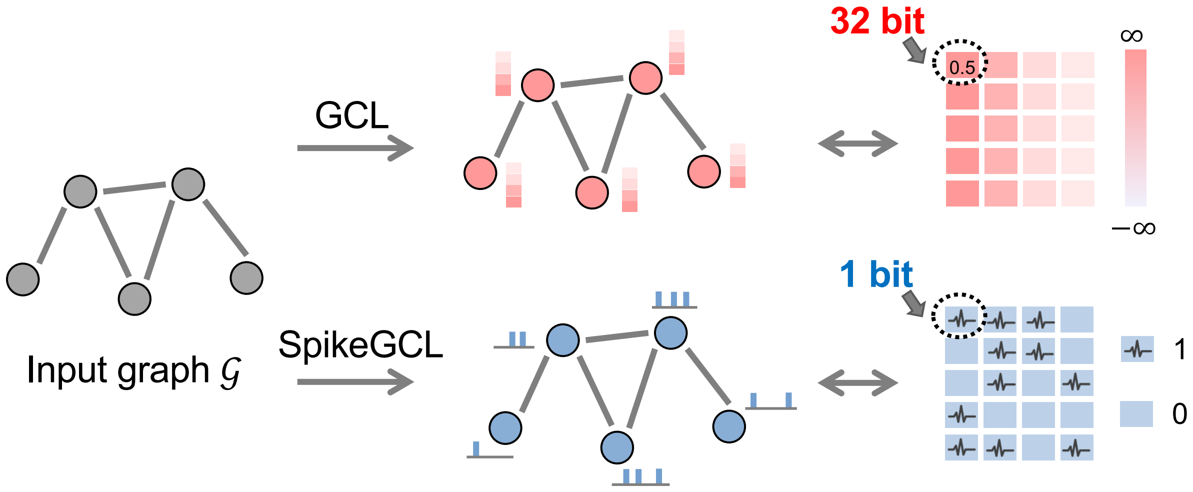


TABLE OF CONTENTS

MOTIVATION

🤔 Can we explore the possibilities of SNNs with contrastive learning schemes to learn sparse, binarized yet generalizable representations?



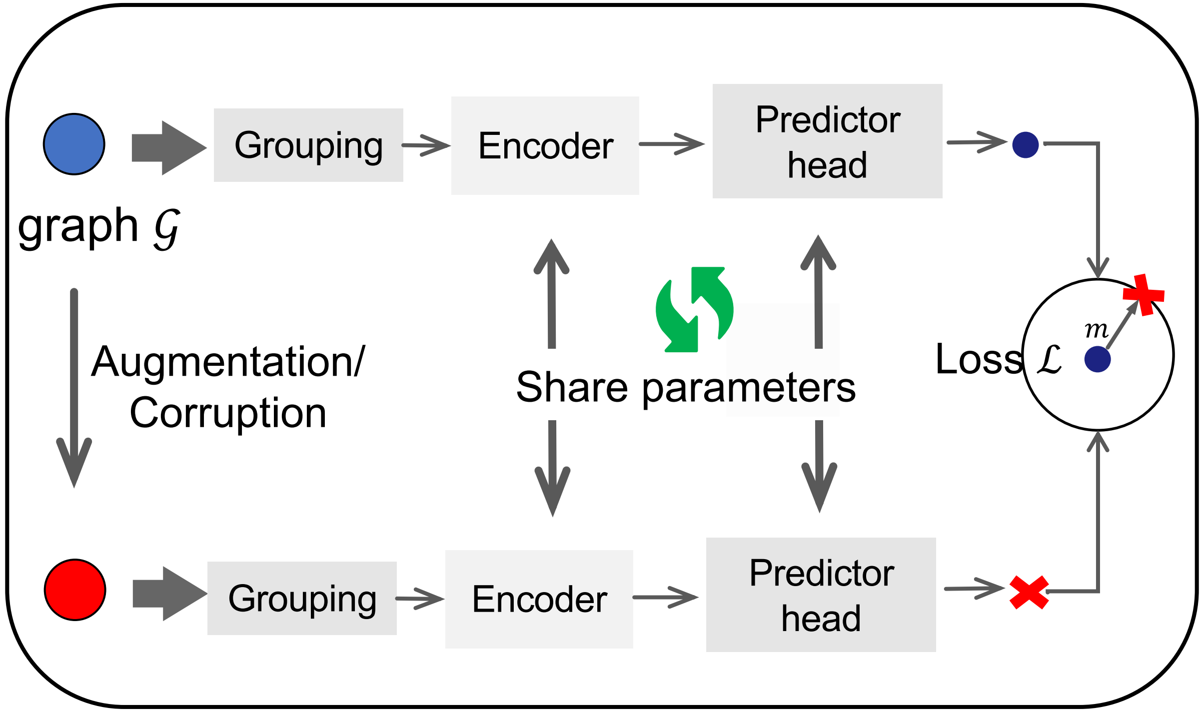
METHOD Spiking Graph Contrastive Learning (**SpikeGCL**)

Incorporating spiking neural networks to graph contrastive learning -> SpikeGCL

SNN + GCL = SpikeGCL

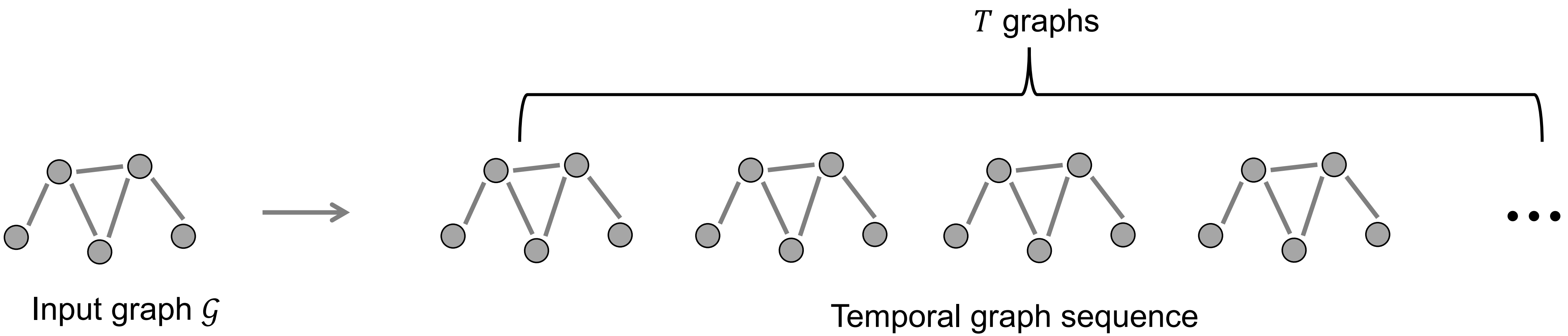
Overall Framework

- ❑ Step1: Augmentation/corruption
- ❑ Step2: Grouping + Encoding
- ❑ Step3: Decoding
- ❑ Step4: Contrasting



