Memory-Efficient Tensorized Embedding Layers for Neural Networks

Chunxing Yin and Richard Vuduc,
Georgia Institute of Technology

Georgia Tech

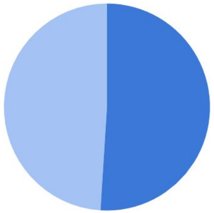
Outline

1. Motivation
2. Tensor Train Decomposition
3. Model Accuracy
    a. Recommendation model
    b. Graph neural network
4. Training time and performance

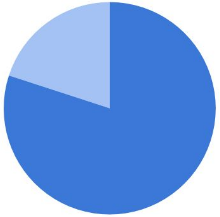
Embedding Layer

- **Deep Learning Recommendation System (DLRM)**

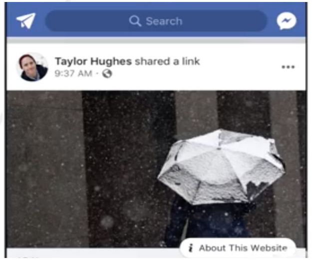
~50% of training



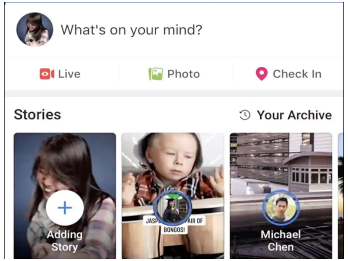
~80% of inference



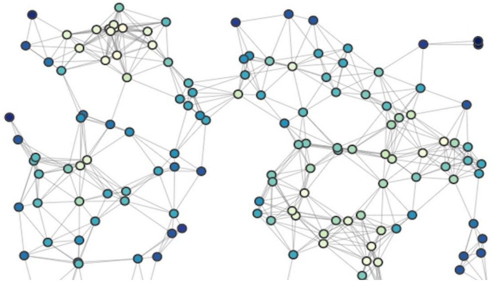
News Feed Ranking



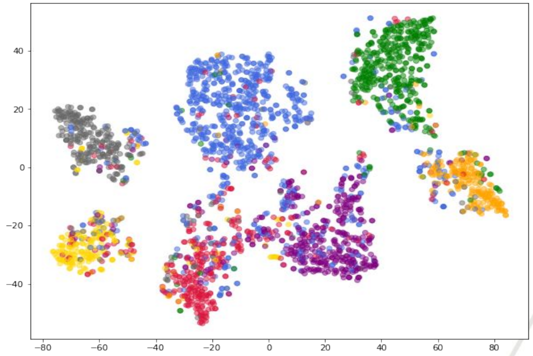
Stories Ranking



- **Graph Neural Network (GNN)**



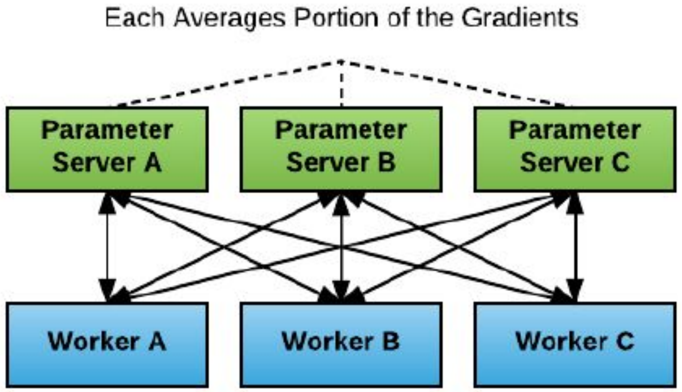
Example graph



Visualization of node embedding

Challenges

- Tens of GB to TB size of embedding table
- Distribute on multiple CPUs
- **80% time** spent in host-device communication [1]
- GPUs in distributed GNN are underutilized
- Parameter update computes on CPU



General Embedding Compression Technique [2]

