VQ-GNN: A Universal Framework to Scale up Graph Neural Networks using Vector Quantization

Mucong Ding*, Kezhi Kong*, Jingling Li, Chen Zhu,
John P Dickerson, Furong Huang, Tom Goldstein

University of Maryland, College Park

COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

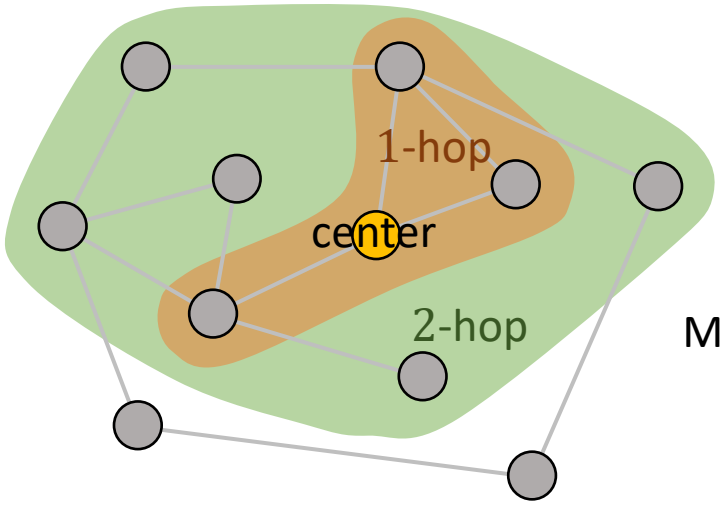
Scalability Problem of GNNs

Most Graph Neural Networks → Defined as → **Graph Convolutions**

Message passing between direct neighbors (or beyond)

"Neighbor Explosion" Problem:

A L-layer GNN (at least) aggregates information from all L-hop neighbors.



Size of neighborhood grows exponentially with L.

$O(bd^L)$ node inputs required on the GPU at once.

Mini-batch size

Average node degree

Memory bottleneck when Scaling-up GNNs

M. Ding*, K. Kong*, J. Li, C. Zhu, J. Dickerson, F. Huang, T. Goldstein. University of Maryland.

Universal Framework that Preserves All Messages

Most scalable methods universal to a variety of GNNs are sampling based.

Drawback: consider only a small subset of messages passed to the nodes in a mini-batch.

1) **Performance not guaranteed** across various tasks and datasets [Hu2020].
2) Require **all neighbors** in the **inference phase** to be non-stochastic.
3) Cannot be applied to **GNNs that utilize many-hop or global context each layer**.

Question: Can we develop a universal framework to scale-up GNNs while preserving all messages in a mini-batch?

Yes, by applying vector quantization to GNNs!

M. Ding*, K. Kong*, J. Li, C. Zhu, J. Dickerson, F. Huang, T. Goldstein. University of Maryland.

Common Framework of GNNs

Cover most of GNNs [Balcilar2020]

We consider all GNNs that can be written as:

Message passing (graph convolution $C^{(s)} \in \mathbb{R}^{n \times n}$)

Activation function

Feature transformation

$$X^{(l+1)} = \sigma\left(\sum_s C^{(l,s)} X^{(l)} W^{(l,s)} \right)$$

Multiple convolutions

Node features ($X^{(l)} \in \mathbb{R}^{n \times f_l}$)

Convolution matrix $C^{(s)}$ can be either: fixed or learnable ($C_{i,j} \propto \mathrm{h}(X_i, X_j)$)

GCN [Kipf2016], SAGE-Mean [Hamilton2017], …

GAT [Veličković2018], GIN [Xu2019], ChebNet [Defferrard2016], Graph Transformers, …

M. Ding*, K. Kong*, J. Li, C. Zhu, J. Dickerson, F. Huang, T. Goldstein. University of Maryland.

Dimensionality Reduction in GNNs

Idea: applying dimensionality reduction to convolution matrix C and node feature matrix X, so that graph convolution can be approximated using the compressed "sketches" of C and X.