Why do we need Grouping? 🤔

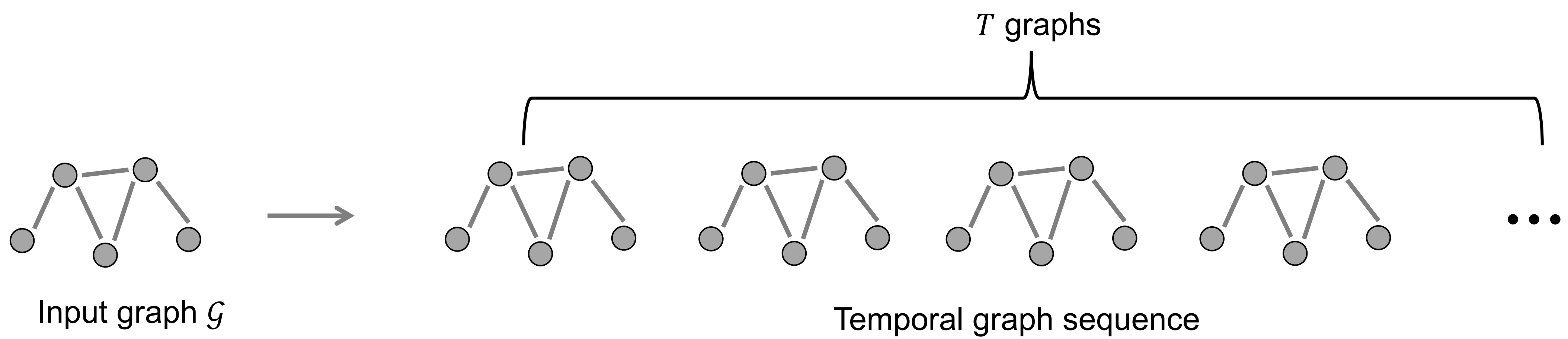
- SNNs are temporal models that require sequential inputs
- Challenge: how to formulate such inputs from a non-temporal graph?
- Existing works: repeat input graph multiple times w/ or w/o augmentations (SpikingGCN, GC-SNN, GA-SNN, …)



METHOD Spiking Graph Contrastive Learning (**SpikeGCL**)

Why do we need Grouping? 🤔

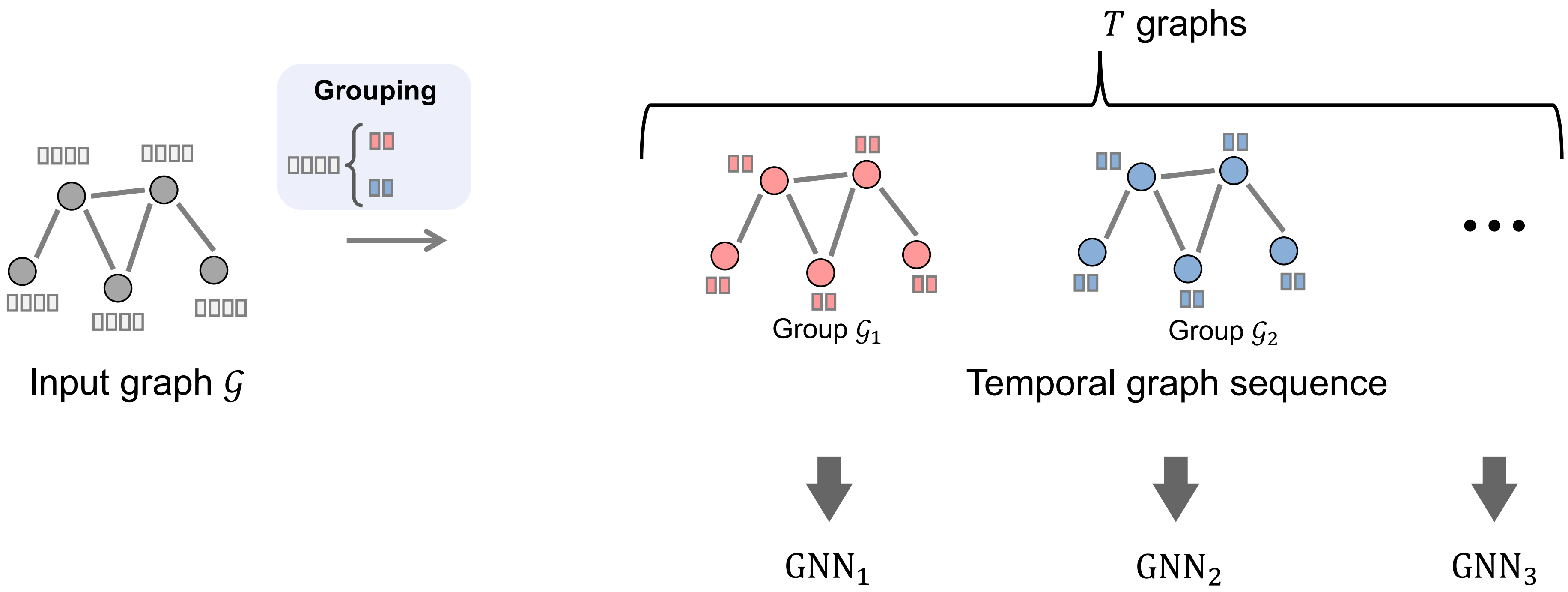
- ❑ SNNs are temporal models that require sequential inputs
- ❑ Challenge: how to formulate such inputs from a non-temporal graph?
- ❑ Existing works: repeat input graph multiple times w/ or w/o augmentations (SpikingGCN, GC-SNN, GA-SNN, …)