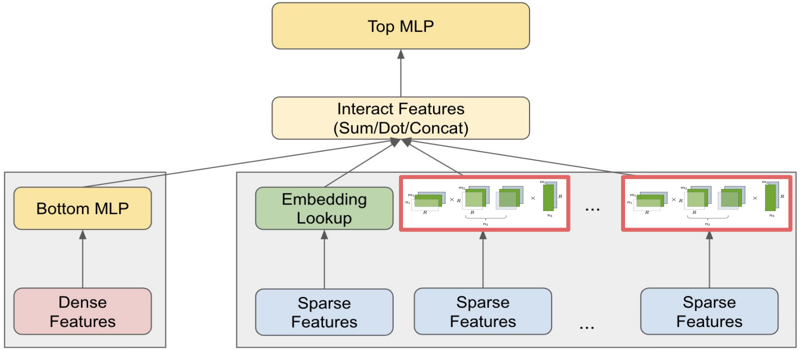
- Quantization
- Pruning
- Hashing

[1] Gandhi, Swapnil, and Anand Padmanabha Iyer. "P3: Distributed Deep Graph Learning at Scale." In *15th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 21)*, 2021.
[2] Gale, Trevor, Erich Elsen, and Sara Hooker. "The state of sparsity in deep neural networks." *arXiv preprint arXiv:1902.09574* (2019).

Tensorize Neural Network

- Replace full matrix/tensor parameters with a low-rank tensor decomposition
- A principled approach to compression

- In distributed GNN training
    - Enables data-parallelism
    - Reduce communication



Network	Task	Compr. Ratio	Accuracy Loss
Wide-ResNet [4]	Image	122x	2%
GRU [5]	Video	3000x	-120%
Transformer [6]	NLP	58.5x	4%

[4] Wang, Wenqi, et al. "Wide compression: Tensor ring nets." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.
[5] Yang, Yinchong, Denis Krompass, and Volker Tresp. "Tensor-train recurrent neural networks for video classification." *arXiv preprint arXiv:1707.01786* (2017).
[6] Khrulkov, Valentin, et al. "Tensorized embedding layers for efficient model compression." *arXiv preprint arXiv:1901.10787* (2019).

Georgia Tech

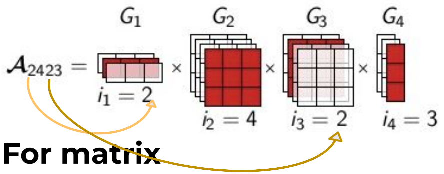
Background – Tensor Train Compression

Tensor Train (TT) decomposition factorize a tensor as a product of small tensors [7]

- For d-way tensor

  $$\mathcal{A}(i_1, i_2, \dots, i_d) = \mathcal{G}_1(:, i_1, :)\mathcal{G}_2(:, i_2, :)\dots\mathcal{G}_d(:, i_d, :)$$

  where G_k is a 3-way tensor of size **R_{k-1} x N_k x R_k**, and $R_0 = R_d = 1$. The sequence R_i is referred to as **TT-ranks**, and each tensor G_i is called a **TT-core**
- Small example*



- **For matrix**

$$\mathcal{W}((i_1, j_1), (i_2, j_2), \dots, (i_d, j_d)) = \mathcal{G}_1(:, i_1, j_1, :)\mathcal{G}_2(:, i_2, j_2, :)\dots\mathcal{G}_d(:, i_d, j_d, :)$$

TT-matrix example

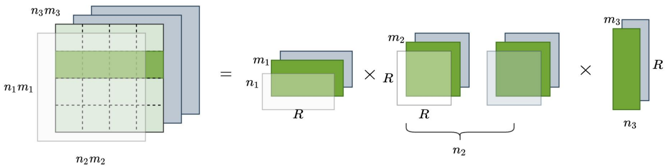
- Matrix W of size 5,000,000 x 24
- Factorize dimensions

  5,000,000 = 100 x 200 x 250,

  24 = 4 x 2 x 3
- Reshape W as a 6-way tensor

  ((100, 4), (200, 2), (250, 3))
- Decompose W using 3 TT-cores

  TT-core Shape:(1, 100, 4, R), (R, 200, 2, R),(R, 250, 3, 1)

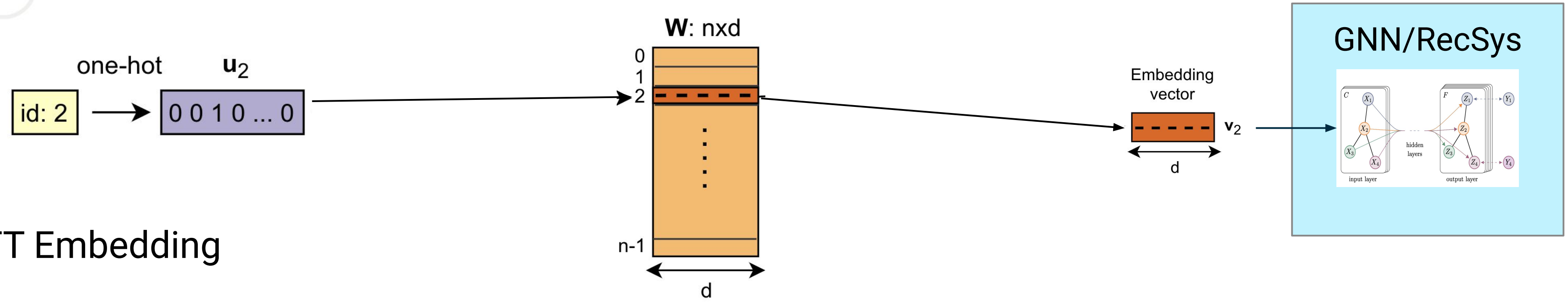


[7] Oseledets, Ivan V. "Tensor-train decomposition." SIAM Journal on Scientific Computing 33.5 (2011): 2295-2317.
* Novikov, Alexander. Tensor Train decomposition in machine learning. Powerpoint presentation.

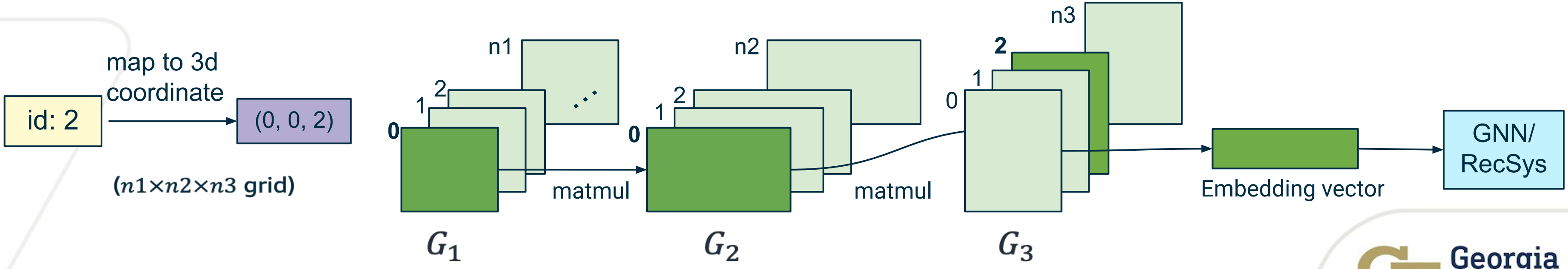
Georgia Tech

Model Overview

Full Embedding

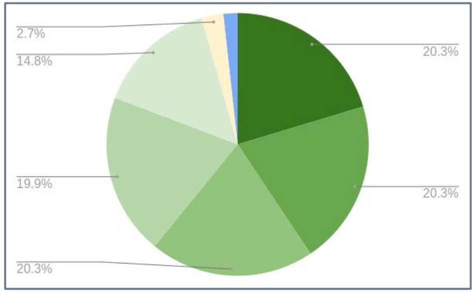


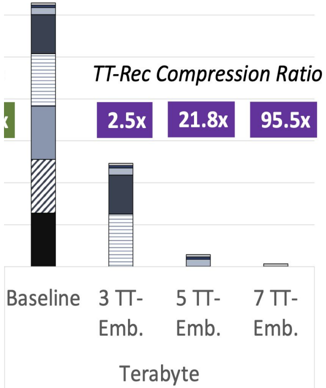
TT Embedding



Memory Reduction with TT

- Compress the largest 3 to 7 embeddings
- Single embedding table reduction up to 1200x
  - Store 10M x 16 emb. by 3 TT-cores:
    (1, 200, 2, R), (R, 200, 2, R), (R, 250, 4, 1)
- Overall model reduction ranges from 4x to 120x





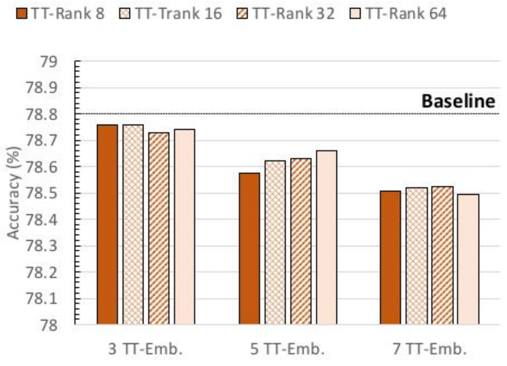
TeraByte Emb. Table Dimensions		Size (FP32)
9994222	64	2.56 GB
9980333	64	2.55 GB
9946608	64	2.55 GB
9758201	64	2.50 GB
7267859	64	1.86 GB
1333352	64	0.34 GB
Others		0.22 GB
Total		**12.58 GB**

Yin, Chunxing, Bilge Acun, Carole-Jean Wu, and Xing Liu. "Tt-rec: Tensor train compression for deep learning recommendation models." *Proceedings of Machine Learning and Systems* 3 (2021): 448-462.

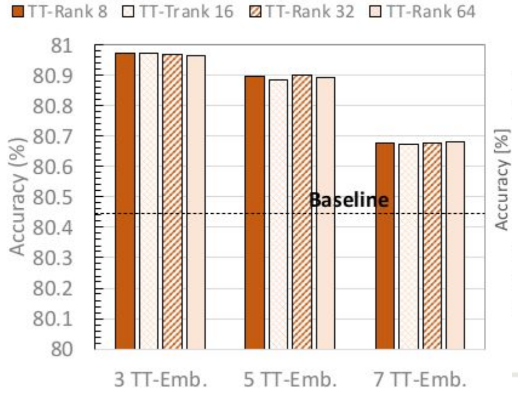
TT Model Quality

- With more emb. in TT format
  - Higher model reduction
  - Lower accuracy
  - For Kaggle, val. accuracy loss ranges from 0.03% to 0.3%
  - For Terabyte, TT-Rec outperforms baseline from 0.23% to 0.4%

* Note: Terabyte baseline have improved since the making of this plot.



(a) Kaggle



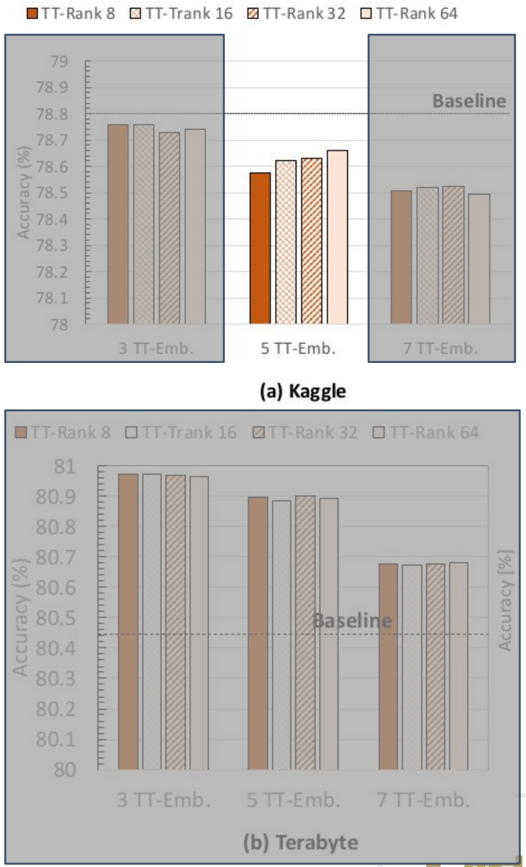
(b) Terabyte

TT Model Quality

- With more emb. in TT format
  - Higher model reduction
  - Lower accuracy
  - For Kaggle, val. accuracy loss ranges from 0.03% to 0.3%
  - For Terabyte, TT-Rec outperforms baseline from 0.23% to 0.4%

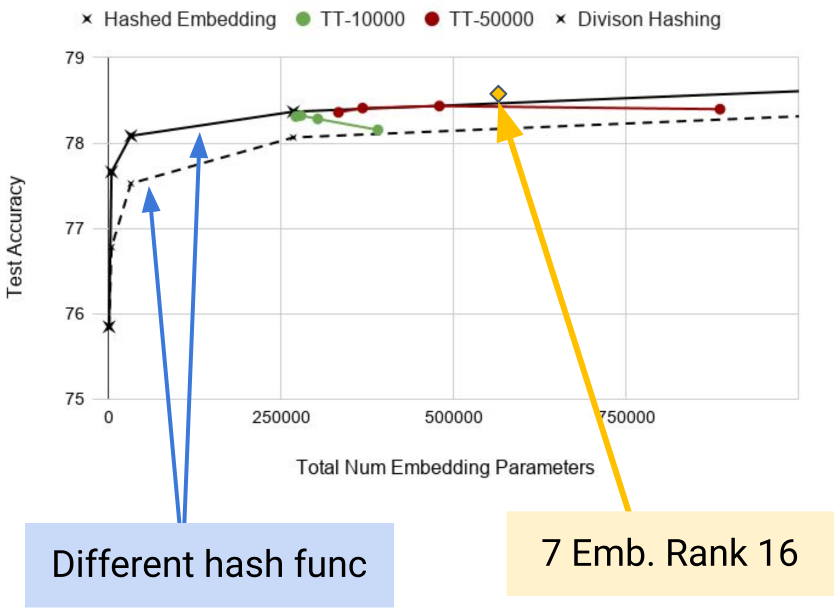
* Note: Terabyte baseline have improved since the making of this plot.