To compute a mini-batch $\langle i_b \rangle = i_1, \ldots, i_b$ of forward-passed features $X_B^{(l+1)} = X_{\langle i_b \rangle, :}^{(l+1)}$

We need a slice of C, i.e., $C_B^{(l,s)} = C_{\langle i_b \rangle, :}^{(l,s)}$ and the whole $X^{(l)}$, each of size $O(n)$.

If we have a projection matrix $R \in \mathbb{R}^{n \times k}$, where $k \ll n$, such that,

Sketched convolution matrix Sketched node features

$$C_B^{(l,s)} R \; R^T X^{(l)} \approx C_B^{(l,s)} X^{(l)}$$

Then we only need sketches $C_B^{(l,s)} R$ and $R^T X^{(l)}$, each of size $O(k)$, now fits in memory.

M. Ding*, K. Kong*, J. Li, C. Zhu, J. Dickerson, F. Huang, T. Goldstein. University of Maryland.

Desired Properties of Dimensionality Reduction

The existence of projection R is guaranteed by the sparse JL-lemma [Kane2014]:

Theorem 1: For any convolution matrix C and column vector $X_{:,a}$ of node feature matrix, there **exists** a projection matrix $R \in \mathbb{R}^{n \times k}$ with only $O(\varepsilon)$-fraction of non-zeros, such that,

$$\Pr\left(\left\|CR\,R^T X_{:,a} - CX_{:,a}\right\| < \varepsilon \left\|CX_{:,a}\right\|\right) > 1 - \delta$$

with $k = O(\log(n)/\varepsilon^2)$ and $\delta = O(1/n)$.

Two properties are desired:

- Sparse projection matrix R is favorable:
  1) Given a sparse C, the sketched convolution matrix CR is still sparse.
  2) Updating sketched node features $R^T X$ requires fewer computations.
- For learnable convolutions $C_{i,j} \propto \mathrm{h}(X_i, X_j)$, we can approximate CR as functions of $R^T X$ with $O(k)$ complexities. It is possible if we can estimate any X_i directly from $R^T X$.

M. Ding*, K. Kong*, J. Li, C. Zhu, J. Dickerson, F. Huang, T. Goldstein. University of Maryland.

Vector Quantization

Vector Quantization (VQ) can be formulated as: given $X \in \mathbb{R}^{n \times f}$,

minimize $\left\|X - R\tilde{X}\right\|_F$ s.t. $R \in \mathbb{R}^{n \times k}$; $R_{i,:} \in \{\boldsymbol{e}_k^1, \ldots, \boldsymbol{e}_k^1\}$ for any i; $\tilde{X} \in \mathbb{R}^{k \times f}$,

which is solved by *k-means*.

- The rows of $\tilde{X}$ are the k codewords, and $\tilde{X} = \mathrm{diag}^{-1}(R^T \mathbf{1}_n) R^T X$.
- R encodes codeword assignment. $R_{i,v} = 1$ if and only if the i-th node is assigned to the v-th cluster in *k-means*.

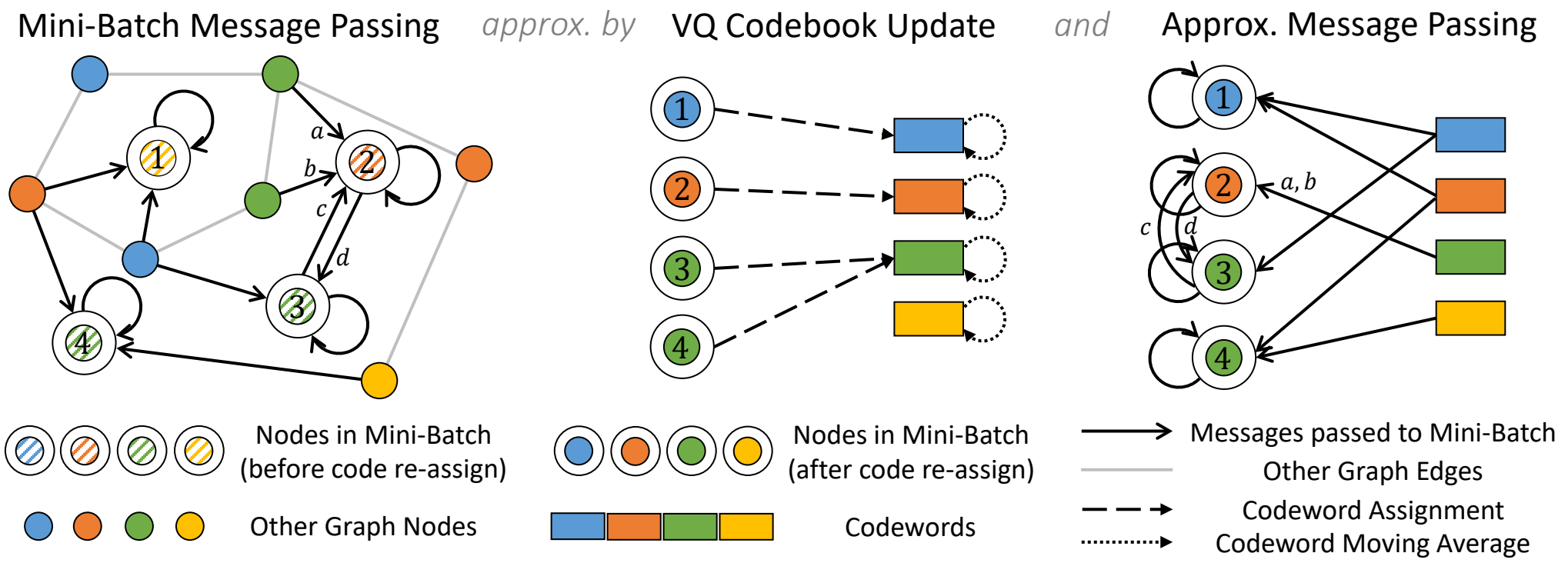
In VQ, each of the n node feature vectors, X_i, is directly approximated by a specific codeword vector $\tilde{X}_j$. The two desired properties naturally hold.

M. Ding*, K. Kong*, J. Li, C. Zhu, J. Dickerson, F. Huang, T. Goldstein. University of Maryland.

Vector Quantized GNNs

Using VQ, we learn and update a small number of quantized reference vectors (codewords) of global node representations in each layer of GNN.
We can approximate all the messages passed to the nodes in a mini-batch.

In VQ-GNN, forward-pass (message passing) in a layer of GNN is approximated by:

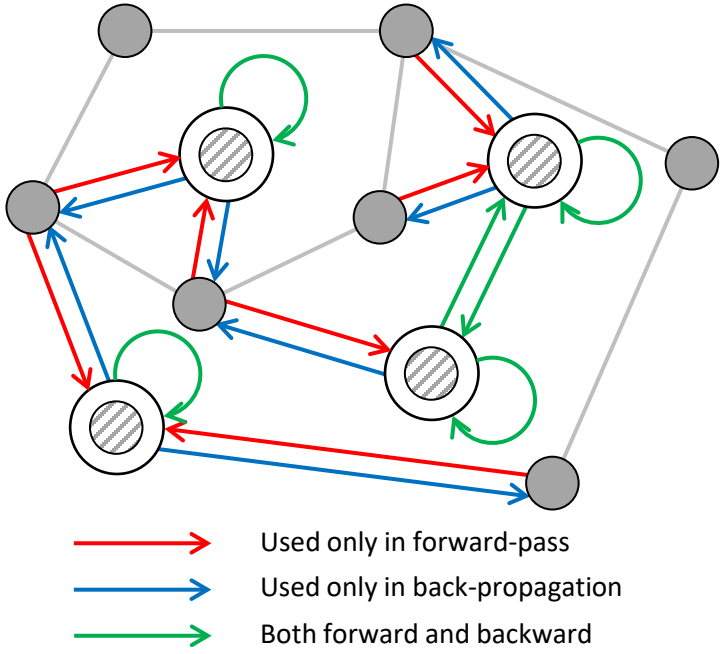


M. Ding*, K. Kong*, J. Li, C. Zhu, J. Dickerson, F. Huang, T. Goldstein. University of Maryland.

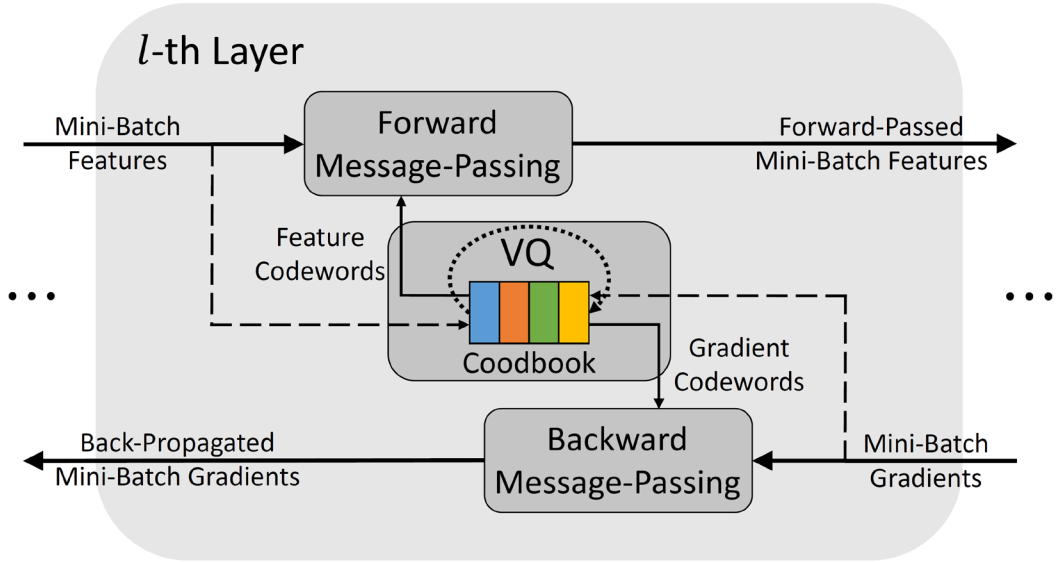
Forward and Backward Message Passing

Back-propagation in a layer of GNN can also be realized by message passing.
In VQ-GNN, we approximate back-propagation (message passing) similarly.

Messages related to a mini-batch are divided into three categories.



We treat forward and backward message passing symmetrically.



M. Ding*, K. Kong*, J. Li, C. Zhu, J. Dickerson, F. Huang, T. Goldstein. University of Maryland.

VQ Update Rule and Error Bounds

We use the VQ update rule proposed in *VQ-VAE* [Oord2017]:
which updates the codewords as exponential moving averages of the mini-batch inputs.

VQ-GNN is guaranteed to approximate the full-graph training:

Theorem 2 & Corollary 3: If the relative error of VQ, ϵ, is upper bounded, under some mild continuity and Lipschitz conditions (only when the convolution is learnable), we prove the errors of approximated forward-pass and back-propagation is also upper bounded.

See our paper for detailed update rules, pseudo code, error-bounds, and discussions.

M. Ding*, K. Kong*, J. Li, C. Zhu, J. Dickerson, F. Huang, T. Goldstein. University of Maryland.

Theoretical Complexities

Our VQ-GNN enjoys competitive training memory and time complexities compared with Cluster-GCN [Chiang2019] and GraphSAINT-RW [Zeng2019], and much faster inference time.

The complexities of L-layer GCN with f-dimensional (hidden) features in each layer, applied to a graph with n nodes and m edges (average degree $d = m/n$) are:

Scalable Method	Memory Usage	Pre-computation Time	Training Time	Inference Time
NS-SAGE	$O(br^L f + Lf^2)$	—	$O(nr^L f + nr^{L-1} f^2)$	$O(nd^L f + nd^{L-1} f^2)$
Cluster-GCN	$O(Lbf + Lf^2)$	$O(m)$	$O(Lmf + Lnf^2)$	
GraphSAINT-RW	$O(L^2 bf + Lf^2)$	—	$O(L^2 nf + L^2 nf^2)$	
VQ-GNN (Ours)	$O(Lbf + Lf^2 + Lkf)$	—	$O(Lbdf + Lnf^2 + Lnkf)$	$O(Lbdf + Lnf^2)$

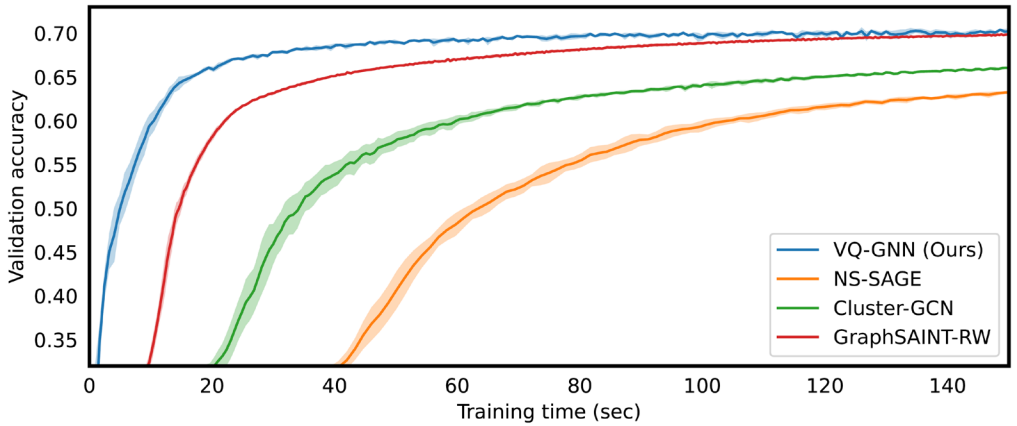
Although we have $O(Lkf)$ memory and $O(Lnkf)$ time overheads to store and update the codewords, they are practically small (compared to other terms), because very small $k \leq 256$ is sufficient in most cases.

M. Ding*, K. Kong*, J. Li, C. Zhu, J. Dickerson, F. Huang, T. Goldstein. University of Maryland.

Experiments: Efficiency of VQ-GNNs

VQ-GNN can converge faster than the sampling-based methods under some setups. Experiments verify the memory overhead of VQ is relatively small.

Convergence curves (left) and peak memory usages (right) of SAGE-Mean on *ogbn-arxiv*.



GNN Model	SAGE-Mean
NS-SAGE	1140.3 MB
Cluster-GCN	514.1 MB
GraphSAINT-RW	519.2 MB
VQ-GNN (Ours)	801.8 MB

The inference time of VQ-GNN is 0.40s, while the sampling-based methods require 1.61s, tested with SAGE-Mean on *ogbn-arxiv.*

M. Ding*, K. Kong*, J. Li, C. Zhu, J. Dickerson, F. Huang, T. Goldstein. University of Maryland.

Experiments: Performance of VQ-GNNs

The performance of VQ-GNN consistently matches the "full-graph" training performance (oracle) across tasks and datasets. But sampling-based methods may fail.

Task	Node Classification *ogbn-arxiv* (Acc.±std.)			Link Prediction *ogbl-collab* (Hits@50±std.)		
GNN Model	GCN	SAGE-Mean	GAT	GCN	SAGE-Mean	GAT
"Full-Graph"	.7029 ± .0036	.6982 ± .0038	.7097 ± .0035	.4475 ± .0107	.4810 ± .0081	.4048 ± .0125
NS-SAGE[1]	—	.7094 ± .0060	.7123 ± .0044	—	.4776 ± .0041	.3499 ± .0142
Cluster-GCN	.6805 ± .0074	.6976 ± .0049	.6960 ± .0062	.4068 ± .0096	.3486 ± .0216	.3905 ± .0152
GraphSAINT-RW	.7079 ± .0057	.6987 ± .0039	.7117 ± .0032	.4368 ± .0169	.3359 ± .0128	.3489 ± .0114
VQ-GNN (Ours)	.7055 ± .0033	.7028 ± .0047	.7043 ± .0034	.4316 ± .0134	.4673 ± .0164	.4102 ± .0099

[1] NS-SAGE is not compatible with GCN.

See our paper for performance results on more datasets, under inductive learning setups, and more ablation studies.

M. Ding*, K. Kong*, J. Li, C. Zhu, J. Dickerson, F. Huang, T. Goldstein. University of Maryland.

Thank you!

COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

Mucong Ding*, Kezhi Kong*, Jingling Li, Chen Zhu, John P Dickerson, Furong Huang, Tom Goldstein. University of Maryland.