High computational and memory overhead, almost T times that of GNNs

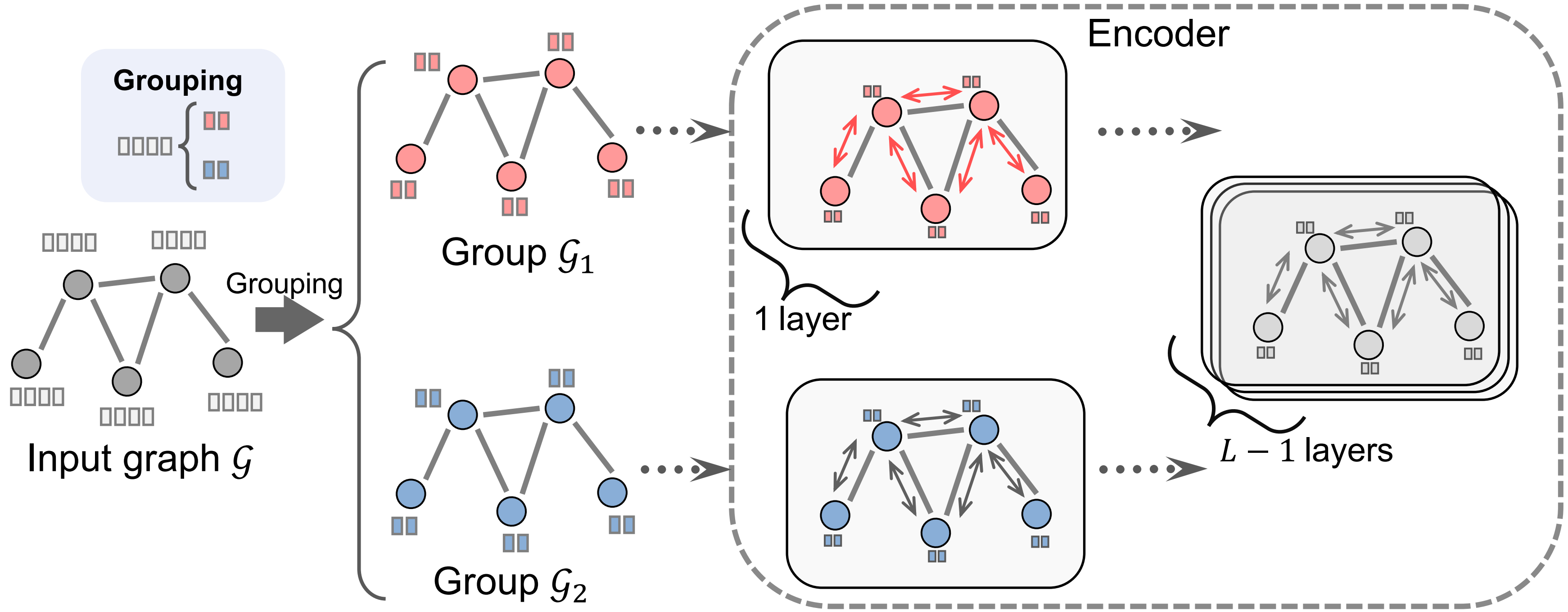
Why do we need Grouping? 🤔

- ❑ Partition the node features into T groups, rather than repeating the graphs
- ❑ Each group of features is paired with a GNN network



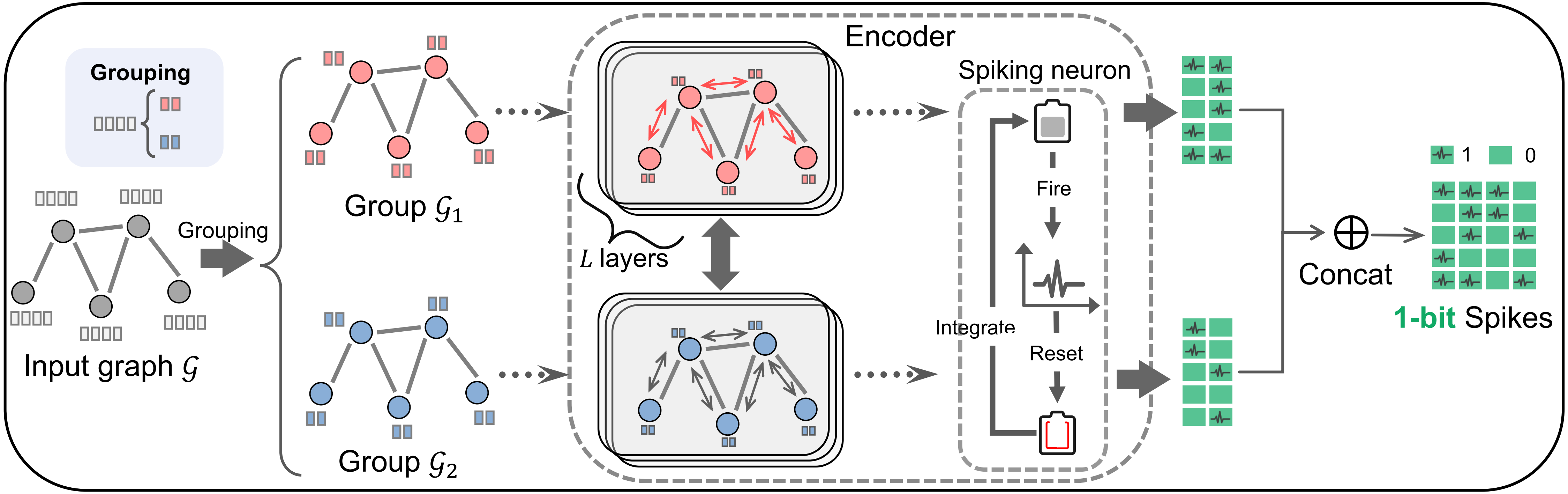
SpikeGCL encoding

- ❑ Only the first layer is different in response to diverse groups of features
- ❑ Parameters of the remaining L-1 layers of the peer GNNs are shared together
- ❑ The overall complexity is almost equivalent to a regular GNN



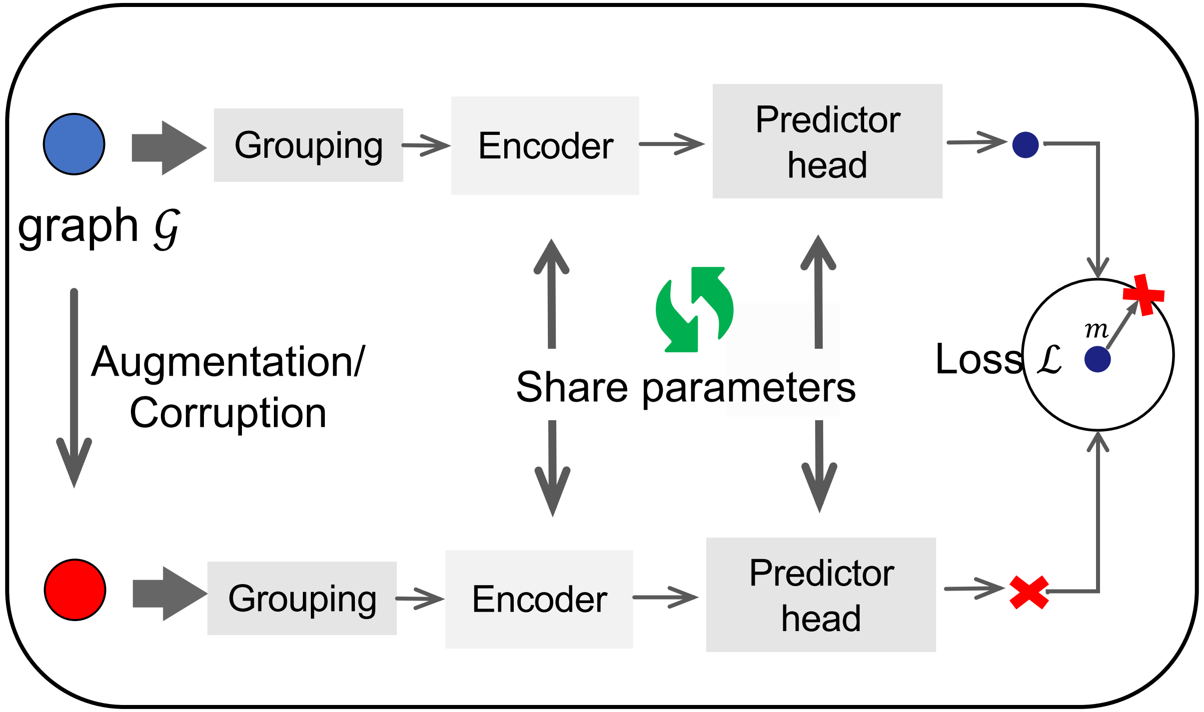
SpikeGCL encoding

- ❑ A spiking neuron is used to output spikes
- ❑ Grouping -> Encoding -> Spiking



