清華大學电子工程系
Department of Electronic Engineering, Tsinghua University

NTGAT: A Graph Attention Network Accelerator with Runtime Node Tailoring

Wentao Hou*, Kai Zhong*, Shulin Zeng, Guohao Dai, Huazhong Yang, Yu Wang

Department of Electronic Engineering, BNRist, Tsinghua University, Beijing, China

NICS EFC

Contents

- Background
- Algorithm Optimizations
- Hardware Design
- Evaluations

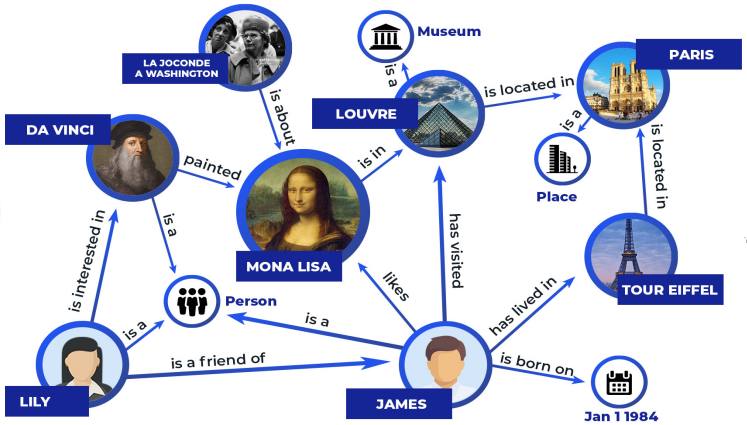
Contents

- **Background**
- Algorithm Optimizations
- Hardware Design
- Evaluations

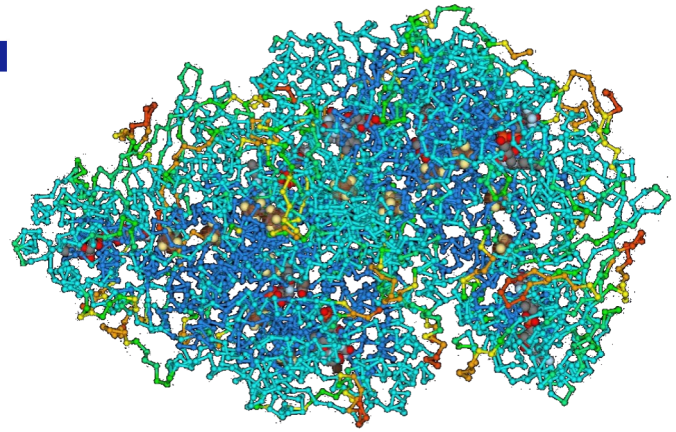
Background

- Graphs are widely-used in many real-world tasks.

