Mem-Rec: Memory Efficient Recommendation System

~2100x Compression & ~3x acceleration

Gopi Krishna Jha, Sameh Gobriel and Nilesh Jain

Intel Labs – Emerging Systems Labs

intel.

Recommendation Systems are the Cash Cow for many of our Platform Customers

37% of Revenue

23.7% BestBuy's Growth

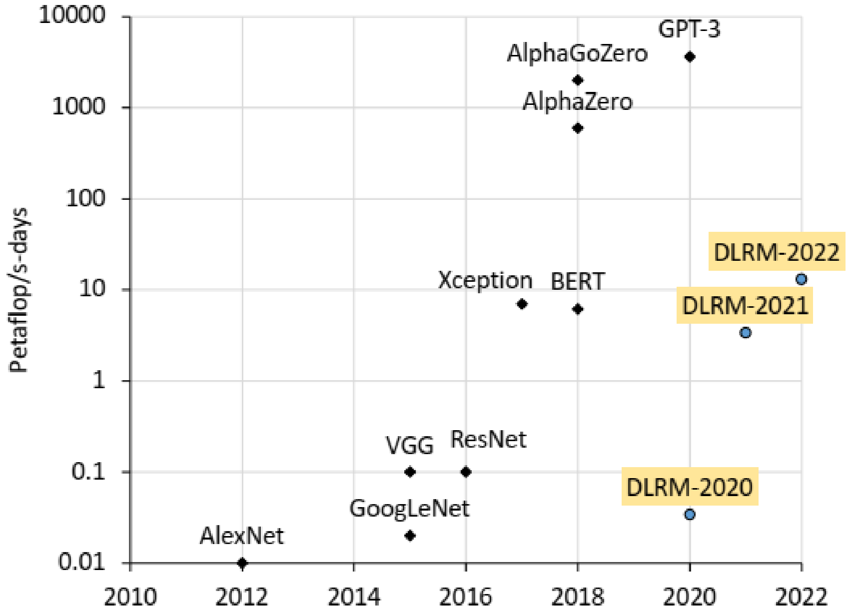
75% Netflix's Video consumption

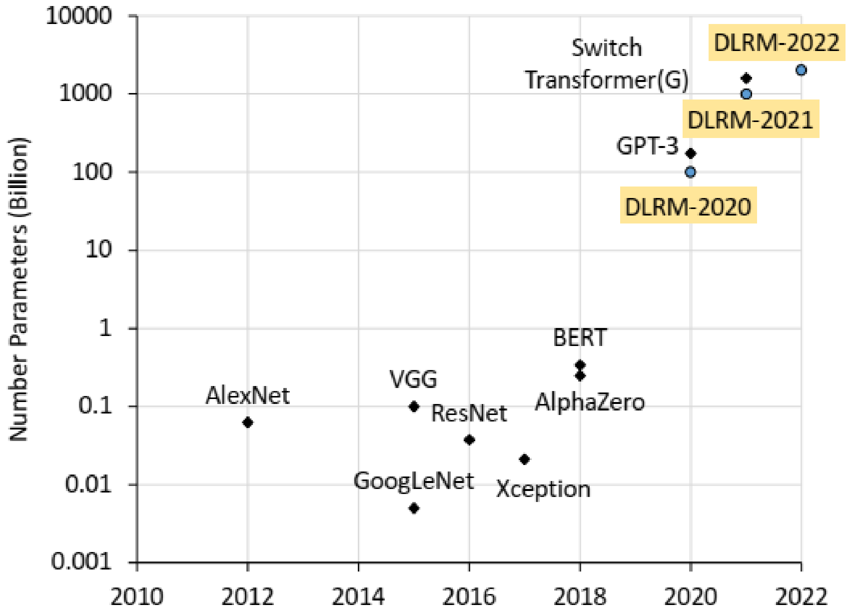
60% YouTube's views

... comes from their recommendation system

Production Scale DLRM are Extremely Large

- Latest FB papers are citing DLRM2022 at ~12 Trillion* Parameter model
- 99.9% of the parameters are coming from the Embedding tables



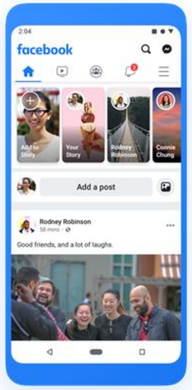


*https://arxiv.org/pdf/2104.05158v1.pdf

The Bottleneck: Embedding for Categorical Inputs

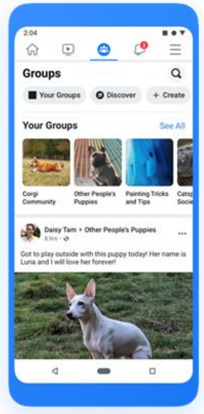
User Dense Features: Age, time of day, number of posts, ..etc.

User Categorical Features



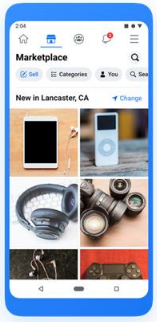
User Liked: F2, F345, F4095

Feed



User Member: G13, G45, G191

Groups



User bought: M123, M5

Marketplace

Size Proportional to alphabet

O(TB's)

Categorical (AKA Sparse) Features

Liked IDs: [..] | GroupIDs: [...] | ... | VideoIDs Watched: [...]

Embedding Table Lookup: ID to Dense Vector

Pool | Pool | ... | Pool

Aggregation (e.g. mean, ...)

Dense Vector
[0.121, 0.750, 0.579, ...]

Feed into Neural Net

Distributed Training of DLRM at FB

- Due to sheer size embedding tables are split across a cluster of specialized nodes*

Total compute	1+ PF/s
Total memory capacity	1+ TB
Total memory BW	100+ TB/s
Network injection BW per worker	100+ GB/s
Network bisection BW	1+ TB/s

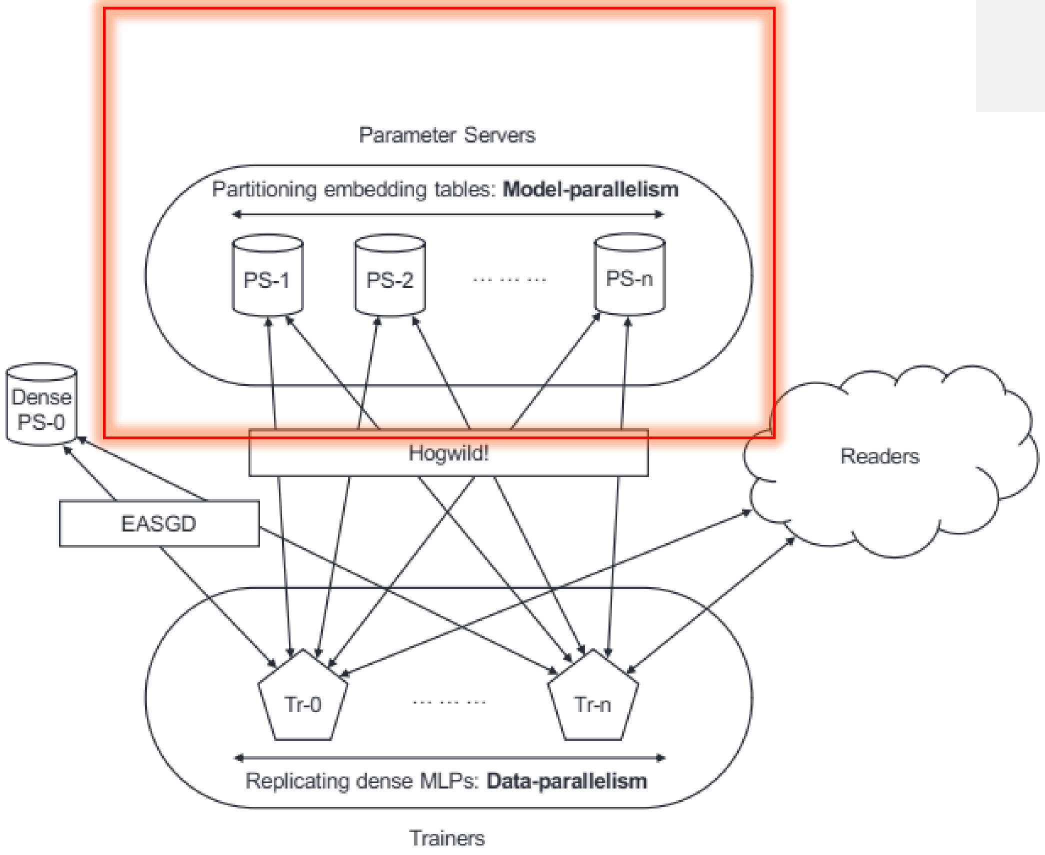


Figure 2: Disaggregated parameter-server based system

*https://arxiv.org/pdf/2104.05158v1.pdf

State of the Art Compression

1. Low-rank approximation
2. Weight Sharing
3. Hashing Trick
4. LSH-Based Compression
5. Double Hashing Trick
6. Lower Precision

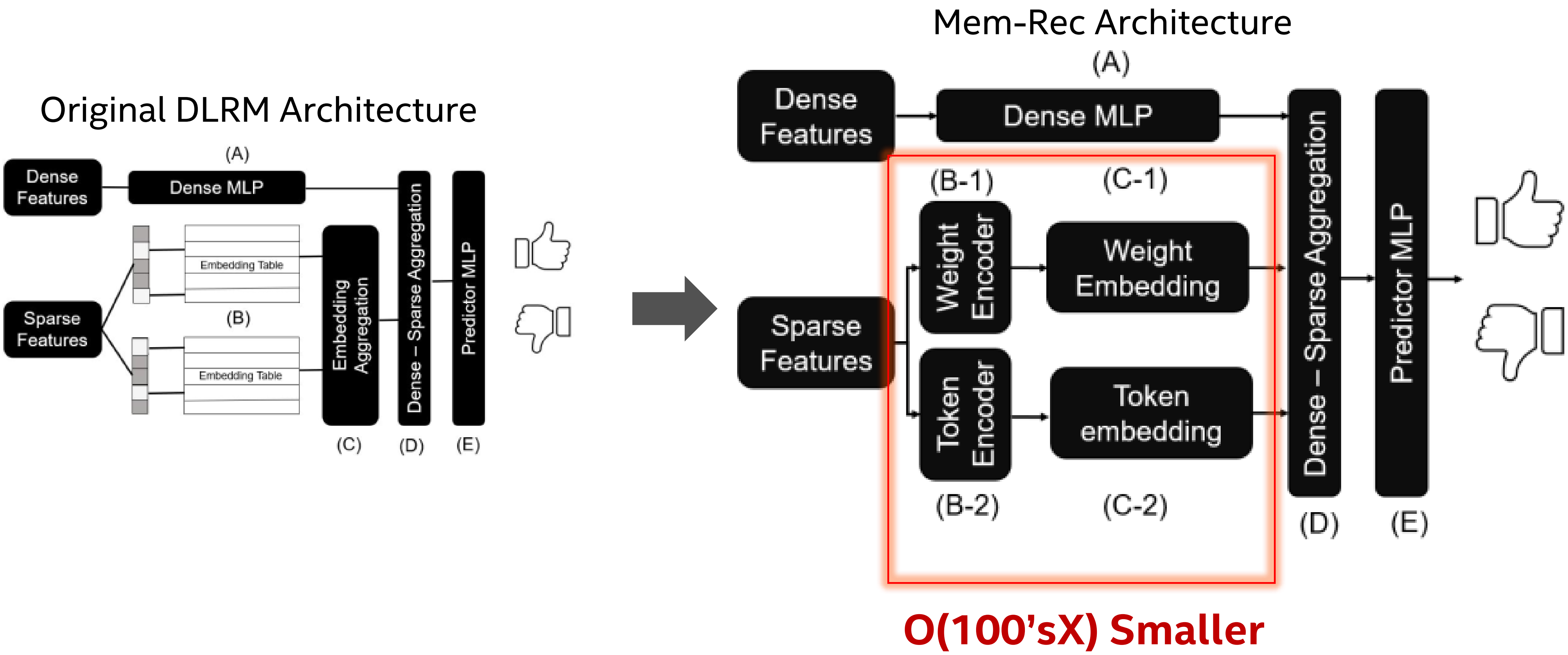
….

No Scheme Provided

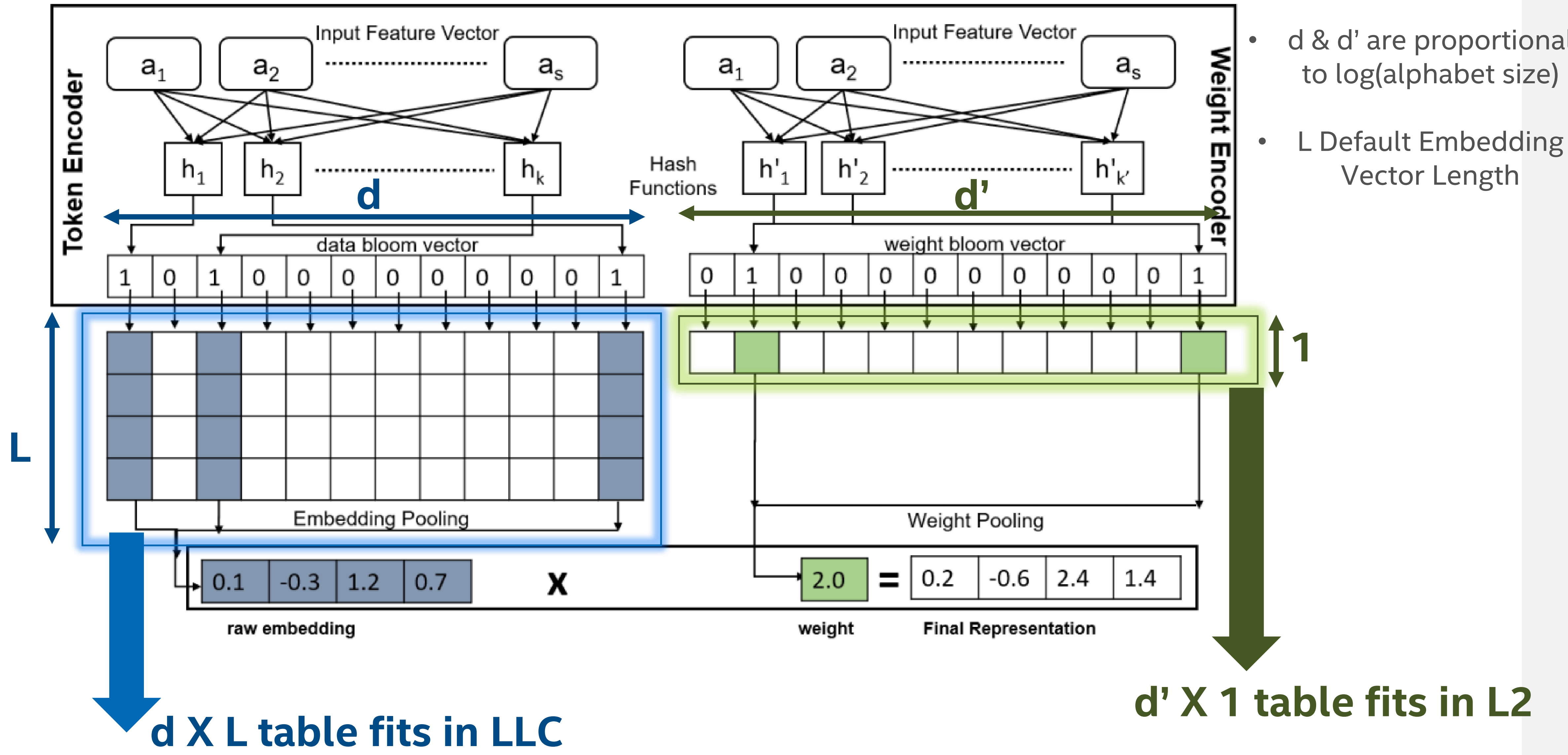
1. High compression Ratio (1000's X)
2. with Same AUC as full Model

Mem-Rec: Memory Efficient Recommendation System

Mem-Rec Architecture



Mem-Rec Sparse Feature Encoding



- d & d' are proportional to log(alphabet size)
- L Default Embedding Vector Length

Mem-Rec Compression Results

Parameters (Millions)	Model Size (MB)	ΔAUC vs DLRM	Compression
Criteo-TB			
5	20	-0.005	4734x
8	33	-0.002	2904x
11	46	0.000	2094x
15	59	0.001	1638x
21	84	0.000	1140x
Criteo-Kaggle			
2	9	-0.002	251x
4	15	0.000	144x
5	21	0.001	101x
7	28	0.001	78x
10	41	0.001	53x
Avazu			
0.8	3	0.000	188x
1.2	5	0.001	126x
1.6	6	0.002	95x
2.0	8	0.002	76x
2.4	10	0.002	63x

2094X Compression for Criteo TB Data with 0% loss in AUC compared to full uncompressed model

Technique	Compression (iso-quality)	Can fit in a 48MB L3 Cache
Criteo-TB		
ROBE	1000x	X
TT-REC	112x	X
MEM-REC	**2094x**	✓

Mem-Rec Hardware Bottleneck Analysis

LLC Size (MB)	14	28	56
num cycles	2.6x	3.2x	3.4x
num cache misses (LLC)	2.3x	6.3x	341x
average dram bandwidth	1.1x	2.2x	98x

DLRM vs MEM-REC
(MEM-REC parameters d&d' =75000, k =1, k'= 4, l = 128)

As LLC size grows → Mem-Rec LLC misses & DRAM BW shrink → **3.4X** better embedding time

We can fit DLRM w/TB Dataset on Client Devices

1. Low inference latency

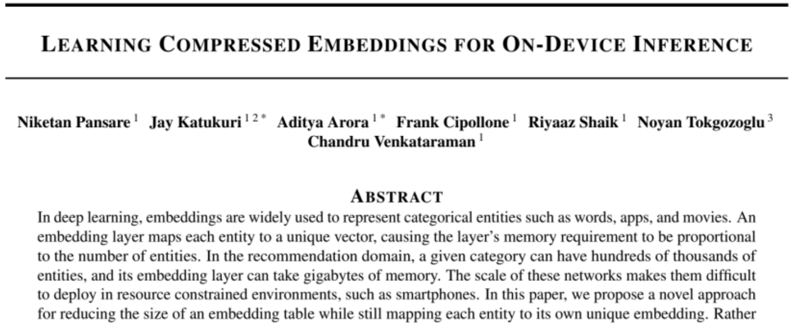
2. Privacy→ users' sensitive data need not be sent to the cloud

3. Reduces cost of hosting cloud-based recommender systems.

4. Fine-tuning or even training from scratch to best fit specific user preferences.

Example of the Trend

Apple - MLSys'22 Paper*



LEARNING COMPRESSED EMBEDDINGS FOR ON-DEVICE INFERENCE

Niketan Pansare [1] Jay Katukuri [1 2 *] Aditya Arora [1 *] Frank Cipollone [1] Riyaaz Shaik [1] Noyan Tokgozoglu [3]
Chandru Venkataraman [1]

ABSTRACT

In deep learning, embeddings are widely used to represent categorical entities such as words, apps, and movies. An embedding layer maps each entity to a unique vector, causing the layer's memory requirement to be proportional to the number of entities. In the recommendation domain, a given category can have hundreds of thousands of entities, and its embedding layer can take gigabytes of memory. The scale of these networks makes them difficult to deploy in resource constrained environments, such as smartphones. In this paper, we propose a novel approach for reducing the size of an embedding table while still mapping each entity to its own unique embedding. Rather

*https://proceedings.mlsys.org/paper/2022/file/812b4ba287f5ee0bc9d43bbf5bbe87fb-Paper.pdf

Mem-Rec Running on Client Platforms*

- DLRM with TB Criteo DataSet running on Alder Lake Platform
- **Unfeasible** to run full model due to large memory footprint required

Latency Per Item

Batch size	ADL-Client	TGL-Client	ICX-Server-DLRM
1	**0.135**	**0.076**	**0.327**
64	**0.007**	**0.006**	**0.010**
128	0.007	0.005	0.007
256	0.006	0.004	0.005
512	0.007	0.004	0.003
1024	0.005	0.004	0.002
16384	0.003	0.004	0.002

- Mem-Rec provides ~3X better inference latency
- Mem-Rec advantage will even be bigger when adding communication latency to backend server makes

*In collaboration with CCG: Vivek Kumar and Michael Rosenzweig

Conclusion and Next Steps

- Mem-Rec shows compelling results for Recommendation Systems
  - 2100X Compression Ratio for Criteo TB Dataset
  - 3X better embedding latency
  - Same AUC as uncompressed full model
- Recommendation Systems is an important workload for our customers.
- Mem-Rec running on
  - Xeon Platforms shows a potential for lower TCO deployment of recommendation systems.
  - Core platforms shows a potential for Low inference latency and offer privacy for users' sensitive data.

Call for Action: customer collaboration to demonstrate application-level benefit

Backup

Mem-Rec Embedding Latency

LLC Size (MB)	14	28	56
num cycles	2.6x	3.2x	3.4x
num cache misses (LLC)	2.3x	6.3x	341x
average dram bandwidth	1.1x	2.2x	98x

Mem-Rec provides ~3X better embedding encoding latency when compared to uncompressed full DLRM model

Mem-Rec On Edge Xeon Platforms

Batch size	ADL-Client	TGL-Client	ICX-Server	ICX-Server-DLRM
1	0.135	0.076	0.186	0.327
64	0.007	0.006	0.004	0.010
128	0.007	0.005	0.004	0.007
256	0.006	0.004	0.002	0.005
512	0.007	0.004	0.002	0.003
1024	0.005	0.004	0.001	0.002
16384	0.003	0.004	0.001	0.002

- Criteo TB Dataset is not embedding heavy (pooling factor =1)
- Latest FB Papers cites a pooling factor of 100 is more realistic