Overall Framework

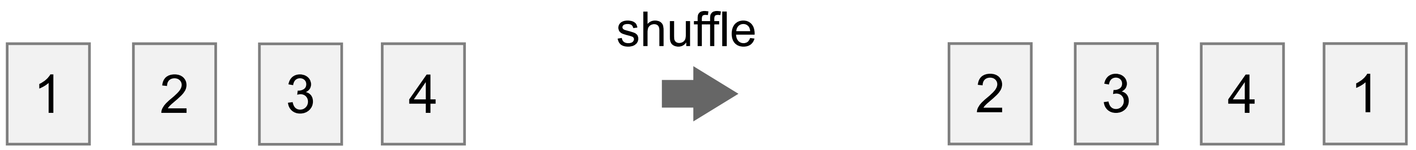
- ❑ Step1: Augmentation/corruption
- ❑ Step2: Grouping + Encoding
- ❑ Step3: Decoding
- ❑ Step4: Contrasting



METHOD Spiking Graph Contrastive Learning (**SpikeGCL**)

Augmentation

- ❑ Edge dropping & feature shuffling
- ❑ Edge dropping: randomly drops a subset of edges from the original graph
- ❑ Feature shuffling: gives a new permutation of the node feature matrix along the feature dimension



Decoding + Contrasting

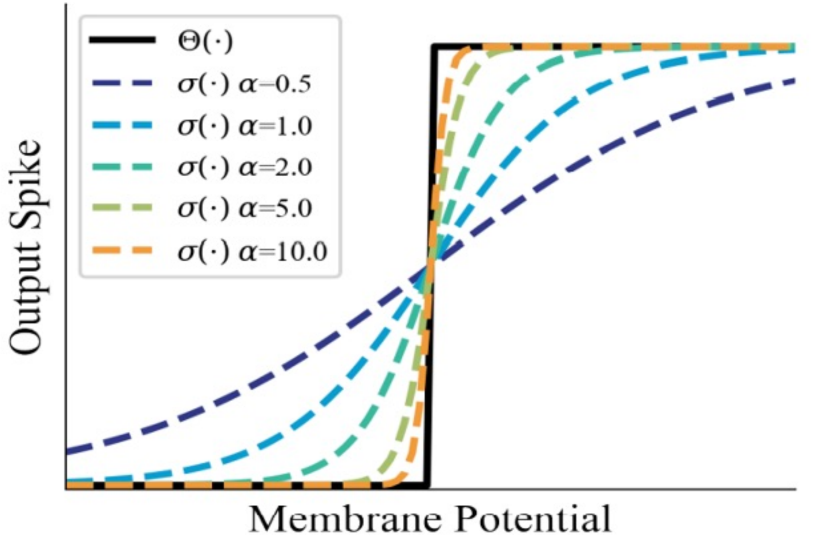
- An MLP g_φ is used as the predictor to decode the binary representations to continuous ones
- Contrastive learning objective: margin ranking loss

$$\mathcal{J} = \frac{1}{N}\sum \max\left(0, \overbrace{\boxed{g_\varphi}}^{\text{MLP predictor}}(\underbrace{\boxed{z_u}}_{\text{Spikes (positive)}}) - g_\varphi(\underbrace{\boxed{\bar{z_u}}}_{\text{Spikes (negative)}}) + \overbrace{\boxed{m}}^{\text{Margin}}\right)$$

Blockwise training

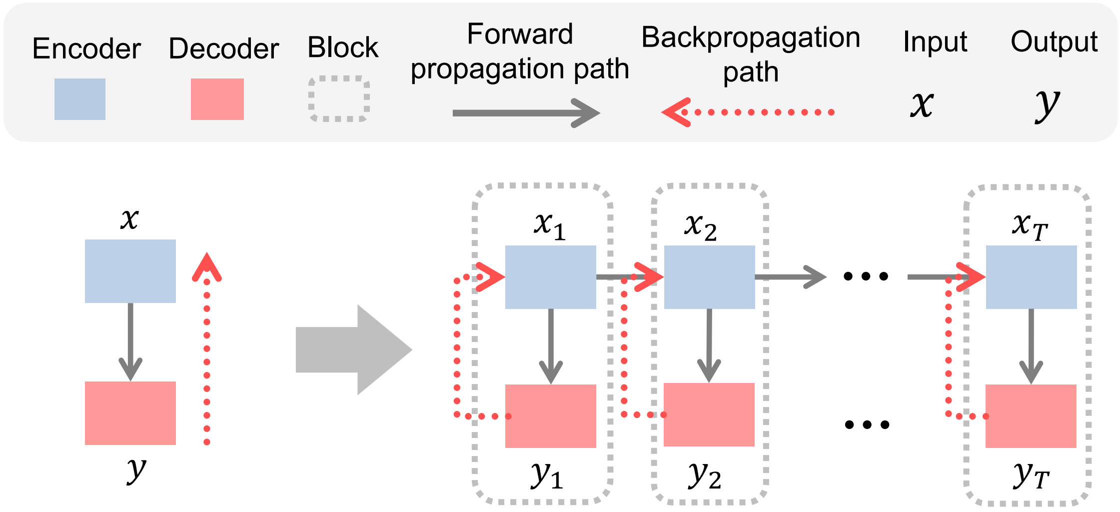
- Directly training SNNs using gradient descent is non-trivial
- Surrogate gradient learning is a viable solution, but it also has limitations
- Surrogate functions -> vanishing gradient
- SNNs requires relatively large time steps -> high overheads

$$\Theta(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \quad \approx \quad \sigma(\alpha x) = 1/\big(1 + \exp(-\alpha x)\big)$$



Blockwise training

- ❑ One or more consecutive time steps as a single block, and limit the length of the backpropagation path to each of these blocks
- ❑ Parameters within each block are optimized locally with a contrastive objective