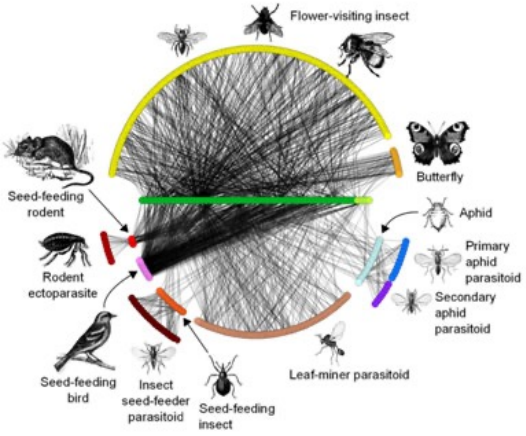
Social Network



Knowledge Database



Biochemistry Macromolecule



Ecosystem

Background

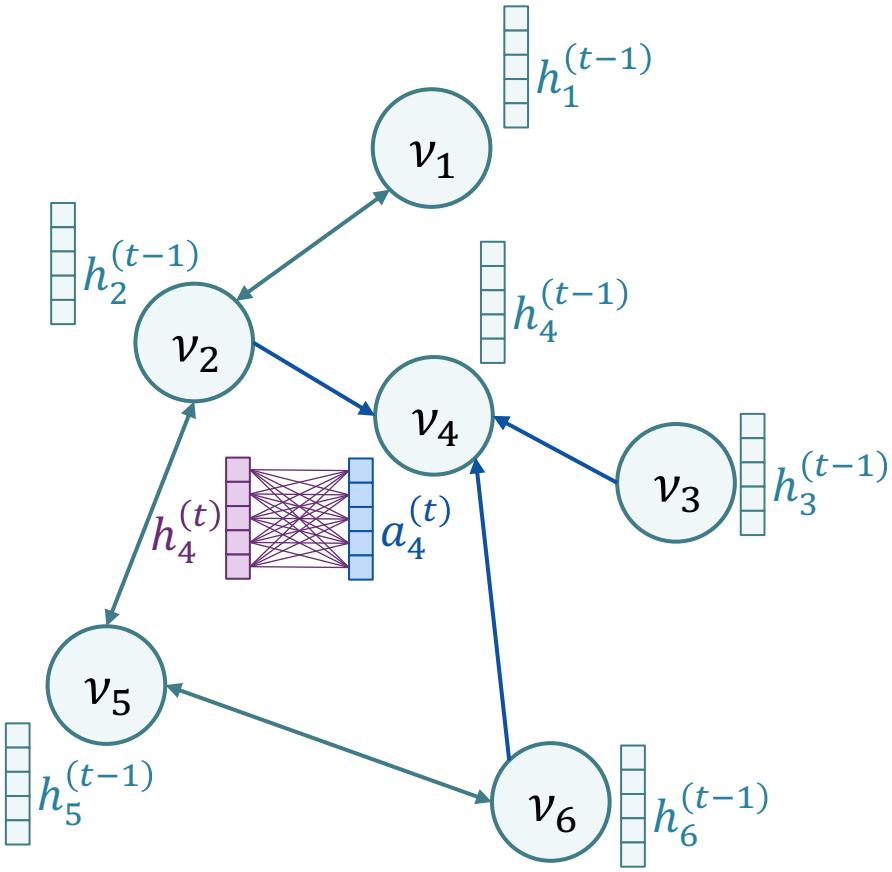
- Graph Neural Network (GNN) is a powerful tool to process graph data[1].
  - It consists of feature aggregation (FA) and feature transformation (FT).

Models	Cora	Citeseer	Pubmed
SemiEmb	59.0%	59.6%	71.1%
LP	68.0%	45.3%	63.0%
Deep Walk	67.2%	43.2%	65.3%
ICA	75.1%	69.1%	73.9%
Planetoid	75.7%	64.7%	77.2%
GCN	**81.5%**	**70.3%**	**79.0%**

GNN:

$$a_v^{(t)} = aggregate\left(h_u^{(t-1)} : u \in N(v)\right)$$

$$h_v^{(t)} = transform\left(a_v^{(t)}, W^{(t-1)}\right)$$



[1] Kipf, Thomas and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks." ArXiv abs/1609.02907 (2016): n. pag.

Background

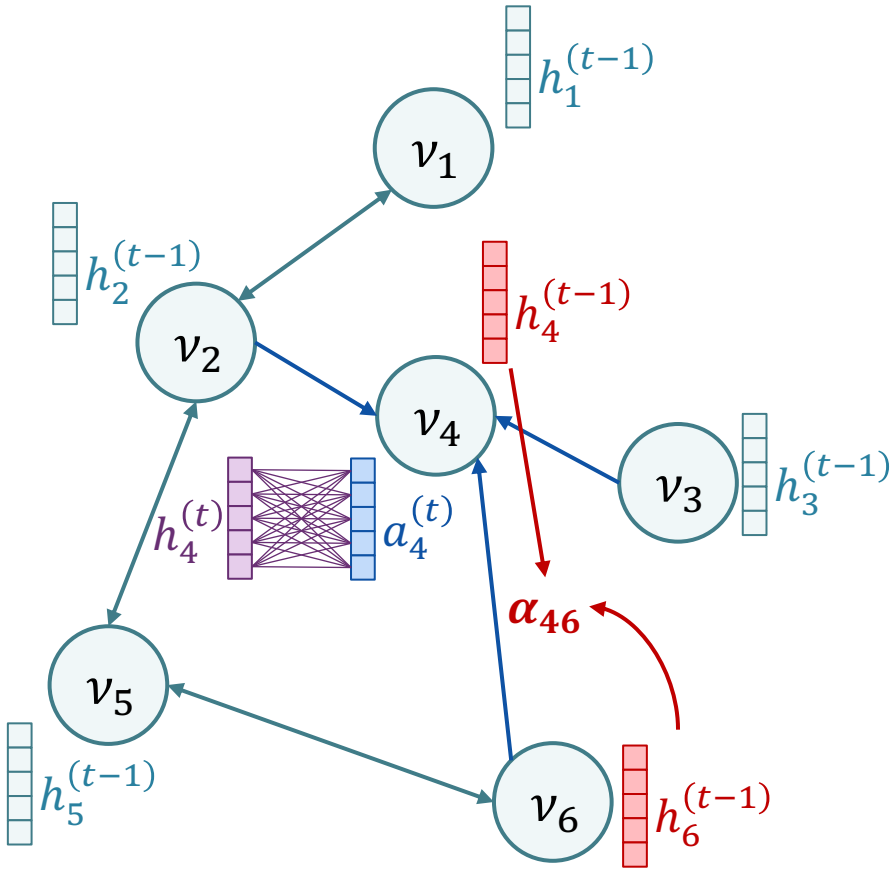
- Graph Attention Networks (GAT) introduce attention mechanism to achieve higher accuracy[1]

Models	Cora	Citeseer	Pubmed
MLP	55.1%	46.5%	71.4%
Chebyshev	81.2%	69.8%	74.4%
GCN	81.5%	70.3%	79.0%
MoNet	81.7±0.5%	—	78.8±0.3%
GCN-64	81.4%±0.5%	70.9±0.5%	79.0±0.3%
GAT	**83.0±0.7%**	**72.5±0.7%**	**79.0±0.3%**

GAT:

$$x_{vu} = f\left(Wh_v^{(t-1)}, Wh_u^{(t-1)}\right)$$

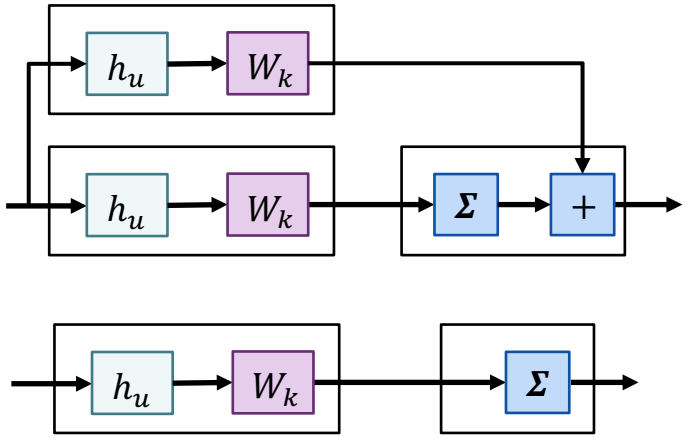
$$\alpha_{vu} = softmax(x_{vu}) = \frac{e^{x_{vu}}}{\sum_{i \in N(v)} e^{x_{vi}}}$$



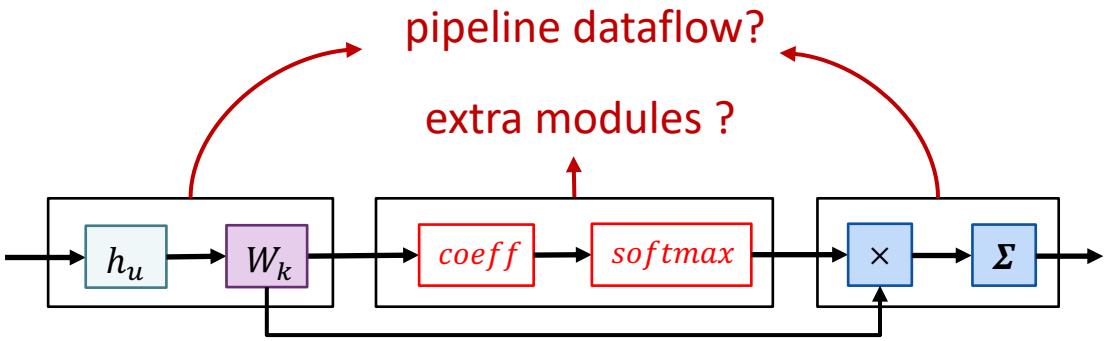
[1] Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio', P., & Bengio, Y. (2018). Graph Attention Networks. ArXiv, abs/1710.10903.

Background

- Existing GNN accelerators[123] focus on GCN, and can not support GAT efficiently.
  - Attention needs new operators: “softmax” calculation and “writing edge data” dataflow do not appear in other GNNs.
  - New attention operators break the pipeline between FA and FT and leads to inefficient memory access.



other GNN dataflow



GAT dataflow

[1] Kiningham, Kevin et al. “GRIP: A Graph Neural Network Accelerator Architecture.” ArXiv abs/2007.13828 (2020): n. pag.
[2] Yan, Mingyu et al. “HyGCN: A GCN Accelerator with Hybrid Architecture.” 2020 IEEE International Symposium on High Performance Computer Architecture (HPCA): 15-29.
[3] Liang, Shengwen et al. “EnGN: A High-Throughput and Energy-Efficient Accelerator for Large Graph Neural Networks.” IEEE Transactions on Computers 70 (2019): 1511-1525.

Background

- GAT has more edge-related computation.
  - Use software-hardware co-design to reduce workload?

GCN-64

$$a_v^{(t)} = aggregate\left(h_u^{(t-1)} : u \in N(v)\right)$$

$$h_v^{(t)} = transform\left(a_v^{(t)}, W^{(t-1)}\right)$$

agg	$trasf$
$64\times 2e$	$64^2\times n$

GAT-8×8

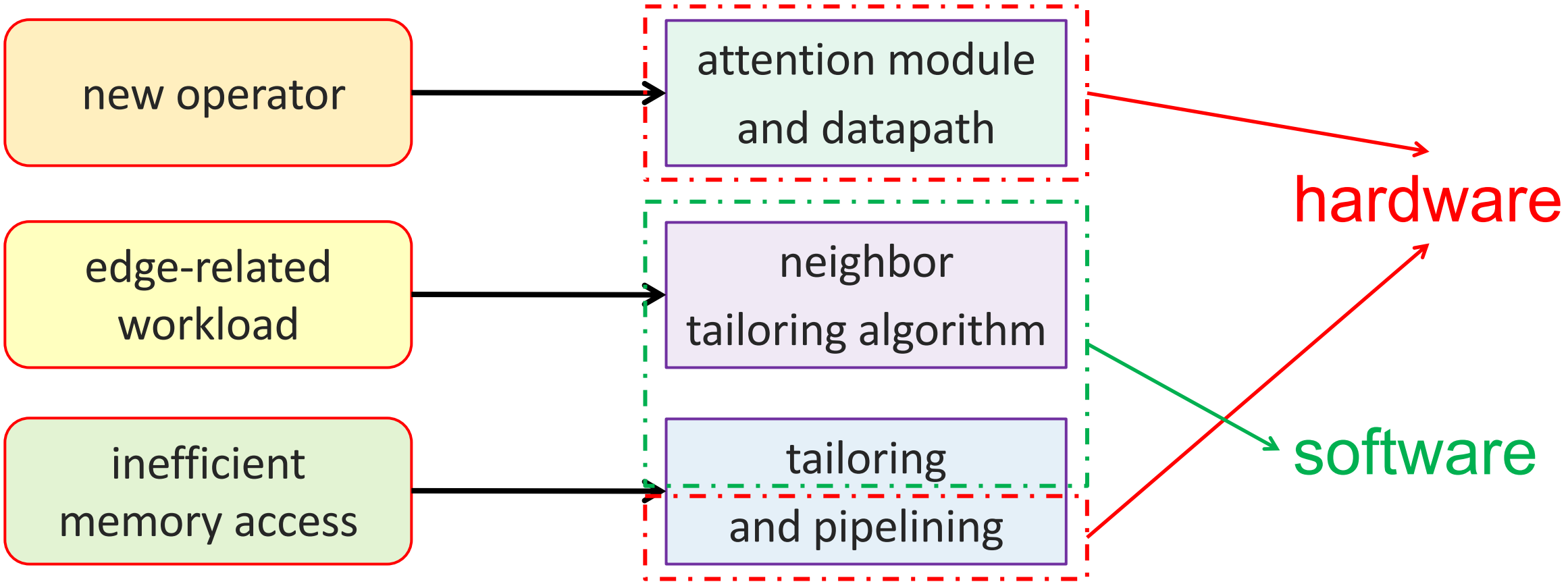
$$x_{vu} = f\left(Wh_v^{(t-1)}, Wh_u^{(t-1)}\right)$$

$$\alpha_{vu} = softmax(x_{vu}) = \frac{e^{x_{vu}}}{\sum_{i\in N(v)} e^{x_{vi}}}$$

agg	$trasf$	$Coeff$	$Softmax$
$64\times 2e$	$64^2\times n$	$8\times 2e$	$8\times 2e$

Proposal

- GAT acceleration with software-hardware co-design



Contents

- Background
- **Algorithm Optimizations**
- Hardware Design
- Evaluations

Algorithm Optimizations

- Neighbors can be tailoring in GAT
  - The sum of largest 10% attention coefficients a destination node accounts for 80% of the total.
  - Our empirical tailoring function.

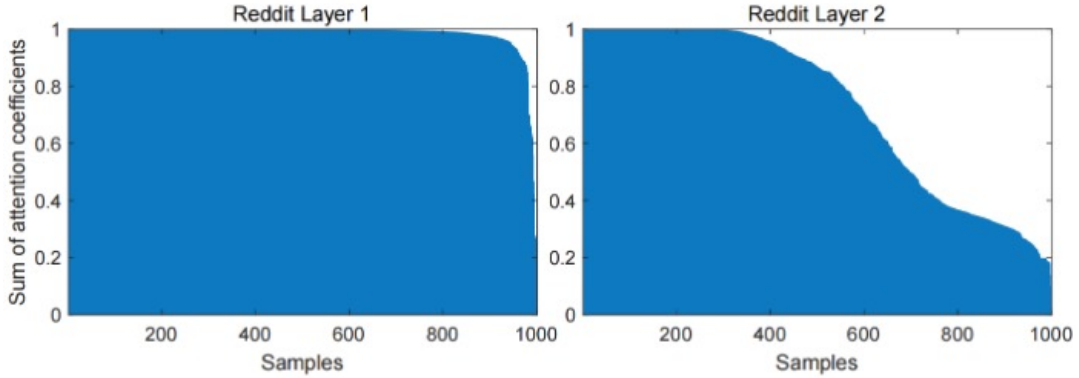
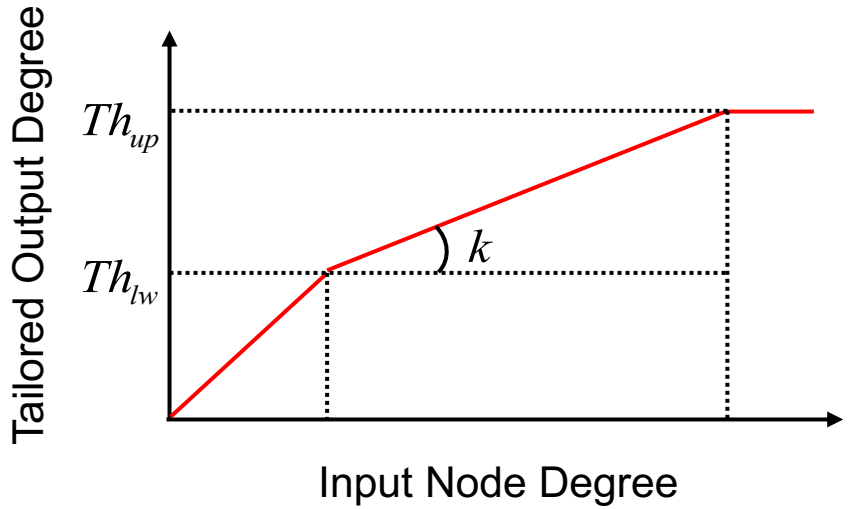


Figure 2: Sampling results of Reddit dataset in descending order. The vertical coordinates are the sum of largest 10% attention coefficients of each sampled node.



$$f_t(d) = \begin{cases} d & d \le Th_{lw} \\ Th_{lw} + (d - Th_{lw}) \times k & Th_{lw} < d \le \frac{Th_{up} + (k-1)Th_{lw}}{k} \\ Th_{up} & else \end{cases}$$

Algorithm Optimizations

- Neighbors can be tailoring in GAT
  - GAT layer with our runtime tailoring algorithm.

Algorithm 1: GAT layer with runtime node tailoring

Input: Graph g, Weight W, Attentional mechanism a, Hidden Features $H = \{h_1, h_2 ...\}$, Tailoring function $f_t()$, Tailoring parameters Th_{up}, Th_{lw}, k

Output: GAT Convolution Result $H^* = \{h_1^*, h_2^* ...\}$

1 **for** *node i in Iterate*$(g.nodes)$ **do**
2 $\quad X_i = Wh_i$;
3 $\quad [e_{iL} \parallel e_{iR}] = \text{Transform}(a, X_i)$;
4 **for** *node i in Iterate*$(g.nodes)$ **do**
5 $\quad d^* = f_t(degree(i), Th_{up}, Th_{lw}, k)$; //tailored degree
6 $\quad$ Fetch $E_i = \{e_{in_1}, e_{in_2} ... e_{in_d}\}$, for n_k in g.Neighbours(i);
7 $\quad$ Sort(E_i) ; // in parallel with fetching
8 $\quad E_i^* = E_i[0 : d^*]$; // only first d^* nodes are kept
9 $\quad \alpha_i = softmax(E_i^*)$;
10 $\quad h_i^* = \sum \alpha_{ij} X_j$ for *node j* in E_i^*;
11 **return** H^*;

decide how many nodes to keep

tailoring small ones by sorting

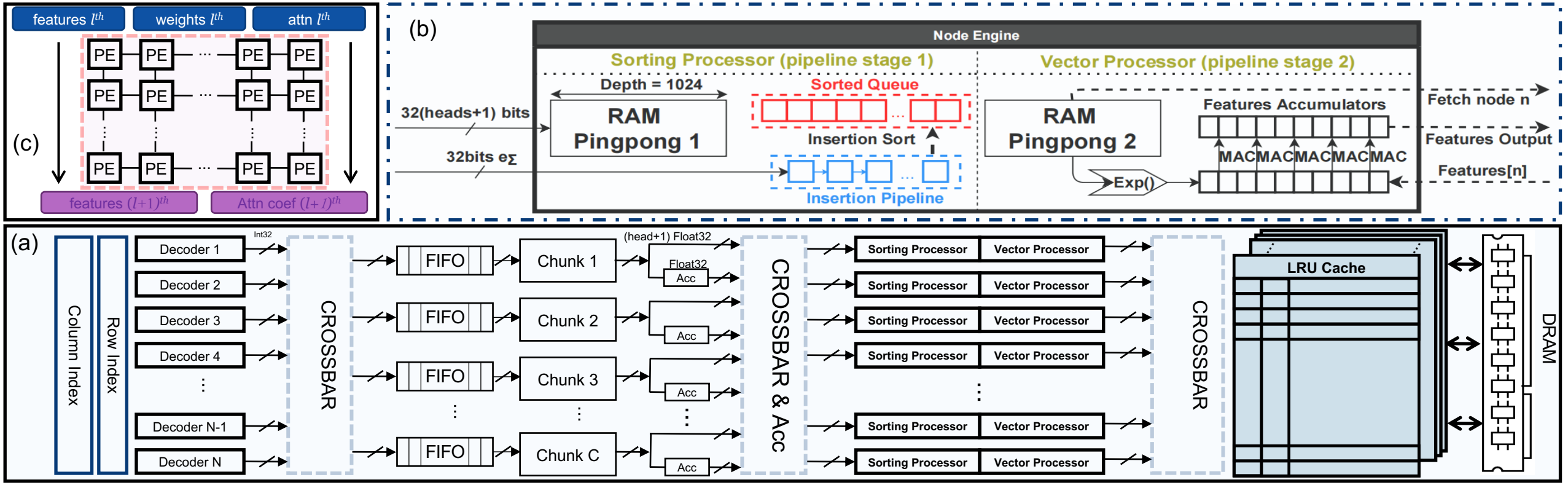
only aggregate the remaining nodes

Contents

- Background
- Algorithm Optimizations
- **Hardware Design**
- Evaluations

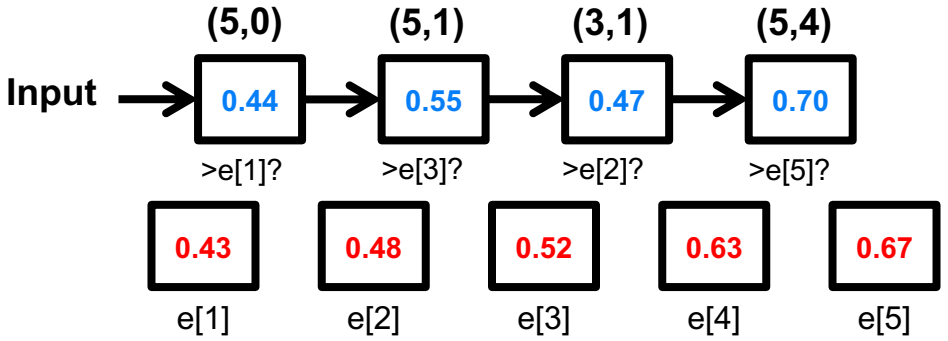
Hardware Design

- Overall Architecture
  - dense engine has a systolic array to compute transformation
  - aggregation engine is a 4-stage pipeline: decode, fetch, sort, aggregate
  - a LRU cache can be configured for different feature lengths

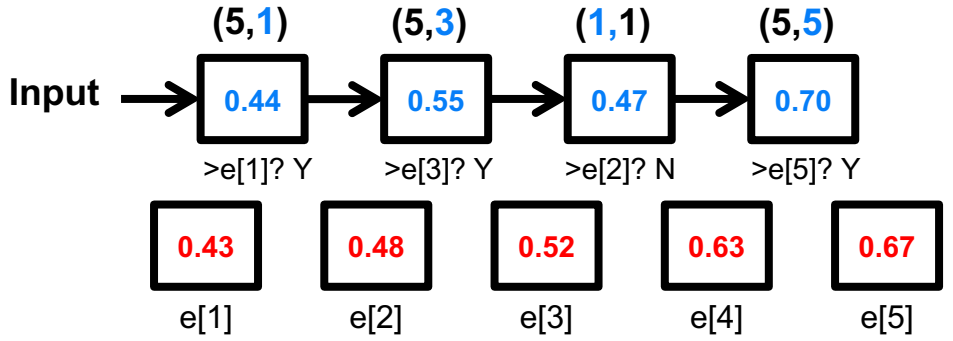


Hardware Design

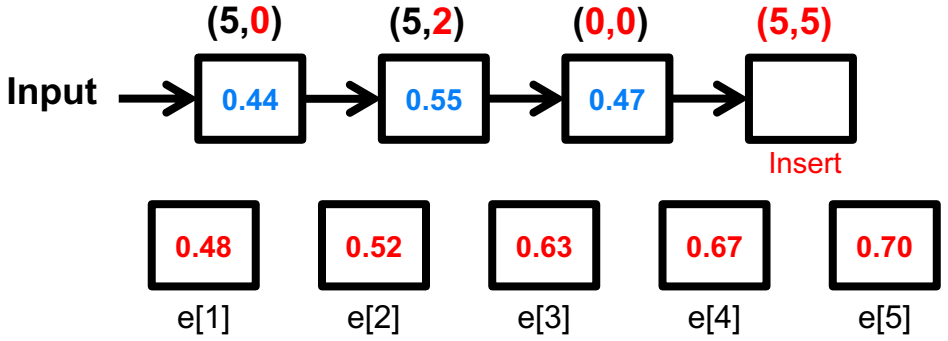
- Sorting Unit
  - sort “e list” with a throughput of one element per cycle
  - pipelined binary sorting



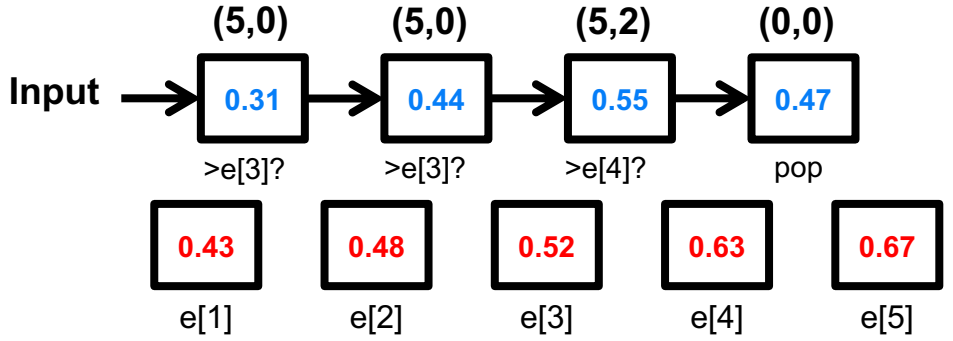
(a) Insertion units compare with registers



(b) Update left and right indices



(c) Inserts one node into queue



(d) Pipeline steps, next cycle starts.

Contents

- Background
- Algorithm Optimizations
- Hardware Design
- **Evaluations**

Evaluations

- Experiment Setup
  - Datasets: CORA, CITESEER, PUBMED, ARXIV, REDDIT
  - GAT Model: layer=2/3, hidden=8/64/256, head=2/3/8
  - Hardware:
    - Xilinx Alveo U200 FPGA
    - 4096+2048+64=6208 DSPs. On-chip RAM consumption is 32.5MB, 8MB used by attention coefficient chunks, 2MB used by ping-pong buffer in dense computing kernel, 2.5MB used by ping-pong buffer in node engines, and 20MB used as feature cache.
  - CPU, GPU baselines:
    - Framework: DGL
    - CPU： two 16-core Intel Xeon Gold 6226R CPU
    - GPU： Nvidia RTX 2080 GPU, RTX 3090, Tesla V100S with CUDA 11.6

Evaluations

- End-to-end results:
  - 3.8× speedup and 4.98× energy efficiency compared to GPU baseline

Table 5: Inference time of different platforms. NTGAT-B denotes hardware baseline without algorithm. NTGAT-T denotes NTGAT plus tailoring. NTGAT-S denotes NTGAT plus tailoring after scaling resources.

Platform	CPU	RTX 2080	Tesla V100S	RTX 3090	NTGAT-B	NTGAT-T	NTGAT-S
Bandwidth	/	448 GB/s	1134 GB/s	936GB/s	77 GB/s	77 GB/s	308 GB/s
FP32 Operation	/	10.6 TFLOPS	15.7 TFLOPS	35.6 TFLOPS	1.8624 TFLOPS	1.8624 TFLOPS	12.3648 TFLOPS
Frequency	2.9 GHz	1.515 GHz	1.245 GHz	1.40 GHz	300 MHz	300 MHz	1.2 GHz
Cora	39.85 ms	2.123 ms	2.256 ms	2.580 ms	0.559 ms	0.559 ms	0.0885 ms
Citeseer	19.24 ms	2.115 ms	2.207 ms	2.516 ms	1.465 ms	1.465 ms	0.205 ms
Pubmed	19.92 ms	2.162 ms	2.230 ms	2.976 ms	2.097 ms	2.096 ms	0.391 ms
ogbn-arxiv	1.452 s	0.200 s	0.136 s	0.111 s	0.262 s	0.257 s	0.0406 s
Reddit	0.257 s	OoM	0.198 s	0.150 s	0.424 s	0.176 s	0.0423 s

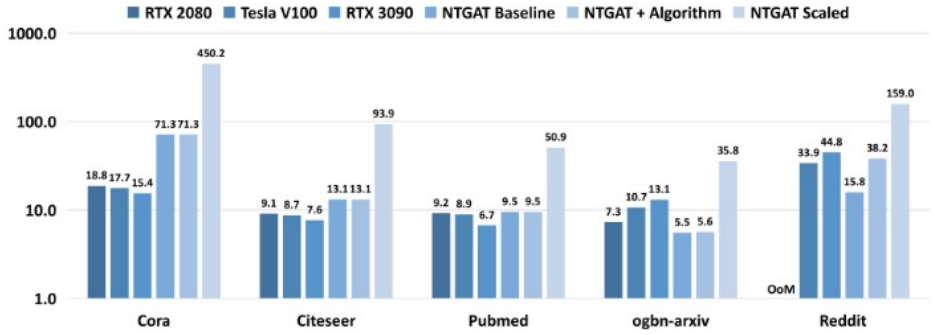


Figure 6: Speedup comparing to CPU baseline.

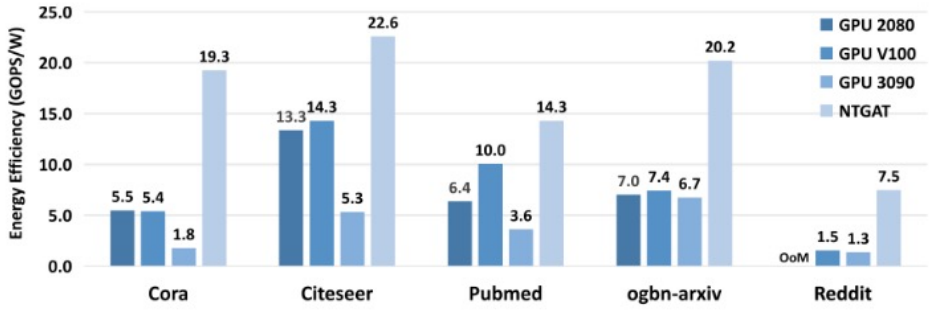


Figure 7: Energy efficiency of different platforms.

Evaluations

- Tailoring algorithm evaluation
  - up to 86% node aggregation workload can be tailored
  - incurring only slight accuracy loss

Table 2: Original accuracy, accuracy of node tailoring algorithm experiment, and accuracy loss

	Cora	Citeseer	Pubmed	ogbn-arxiv	Reddit
Original	81.90%	69.92%	77.24%	73.19%	94.45%
Ours	81.85%	69.86%	77.18%	73.00%	94.29%
Loss	-0.32%	-0.06%	-0.06%	-0.19%	-0.15%

Table 3: Nodes tailored. ogbn-arxiv includes two layers.

	Cora	Citeseer	Pubmed	ogbn-arxiv	Reddit
Total	13264	12555	108368	2484941	114848857
Tailored	536	1949	3973	370120/232418	99320583
(%)	4.04	15.52	3.67	14.89/9.35	86.45

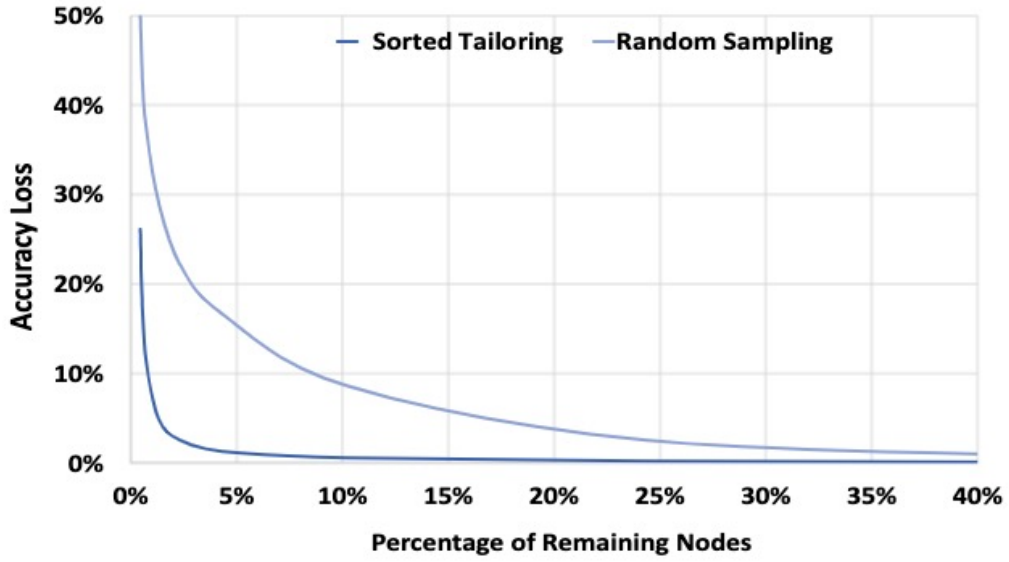


Figure 5: The accuracy loss when using sorted node tailoring or random neighbor sampling during inference.

Thanks!

Kai Zhong

zhongk19@mails.tsinghua.edu.cn