SGQuant: Squeezing the Last Bit on Graph Neural Networks with Specialized Quantization

Abstract

文章针对【减少现有GNN模型的内存占用】，提出了【一种针对GNN的量化方法。具体来说，第一，文章针对GNN的不同组合部分提出了三种量化方式。这三种方式分别正对GNN中的不同数据成分、不同拓扑结构和不同层的特征进行不同的量化。第二，文章提出了一种自动选择量化比特的算法，为不同的量化部分选择不同的量化比特数】，取得了【将现有GNN的内存占用最多压缩到31.9倍的成果，且模型准确率平均只下降0.4%】。

Motivation

1) what types of data (weight or features) should be quantized?

2) what is the efficient quantization scheme suitable for GNNs?

3) How to determine the quantization bits?

Observation: features take up to 99.89% of the overall memory size

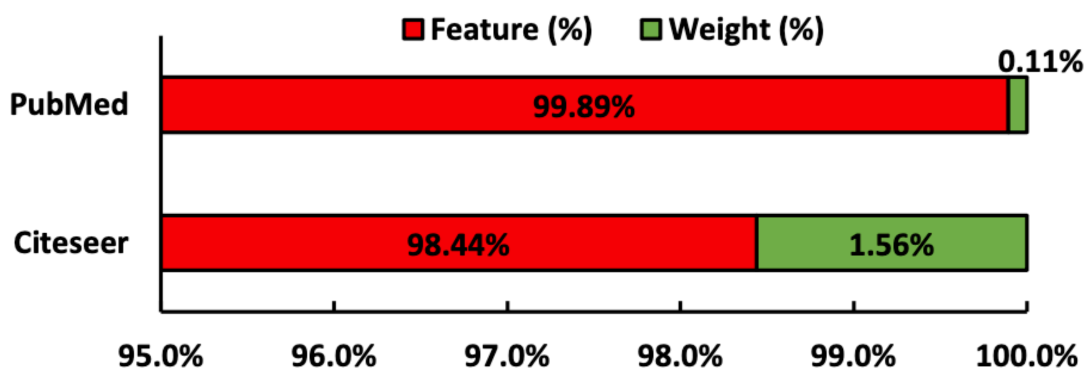


Fig. 1: GAT Feature/Weight Memory Size Ratio.

- SGQuant only quantizes the GNN features

Details

Quantization Method

文章采用了简单的线性量化的方式：

$$\alpha^{k,(q)} = \left\lfloor \frac{\alpha^k - \alpha_{min}}{scale} \right\rfloor.$$

Unmatching Problem

当不同量化比特的数据相乘的时候会出现不匹配的问题：将数据恢复成32bit的数据再相乘

$$\alpha_{u,v}^{k,(q)'} = scale \cdot \alpha_{u,v}^{k,(q)} + \alpha_{min}$$

例如将attention和feature的数值都变为32bit的时候相乘：

$$h_v^{k+1} = \mathcal{W}_{com}^{k+1} \cdot \sum_{u \in \mathcal{N}(v)} \alpha_{u,v}^{k,(q)'} h_u^{k,(p)'}$$

作者提到：这种rematching的方式不会带来过多内存开销，因为计算一个node的feature时只恢复了部分数值（存疑）

Fintuning Method

量化结束后进行网络的微调

Backpropagation Follows：

$$\frac{\partial L}{\partial \alpha_{u,v}^{k,(q)'}} = \mathcal{W}_{com}^{k+1} \cdot \left(\frac{\partial L}{\partial h_v^{k+1}} \cdot h_u^{k,(p)'} + \frac{\partial L}{\partial h_u^{k+1}} \cdot h_v^{k,(p)'}\right)$$
$$\frac{\partial L}{\partial \alpha_{u,v}^{k}} = \frac{\partial L}{\partial \alpha_{u,v}^{k,(q)'}} \cdot scale \cdot \frac{\partial \alpha_{u,v}^{k,(q)}}{\partial \alpha_{u,v}^{k}}$$

其中attention数值的反传为：

$$\begin{aligned} \frac{\partial L}{\partial \alpha_{u,v}^{k}} &= \frac{\partial L}{\partial \alpha_{u,v}^{k,(q)'}} \cdot scale \cdot \frac{\partial \alpha_{u,v}^{k,(q)}}{\partial \alpha_{u,v}^{k}} \\ &= \frac{\partial L}{\partial \alpha_{u,v}^{k,(q)'}} \end{aligned}$$