(a) End-to-end backpropagation

(b) Blockwise backpropagation

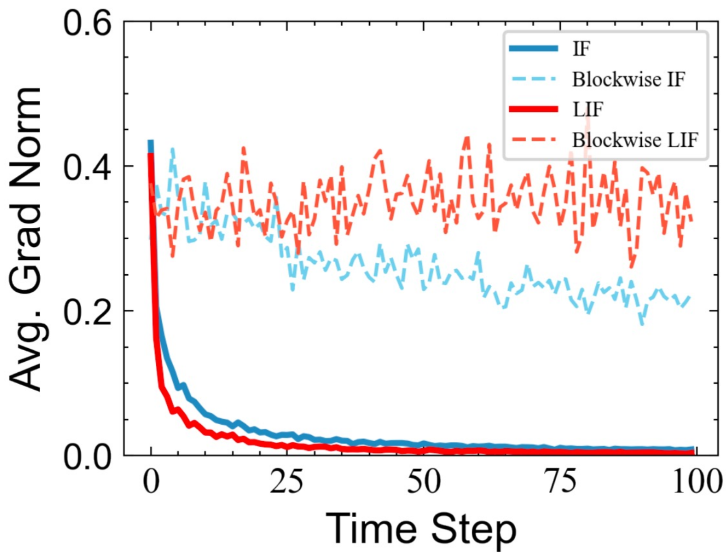


TABLE OF CONTENTS

THEORETICAL INSIGHTS

SpikeGCL is able to approximate a full-precision GNN with a relatively large time step T

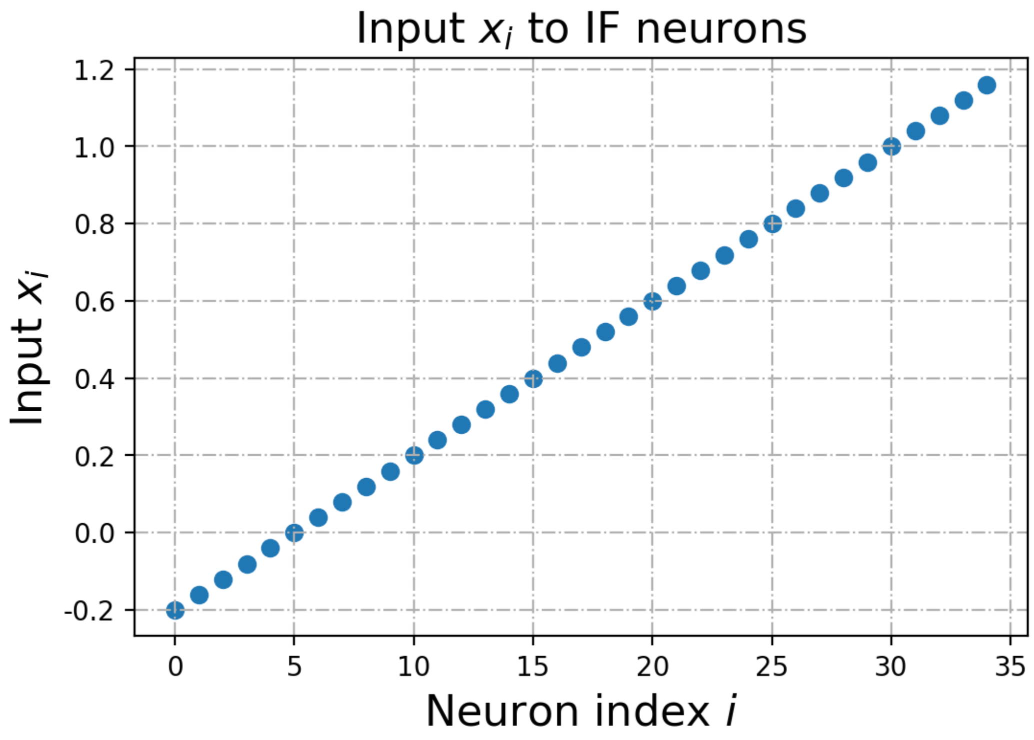
Theorem 1 (Informal). For any full-precision GNN with a hidden dimension of d/T , there exists a corresponding SpikeGCL such that its approximation error, defined as the ℓ_2 distance between the firing rates of the SpikeGCL representation and the GNN representation at any single node, is of the order $\Theta(1/T)$.

Detailed Proof -> Appendix A

A Toy Example: How an IF neuron approximates the ReLU activation

Generate random inputs:

x = torch.arange(-0.2, 1.2, 0.04)

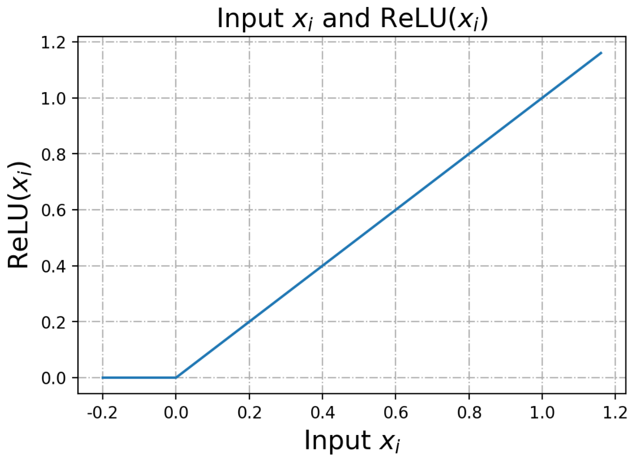


THEORETICAL INSIGHTS

A Toy Example: How an IF neuron approximates the ReLU activation

Output of ReLU:

$$\mathrm{ReLU}(x) = \max(x, 0)$$

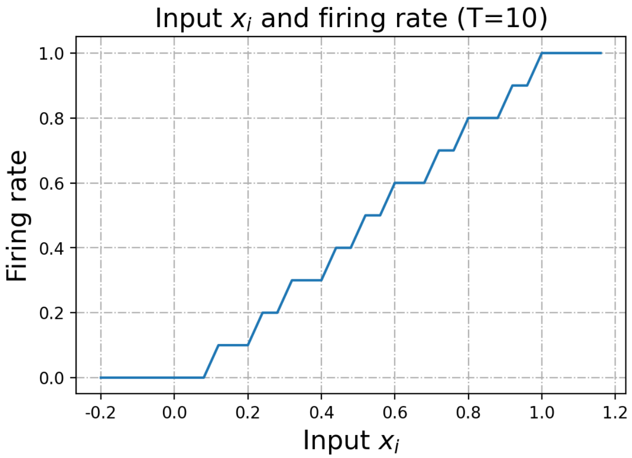


ReLU

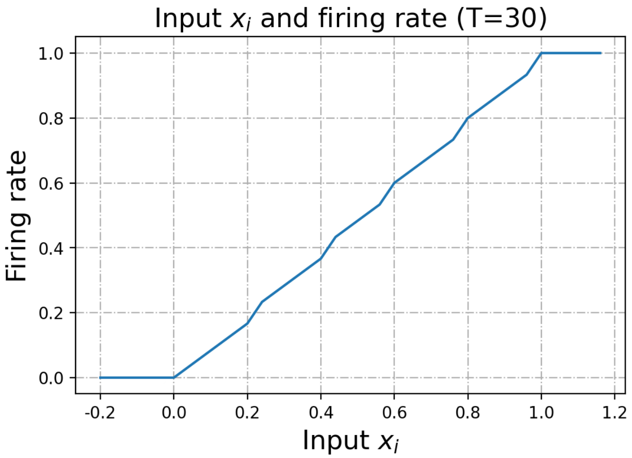
THEORETICAL INSIGHTS

A Toy Example: How an IF neuron approximates the ReLU activation

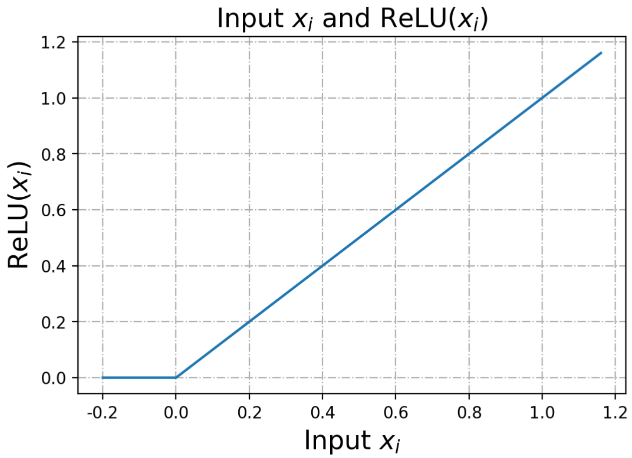
$$\text{Firing rate of IF} = \frac{\#\ \text{spikes}}{t}$$



T=10



T=30



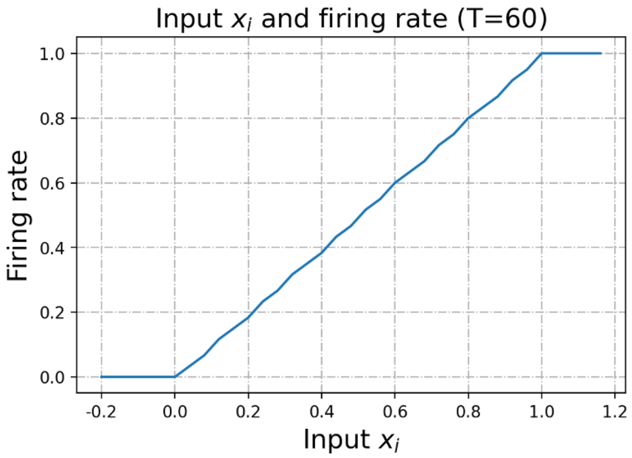
ReLU

THEORETICAL INSIGHTS

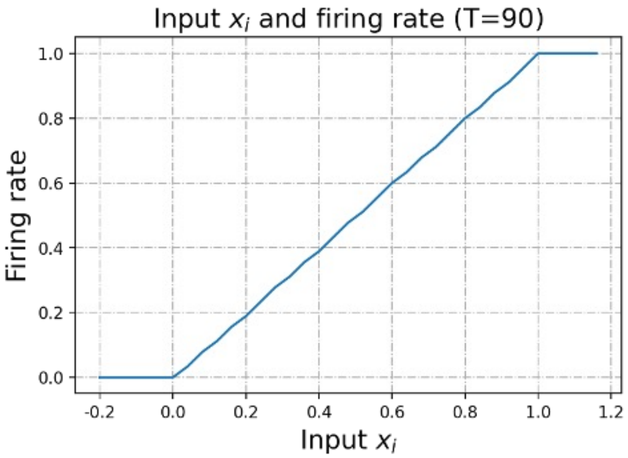
A Toy Example: How an IF neuron approximates the ReLU activation

$$\text{Firing rate of IF} = \frac{\#\ \text{spikes}}{t}$$

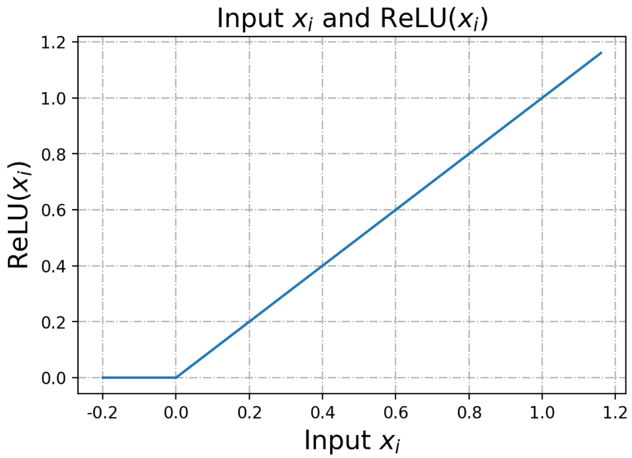
A larger T leads to a better approximation of ReLU



T=60



T=90



ReLU

TABLE OF CONTENTS

Experiment Setup

- 9 real-world graph datasets 2k → 736k scale
- Node classification task with ACC as the metric
- Supervised and self-supervised baselines, including binarized, full-precision, and spike-based methods
- SpikeGCL is a self-supervised, binarized, and spike-based method

	Computers	Photo	CS	Physics	Cora	CiteSeer	PubMed	arXiv	MAG
#Nodes	13,752	7,650	18,333	34,493	2,708	3,327	19,717	16,9343	736,389
#Edges	491,722	238,162	163,788	495,924	10,556	9,104	88,648	2,315,598	10,792,672
#Features	767	745	6,805	8,415	1,433	3,703	500	128	128
#Classes	10	8	15	5	7	6	3	40	349
Density	0.144%	0.082%	0.023%	0.407%	0.260%	0.049%	0.042%	0.008%	0.002%

RESULTS Node Classification

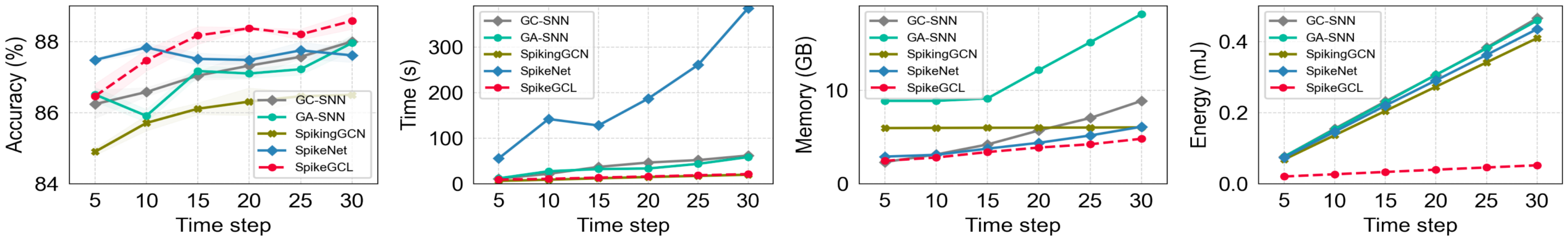
U: unsupervised or self-supervised; S: spike-based; B: binarized