- Using larger TT-ranks produces more accurate model at the expense of lower compression ratio



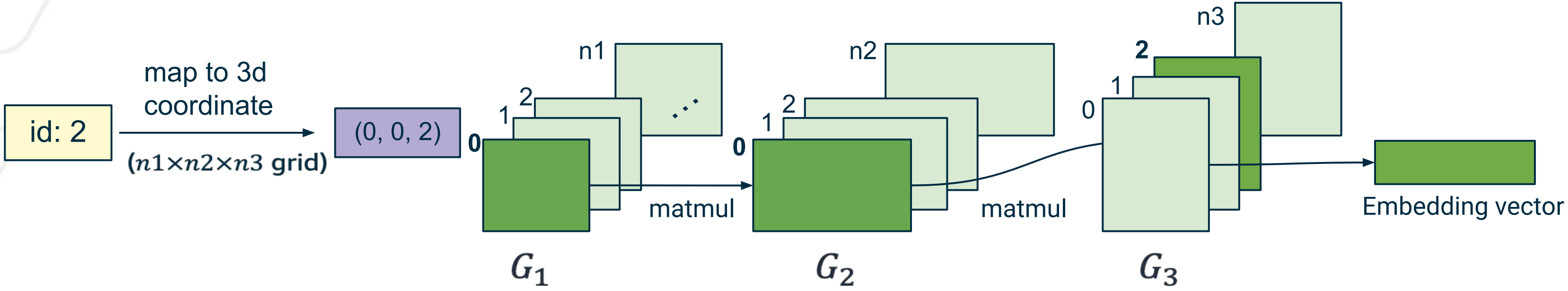
Advantages of Tensorized Embedding

- Low rank representation
  - Compression
  - Preserve accuracy
  - Robust to overfitting and noise
- Generate a unique vector for each item
  - More stable than hashing
- Implicit item grouping and weight sharing

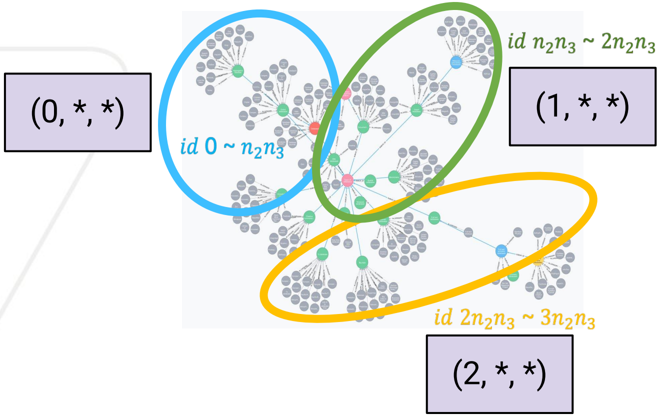


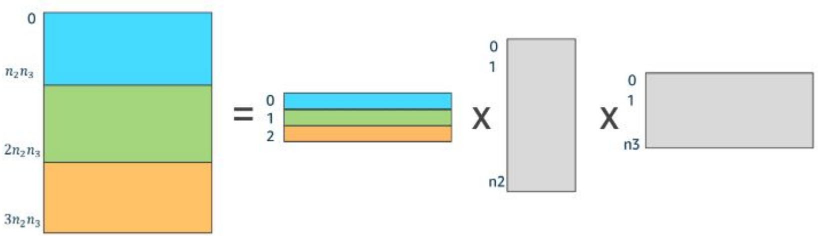
Connection to Graph

- TT-emb vector construction



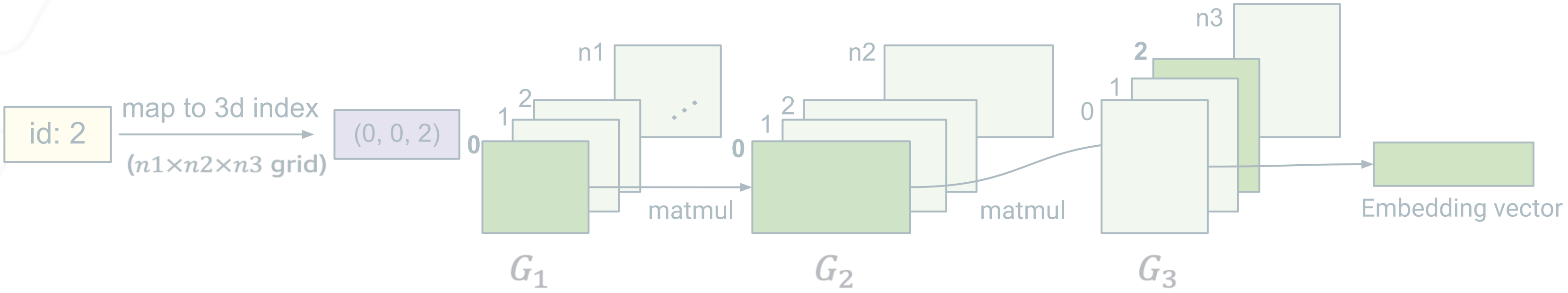
- Connection to graph topology



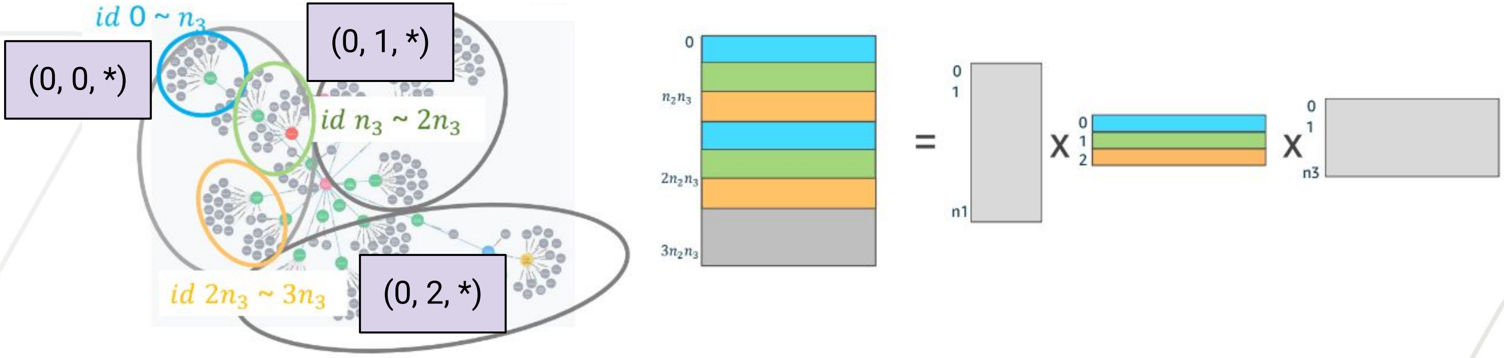


Connection to Graph

- TT-emb vector construction

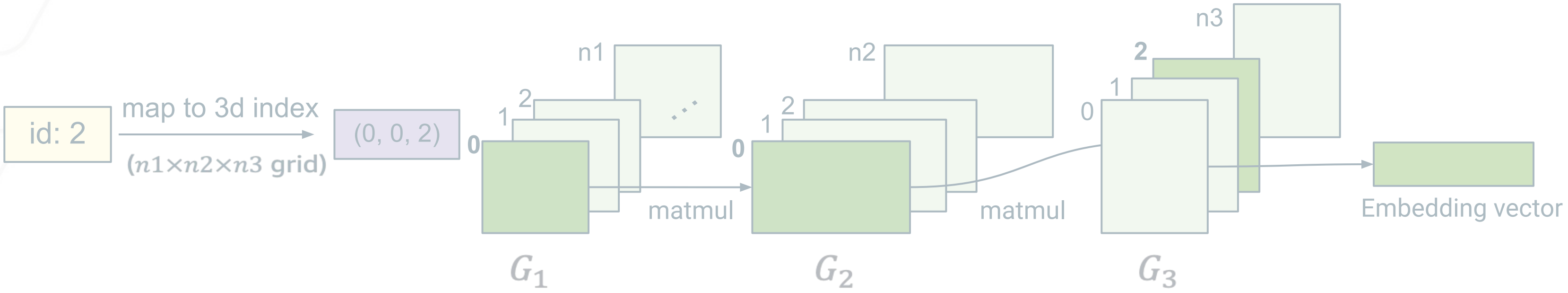


- Connection to graph topology

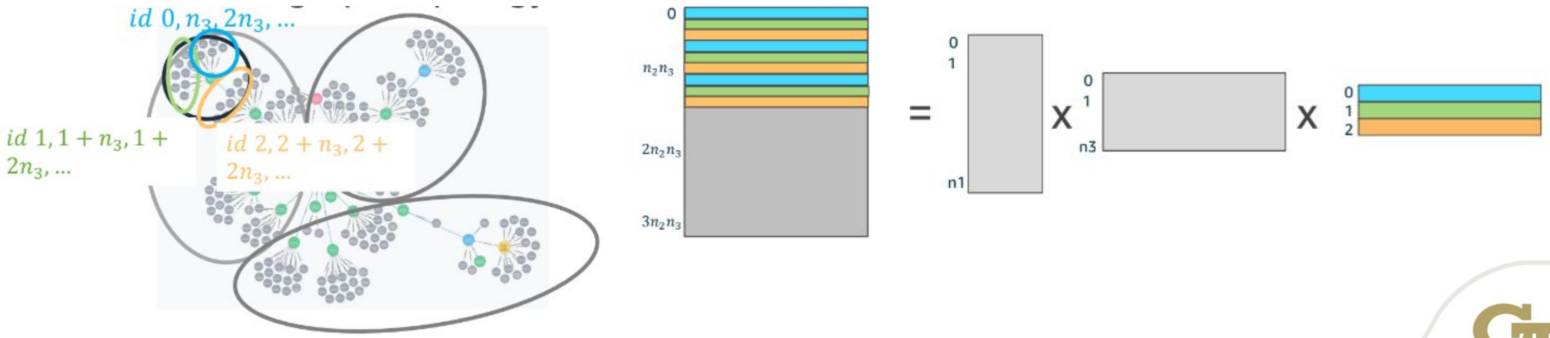


Connection to Graph

- TT-emb vector construction

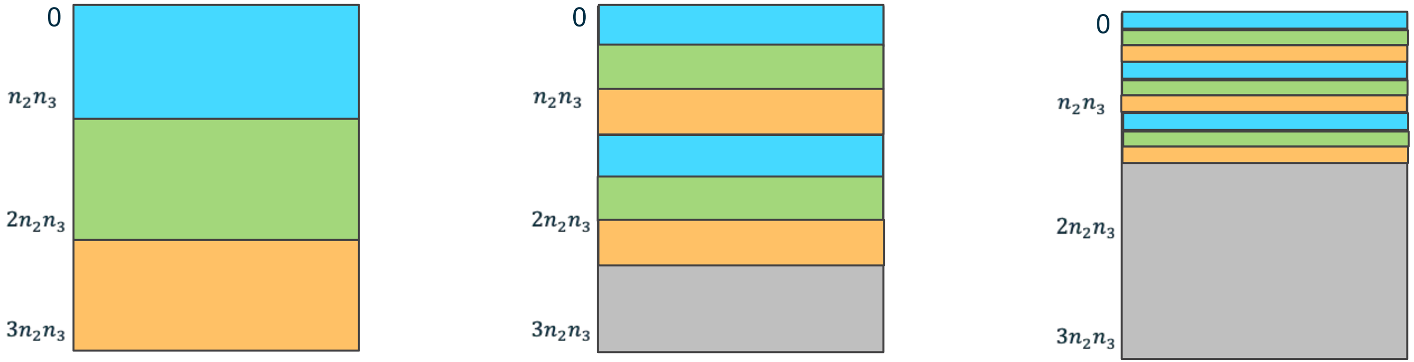


- Connection to graph topology



Connection to Graph

- Parameter sharing through node reordering
- TT-cores correspond to recursive graph partitioning
- Align TT structure with graph topology to produce homophily representation



Accuracy with Node Reordering

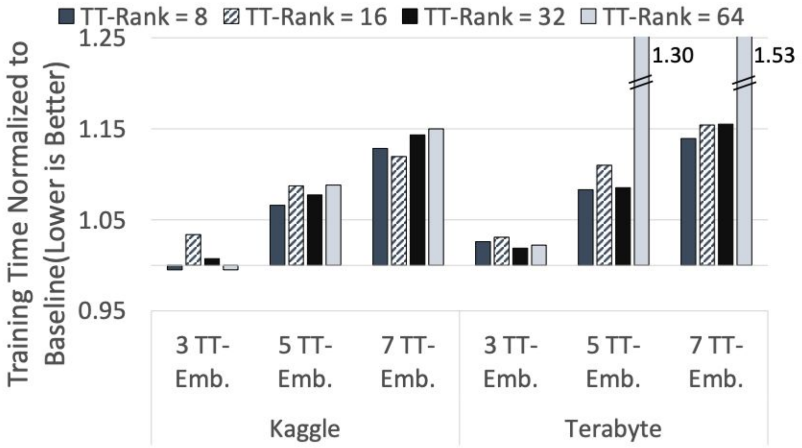
- ognb-products graph(2.5M nodes) trained with Graph Attention Network (GAT)
- Outperform the full embedding baseline
- Fine-grained graph partitioning helps improving model accuracy
- Produce better node embedding than the original dataset

Partition	0	4	16	256	800	1600	3200	**#param Reduction**
Full Emb	0.7485	-	-	-	-	-	-	1x
Rank 8	0.7448	0.745	0.748	0.7578	0.7535	0.7615	0.7628	5762x
Rank 16	0.7544	0.7629	0.7564	0.7681	0.7756	0.7713	0.7626	1580x
Rank 32	0.7632	0.7719	0.7671	0.7678	0.79	0.7647	0.78	415x

Training Time of DLRM

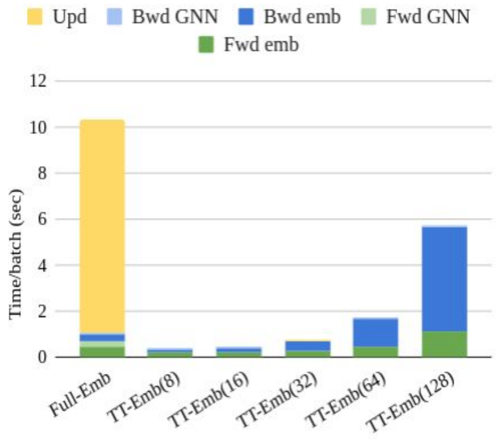
- Compare with Pytorch EmbeddingBag

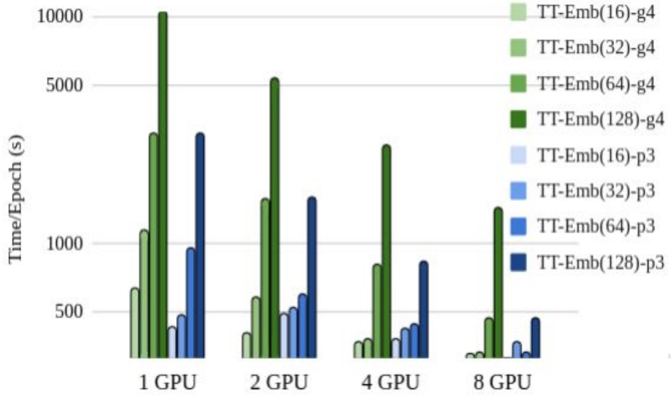
- Increase emb. in TT format from 3 to 7
  - Reduces the model size by 46.5 and 37.4x for Kaggle and Terabyte respectively
  - Increase training time by 12.5% for Kaggle, and 11.8% for Terabyte with the optimal TT-rank
- Higher model size reduction come with higher training time overheads



Training Time of GNNs

- Full Emb: 90% time spent on update
- 30% time for emb lookup, 60% time for emb backprop
- Reduce training time of ogbn-papers100M by 4.6X
- Scales almost linearly with large TT-ranks
- Hardware
  - AWS EC2 g4dn-metal
  - 8 T4 GPUs, 2x24 cores, 384GB RAM





Summary

- Applied Tensor-train decomposition to compress embedding layers in recommendation system and GNNs
- Compress the embedding tables by 100x and 424x for the 2 models while preserving/improving the model accuracy
- Combine TT with hierarchical graph partitioning to generate homophily embedding
- Efficient implementation of TT-emb kernel

facebookresearch / **FBTT-Embedding**