量化数值对于原数值的偏导数近似为1/scale

Multi-Granularity Quantization

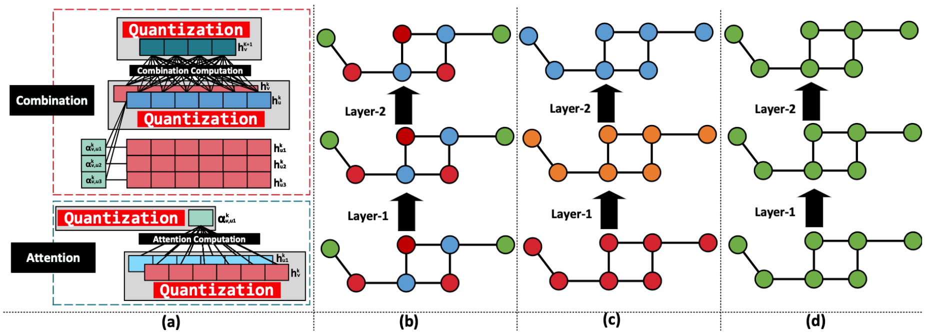


Fig. 4: Multi-Granularity Quantization: (a) Component-wise, (b) Topology-aware, (c) Layer-wise, and (d) Uniform Quantization. NOTE: the same color represents the same quantization bit.

Component-Wise Quantization (CWQ)

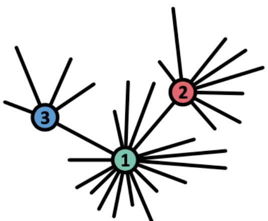
Quantize attention matrix and embedding matrix differenly

- Attention values are quantized to lower bits

Topology-Aware Quantization (TAQ)

Quantize node feature of different degrees differenly

- Higher degree node can be quantized to lower bits



Node ID	Degree	QBit
Node-1	17	Fbit(17)
Node-2	9	Fbit(9)
Node-3	5	Fbit(5)

(a)

```
func Fbit(degree):
    std_qbit = [1, 2, 4, 8]
    split_point = [D1, D2, D3]
    if degree in [0, D1):
       return std_qbit[3]
    if degree in [D1, D2):
       return std_qbit[2]
    if degree in [D2, D3):
       return std_qbit[1]
    if degree in [D3, +inf):
       return std_qbit[0]
```

(b)

Fig. 5: Topology-aware Quantization.

- also use the “rematching” technique on node embeddings and compute the combination component

Layer-Wise Quantization (LWQ)

Quantize values of different layers differenly

- leading layers need more quantization bits
- Succeeding layers are quantied to lower bits

Auto-Bit Selection

How can we assign quantization bits for different granularities to achieve the sweet point between accuracy and memory saving?

Formulate quantization problem as combinatorial optimization

$$\min_{\substack{q_{k,att},\\ q_{k,com,D_j}}} Loss(\alpha_{u,v}^{k,(q_{k,att})}, h_u^{k,(q_{k,com,D_j})}, W_{com}^k, W_{att}^k)$$

- large design space
- large diversity exists in the GNN model
- graph topology varies

Use a learning cost model to predict the accuracy of the quantized GNN (Like autoML) **iteratively**

- **Step 1** Randomly select a small number *N_mea* of configurations, extract features, and measure their accuracies.
- **Step 2** Train the ML cost model based on the collected features and labels.
- **Step 3** Sample a large number *N_sample* of configurations, use the ML cost model to predict their accuracy, and find the ones with the top-*N_mea* accuracy.
- **Step 4** Extract features of the selected configurations and measure their accuracies.
- **Step 5** Repeat Step2 - Step4 until reaching *N_iter* iterations.

negligible latency (< 0.1 seconds) at each iteration

Results

Configuration

TABLE I: GNN Architectures.

Arch	Specification
GCN	hidden=32, #layers=2
AGNN	hidden=16, #layers=4
GAT	hidden=256, #layers=2

TABLE II: Datasets for Evaluation.

Dataset	#Vertex	#Edge	#Dim	#Class
Citeseer	3,327	9,464	3,703	6
Cora	2,708	10,858	1,433	7
Pubmed	19,717	88,676	500	3
Amazon-computer	13,381	245,778	767	10
Reddit	232,965	114,615,892	602	41

Quantitative Results

TABLE III: Overall Quantization Performance.

Dataset	Network	Accuracy (%)	Average Bits	Memory Size (MB)	Saving
Cora	GCN (Full-Precision)	82.2	32	15.42	-
	GCN (Reduced-Precision)	81.72	1.22	0.59	26.1×
	AGNN (Full-Precision)	83.16	32	15.94	-
	AGNN (Reduced-Precision)	82.75	2.15	1.07	14.90×
	GAT (Full-Precision)	82.50	32	16.21	-
	GAT (Reduced-Precision)	82.10	2.58	1.31	12.37×
Citeseer	GCN (Full-Precision)	71.82	32	51.06	-
	GCN (Reduced-Precision)	71.54	1.01	1.6	31.9×
	AGNN (Full-Precision)	71.58	32	50.01	-
	AGNN (Reduced-Precision)	71.18	1.08	1.69	29.59×
	GAT (Full-Precision)	71.10	32	59.49	-
	GAT (Reduced-Precision)	70.70	2.42	3.82	13.2×
Pubmed	GCN (Full-Precision)	80.36	32	43.71	-
	GCN (Reduced-Precision)	80.28	2.9	4.01	10.9×
	AGNN (Full-Precision)	80.44	32	43.46	-
	AGNN (Reduced-Precision)	80.31	3.07	4.17	10.42×
	GAT (Full-Precision)	78.00	32	44.48	-
	GAT (Reduced-Precision)	77.30	3.77	5.26	8.47×
Reddit	GCN (Full-Precision)	81.07	32	328.70	-
	GCN (Reduced-Precision)	80.36	3.72	38.25	8.59×
	AGNN (Full-Precision)	74.63	32	643.92	-
	AGNN (Reduced-Precision)	74.40	4	113.92	5.65x
	GAT (Full-Precision)	92.66	32	311.85	-
	GAT (Reduced-Precision)	92.23	4.07	39.70	7.86×
Amazon-Computer	GCN (Full-Precision)	89.57	32	44.58	-
	GCN (Reduced-Precision)	89.39	3.29	4.59	9.72×
	AGNN (Full-Precision)	77.69	32	44.16	-
	AGNN (Reduced-Precision)	77.33	4	5.99	7.37×
	GAT (Full-Precision)	93.10	32	45.71	-
	GAT (Reduced-Precision)	92.60	7.53	10.75	4.25×

- Lower average bits on smaller datasets
- SGQuant would select higher average bits for more complex model
  - involve more intricate computations

Breakdown Analysis of Multi-granularity Quantization

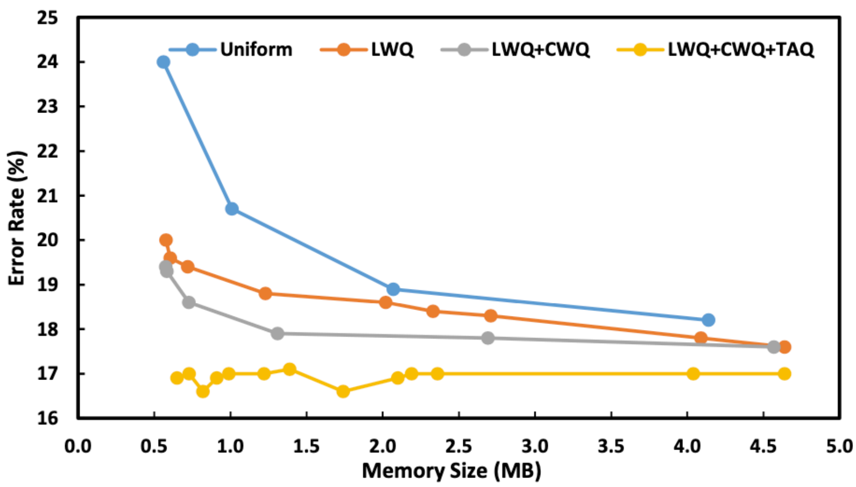


Fig. 7: Breakdown Analysis of Multi-granularity Quantization.

fine-grained granularities generally lead to lower error rate at a given memory size.

Effectiveness of Auto-Bit Selection

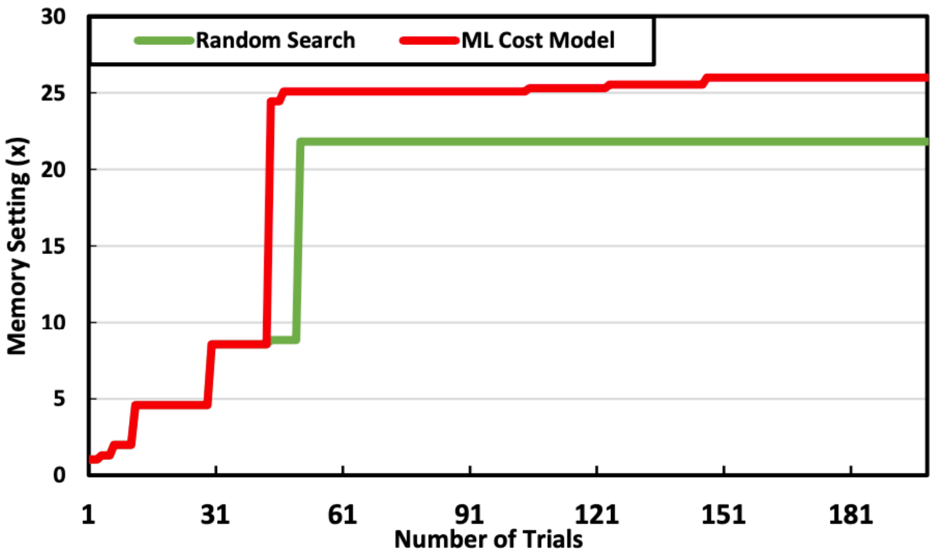


Fig. 8: Benefit of ML Cost Model.

ML cost model can pinpoint a more ”optimal” value for bits that offers higher memory saving (25×) compared with random search (20×)

Comments

文章对GNN模型提出了针对不同部分的多bit的量化方式，并且提出了一个启发式的搜寻最佳量化bit的方法。方法比较简单但效果任然显著。但文章在rematching操作上对内存占用的影响讨论较少，存在占用过多内存的疑问。