	U	S	B	Computers	Photo	CS	Physics	arXiv	MAG
GCN Kipf & Welling (2016)				$86.5_{\pm 0.5}$	$92.4_{\pm 0.2}$	$92.5_{\pm 0.4}$	$95.7_{\pm 0.5}$	$70.4_{\pm 0.3}$	$30.1_{\pm 0.3}$
GAT Veličković et al. (2018)				$86.9_{\pm 0.2}$	$92.5_{\pm 0.3}$	$92.3_{\pm 0.2}$	$95.4_{\pm 0.3}$	$70.6_{\pm 0.3}$	$30.5_{\pm 0.3}$
SpikeNet Li et al. (2023b)		✓		$88.0_{\pm 0.7}$	$92.9_{\pm 0.1}$	**$93.4_{\pm 0.2}$**	**$95.8_{\pm 0.7}$**	$66.8_{\pm 0.1}$	-
SpikingGCN Zhu et al. (2022)		✓		$86.9_{\pm 0.3}$	$92.6_{\pm 0.7}$	$92.6_{\pm 0.3}$	$94.3_{\pm 0.1}$	$55.8_{\pm 0.7}$	-
GC-SNN Xu et al. (2021)		✓		$88.2_{\pm 0.6}$	$92.8_{\pm 0.1}$	$93.0_{\pm 0.4}$	$95.6_{\pm 0.7}$	-	-
GA-SNN Xu et al. (2021)		✓		$88.1_{\pm 0.1}$	**$93.5_{\pm 0.6}$**	$92.2_{\pm 0.1}$	$95.8_{\pm 0.5}$	-	-
Bi-GCN Wang et al. (2021)			✓	$86.4_{\pm 0.3}$	$92.1_{\pm 0.9}$	$91.0_{\pm 0.7}$	$93.3_{\pm 1.1}$	$66.0_{\pm 0.8}$	$28.2_{\pm 0.4}$
BinaryGNN Bahri et al. (2021)			✓	$87.8_{\pm 0.2}$	$92.4_{\pm 0.2}$	$91.2_{\pm 0.1}$	$95.3_{\pm 0.1}$	$67.2_{\pm 0.9}$	-
BANE Yang et al. (2018)	✓		✓	$72.7_{\pm 0.3}$	$78.2_{\pm 0.3}$	$92.8_{\pm 0.1}$	$93.4_{\pm 0.4}$	>3days	>3days
DGI Velickovic et al. (2019)	✓			$84.0_{\pm 0.5}$	$91.6_{\pm 0.2}$	$92.2_{\pm 0.6}$	$94.5_{\pm 0.5}$	$65.1_{\pm 0.4}$	$31.4_{\pm 0.3}$
GRACE Zhu et al. (2020)	✓			$86.3_{\pm 0.3}$	$92.2_{\pm 0.2}$	$92.9_{\pm 0.0}$	$95.3_{\pm 0.0}$	$68.7_{\pm 0.4}$	$31.5_{\pm 0.3}$
CCA-SSG Zhang et al. (2021)	✓			$88.7_{\pm 0.3}$	$93.1_{\pm 0.1}$	$93.3_{\pm 0.1}$	$95.7_{\pm 0.1}$	$71.2_{\pm 0.2}$	$31.8_{\pm 0.4}$
BGRL Thakoor et al. (2021)	✓			**$90.3_{\pm 0.2}$**	$93.2_{\pm 0.3}$	$93.3_{\pm 0.1}$	$95.7_{\pm 0.0}$	$71.6_{\pm 0.1}$	$31.1_{\pm 0.1}$
SUGRL Mo et al. (2022)	✓			$88.9_{\pm 0.2}$	$93.2_{\pm 0.4}$	$93.4_{\pm 0.0}$	$95.2_{\pm 0.0}$	$68.8_{\pm 0.4}$	**$32.4_{\pm 0.1}$**
GGD Zheng et al. (2022)	✓			$88.0_{\pm 0.1}$	$92.9_{\pm 0.2}$	$93.1_{\pm 0.1}$	$95.3_{\pm 0.0}$	**$71.6_{\pm 0.5}$**	$31.7_{\pm 0.7}$
SpikeGCL	✓	✓	✓	$88.9_{\pm 0.3}$	$93.0_{\pm 0.1}$	$92.8_{\pm 0.1}$	$95.2_{\pm 0.6}$	$70.9_{\pm 0.1}$	$32.0_{\pm 0.3}$

Comparable performance on node classification tasks

RESULTS Efficiency

	Computers		CS		Physics		arXiv		MAG	
	#Param↓	Energy↓	#Param↓	Energy↓	#Param↓	Energy↓	#Param↓	Energy↓	#Param↓	Energy↓
DGI	917.5	0.5	4008.9	8	4833.3	6	590.3	5	590.3	568
GRACE	656.1	1.1	3747.5	17	4571.9	13	328.9	21	328.9	4463
CCA-SSG	262.4	17	1808.1	152	2220.2	352	98.8	78	98.8	340
BGRL	658.4	25	3749.8	163	4574.2	373	331.2	180	331.2	787
SUGRL	193.8	13	2131.2	147	2615.1	342	99.5	26	99.5	117
GGD	254.7	15	3747.3	140	4571.6	340	30.0	100	30.0	1400
Average	490.4	11.9	3198.7	104.5	3906.0	237.6	246.4	68.3	246.4	1279.1
SpikeGCL	60.9	0.038	460.7	0.048	564.4	0.068	7.3	0.2	6.6	0.18



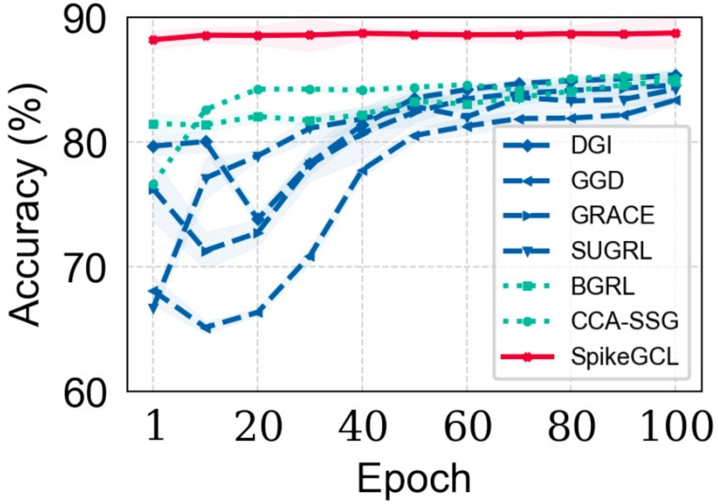
SpikeGCL has significant advantages in terms of parameters, speed, memory usage, and energy consumption

RESULTS Ablation Study

➤ GNN Encoder

	Computers	Photo	CS	Physics	arXiv	MAG
SAGE	84.4$_{\pm 0.9}$	91.4$_{\pm 0.3}$	90.8$_{\pm 0.9}$	94.6$_{\pm 0.1}$	65.7$_{\pm 0.7}$	28.2$_{\pm 0.1}$
GAT	87.9$_{\pm 0.6}$	92.4$_{\pm 0.5}$	90.5$_{\pm 0.8}$	92.6$_{\pm 0.3}$	68.0$_{\pm 0.1}$	30.8$_{\pm 0.8}$
GCN	**88.9**$_{\pm 0.3}$	**93.0**$_{\pm 0.1}$	**92.8**$_{\pm 0.1}$	**95.2**$_{\pm 0.6}$	**70.9**$_{\pm 0.1}$	**32.0**$_{\pm 0.3}$

➤ Convergence

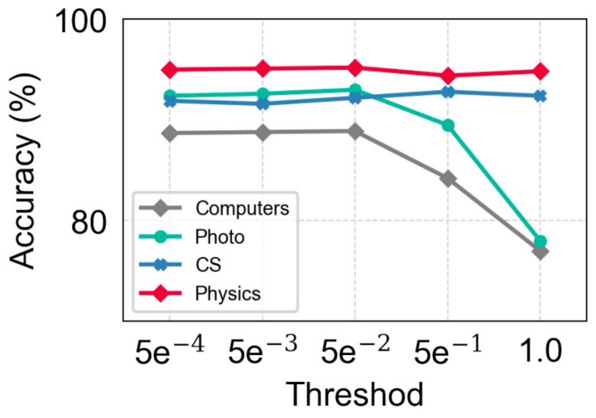


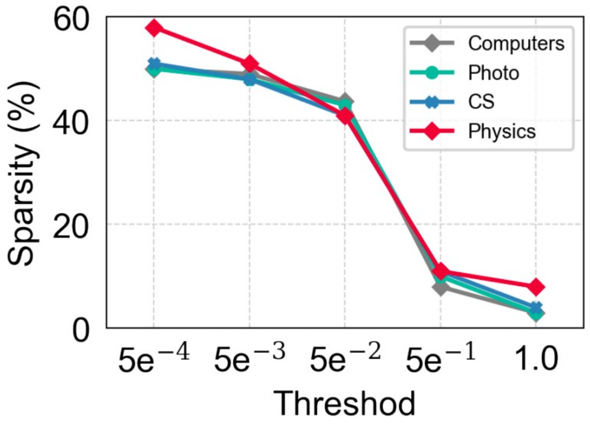
➤ Spiking Neuron

	Computers	Photo	CS	Physics	arXiv	MAG
IF	88.1$_{\pm 0.6}$	92.9$_{\pm 0.2}$	91.2$_{\pm 0.6}$	95.0$_{\pm 0.5}$	68.2$_{\pm 0.6}$	31.0$_{\pm 0.5}$
LIF	88.6$_{\pm 0.1}$	92.9$_{\pm 0.1}$	92.1$_{\pm 0.3}$	**95.3**$_{\pm 0.1}$	68.7$_{\pm 0.8}$	29.8$_{\pm 0.5}$
PILF	**88.9**$_{\pm 0.3}$	**93.0**$_{\pm 0.1}$	**92.8**$_{\pm 0.1}$	95.2$_{\pm 0.6}$	**70.9**$_{\pm 0.1}$	**32.0**$_{\pm 0.3}$

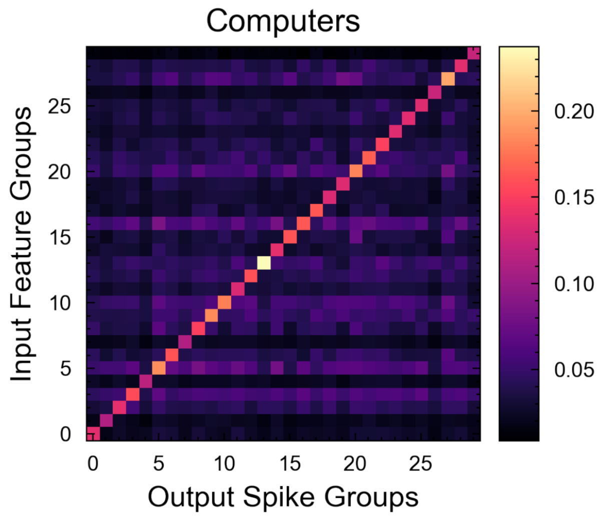
RESULTS Ablation Study

➢ Firing Threshold





➢ Connections between Input Features and Output Spikes



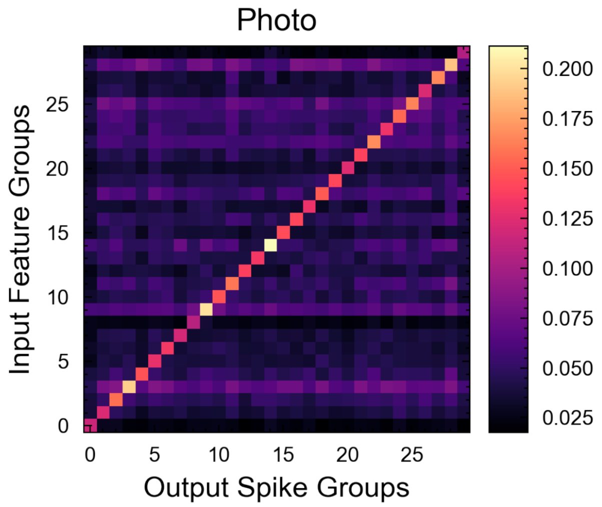


TABLE OF CONTENTS

CONCLUSION

1. Self-supervised and binarized representation learning problem on graphs with SNNs

2. SpikeGCL, a binarized contrastive learning framework for graphs with theoretical performance guarantees

3. Blockwise training, a local learning paradigm to prevent SNNs from vanishing gradients and network degradation

4. SpikeGCL exhibits comparable performance and additional advantages in terms of parameters, speed, memory usage, and energy consumption

Thanks & QA?

Jintang Li

Sun Yat-sen University

唐今里

https://github.com/EdisonLeeeee/SpikeGCL

https://edisonleeeee.github.io/

lijt55@mail2.sysu.edu.cn

中山大學 SUN YAT-SEN UNIVERSITY

ANT GROUP

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE