

# Exploring Heterogeneity-Aware Optimizations for Resource Efficient Edge Recommendation

Yerin Lee<sup>1</sup>, Gyudong Kim<sup>1</sup>, Eunjin Lee<sup>1</sup>, Jeff Zhang<sup>2</sup>, Young-Ho Gong<sup>3\*</sup>, Young Geun Kim<sup>1\*</sup>, Carole-Jean Wu<sup>4</sup>

<sup>1</sup>Korea University, {yerin\_lee, gyudong\_kim, eunjin\_lee, younggeun\_kim}@korea.ac.kr

<sup>2</sup>Arizona State University, jeffzhang@asu.edu; <sup>3</sup>Soongsil University, yhgong@ssu.ac.kr; <sup>4</sup>Meta AI, carolejean.wu@gmail.com

**Abstract**—Recommendation systems are widely deployed on edge devices to enable personalized user experience. While recommendation inference has traditionally been performed on centralized servers, recent advances in mobile SoCs have motivated a shift toward on-device execution. However, achieving efficient recommendation inference on edge devices remains challenging due to edge-specific execution characteristics and heterogeneity. In this paper, we characterize resource inefficiencies under realistic edge constraints and propose optimization strategies.

## I. INTRODUCTION

Recommendation systems are widely used to personalize user experience. In recommendation systems, deep learning recommendation models (DLRMs) have been widely adopted to capture complex interactions between user and item features.

Conventionally, recommendation inference has been executed on centralized servers due to the high computational and memory demands of DLRMs [1]. Recently, with the advancement of mobile SoCs, there have been growing pushes to execute recommendation inference locally on edge devices [2], [3]. Edge execution offers several benefits: it eliminates communication overhead, reduces server-side computation and energy burden, and preserves user privacy.

Despite the benefits, efficient on-device recommendation inference remains challenging due to edge-specific execution characteristics. Unlike server-side execution, edge execution is inherently request-driven: each query corresponds to a single user, and the batch size is constrained by application-level dynamics as well as limited DRAM capacity. Such a constrained batch size degrades cache locality and reduces arithmetic intensity, rendering both EMB lookups and MLPs memory-bound.

The challenge is further compounded by heterogeneity across both models and devices. Recommendation models vary widely in configurations, including embedding dimension, the number of EMBs, and MLP hidden dimension. Moreover, differences in device characteristics — such as cache capacity, memory bandwidth, and compute throughput — lead to substantially different performance even for the same workloads. This heterogeneity complicates static optimization and limits effective utilization of edge resources.

In this paper, we first characterize the resource inefficiencies of on-device recommendation inference under realistic edge constraints and present optimization strategies to improve recommendation performance.

\*Corresponding authors.

<sup>4</sup>Work done while at Arizona State University.

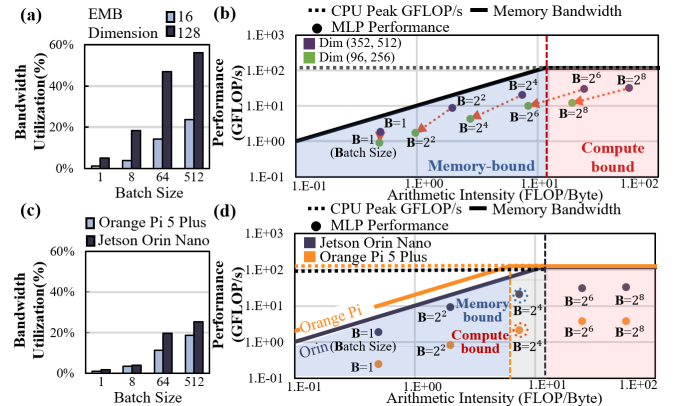


Fig. 1. (a) EMB lookup efficiency across embedding dimensions. (b) Roofline analysis of MLP across hidden dimensions. (c) EMB lookup efficiency across devices. (d) Roofline analysis of MLP across devices.

## II. CHARACTERIZATION OF ON-DEVICE RECOMMENDATION

### A. Edge-Specific Execution Characteristics

Naive execution of recommendation inference on edge devices often fails to fully exploit hardware resources, due to small and variable batch sizes on edge devices. To characterize the inefficiencies, we analyze the two dominant DLRM components under edge-specific execution constraints.

**EMB Lookups:** Fig. 1(a) shows the effective memory bandwidth utilization of the EMB lookups across various batch sizes and embedding dimensions, defined as the ratio of embedding vectors used during inference to those fetched from DRAM [4]. In lower batch sizes, limited memory-level parallelism and inherently irregular access patterns of EMB lookup degrade memory bandwidth utilization, making cache locality critical for mitigating memory latency. As batch size increases, memory bandwidth utilization improves but still fails to reach the peak, leaving room for resource utilization improvement. These results highlight an edge-specific inefficiency in EMB lookups that arises from dynamic batching and constrained caches.

**MLPs:** Fig. 1(b) shows a roofline analysis of an MLP layer on a Jetson Orin Nano across different hidden dimensions and batch sizes. At small batch sizes, limited batching fails to amortize weight-fetching costs, resulting in low arithmetic intensity and resource underutilization on edge devices. In large batch sizes, on the other hand, execution shifts from memory-bound to compute-bound, with arithmetic units becoming well utilized. These results indicate that the same MLP can exhibit different performance regimes depending on the serving context, making static execution strategies inadequate at the edge.

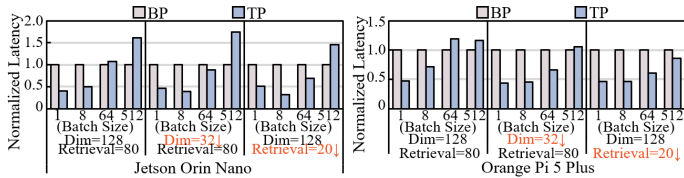


Fig. 2. BP-TP comparison across batch sizes & embedding dimensions.

### B. Impact of Heterogeneity

Heterogeneity in terms of cache capacity, memory bandwidth, and compute throughput exacerbates layer-level inefficiencies.

**Model Heterogeneity:** Heterogeneity in model parameters, such as embedding dimensions and MLP hidden dimensions, results in different performance characteristics even at a fixed batch size. Larger embedding dimensions (e.g., 128) achieve 2.3x higher average memory bandwidth utilization (up to 3.0x) than smaller ones (e.g., 16). Similarly, as shown in Fig. 1(b), larger hidden dimensions achieve 2.3x higher average performance (up to 3.0x) by crossing the ridge point and becoming compute-bound. These results show that model parameters can induce regime shifts in the dominant bottleneck.

**Device Heterogeneity:** Even with identical models and batch sizes, device heterogeneity leads to divergent resource efficiency. For EMB lookups, Fig. 1(c) shows that Jetson Orin Nano achieves 1.9x higher average memory bandwidth utilization (up to 2.1x) than Orange Pi 5 Plus. Fig. 1(d) further shows that at batch size 16 with an arithmetic intensity of 7.43, the same workload is memory-bound on Orin (ridge point 11.57) but compute-bound on the Orange Pi (ridge point 6.14). These results demonstrate that identical workloads can fall into different execution regimes across devices, thereby making static optimization insufficient to fully utilize edge resources.

## III. OPTIMIZATION STRATEGIES

As we observe in Section II, model and device heterogeneity cause varying resource inefficiency across DLRM layers. We address this via: 1) parallelization strategies for EMB to address resource underutilization; 2) a roofline-based analysis of recomputation and memory reuse trade-offs in top MLP layers.

### A. EMB lookup

To address batch size dependent cache locality characteristics, we leverage batch-level parallelism (BP) and table-level parallelism (TP), which exhibit different locality and access patterns. In BP, a batch (of rating items) is split into a number of threads that run in parallel across different cores. On the other hand, in case of TP, it assigns one or more EMBs to individual threads, enabling lookups for each EMB to be executed independently on different cores.

BP and TP exhibit distinct cache locality as batch size and embedding configurations vary. Fig. 2 shows normalized execution time of BP and TP on Jetson Orin Nano and Orange Pi 5 Plus across batch sizes. BP benefits from concurrent accesses to the same EMB, increasing opportunities for shared cache reuse across threads. In contrast, as the per-table working set size decreases due to smaller embedding dimensions or fewer retrievals per item, the crossover point shifts toward TP,

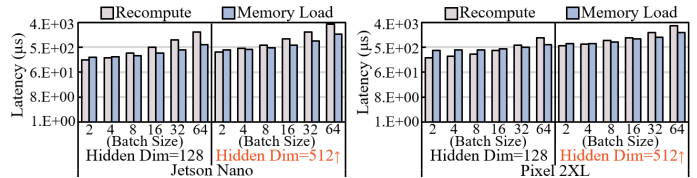


Fig. 3. Recompute-Memory load comparison on devices & hidden dimensions.

since each thread accesses a small set of vectors that fit well in private caches. This crossover point differs across devices due to differences in private and shared cache capacity.

### B. Top MLP

To address batch-size-driven computational regime shifts, we leverage input redundancy in top MLP. Since the user component in the first layer is identical across all rating items for a given query, we leverage memory load and recomputation. The memory load strategy reuses pre-computed user features through memory load, reducing redundant computation in compute-dominated regimes. In contrast, the recomputation strategy avoids additional memory traffic by recomputing the user contribution when memory access becomes the bottleneck, thereby better utilizing available compute resources.

The trade-off between recomputation and memory load depends on batch size, model parameters, and device-specific roofline characteristics. Fig. 3 compares the execution time of recomputation and memory load across different devices, hidden dimension of MLP, and batch sizes. Under bandwidth-limited conditions commonly observed at small batch sizes, recomputation can reduce memory pressure. As the MLP hidden dimension increases, higher arithmetic intensity shifts the crossover point between recomputation and memory load toward smaller batch sizes. This transition is further influenced by device-specific roofline characteristics, leading to different efficiency trends on the Jetson Nano and Pixel 2XL under identical workloads.

The above results demonstrate that execution efficiency is highly scenario-dependent across both EMB lookups and top MLP execution. This scenario dependence is driven by request-level dynamics, model characteristics, and device-specific constraints, motivating the importance of understanding such variability when employing existing execution strategies.

## IV. CONCLUSION

In this paper, we characterize resource inefficiencies in EMB lookups and MLP under edge constraints and show that the effectiveness of optimization strategies depends on batch size, model configurations, and hardware characteristics. These findings highlight the need for heterogeneity-aware execution strategies to achieve resource-efficient edge recommendation.

## ACKNOWLEDGMENT

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2023-00212711 and No. RS-2025-24534857), Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by MSIT (No. RS-2024-00398353), ICT Creative Consilience Program through IITP grant funded by the MSIT (IITP-2026-RS-2020-II201819).

## REFERENCES

- [1] M. Naumov, D. Mudigere, H.-J. M. Shi, J. Huang, N. Sundaraman, J. Park, X. Wang, U. Gupta, C.-J. Wu, A. G. Azzolini *et al.*, “Deep learning recommendation model for personalization and recommendation systems,” *arXiv preprint arXiv:1906.00091*, 2019.
- [2] C. A. Gomez-Uribe and N. Hunt, “The netflix recommender system: Algorithms, business value, and innovation,” *ACM Transactions on Management Information Systems (TMIS)*, vol. 6, no. 4, pp. 1–19, 2015.
- [3] F. Fessahaye, L. Perez, T. Zhan, R. Zhang, C. Fossier, R. Markarian, C. Chiu, J. Zhan, L. Gewali, and P. Oh, “T-recsys: A novel music recommendation system using deep learning,” in *2019 IEEE international conference on consumer electronics (ICCE)*. IEEE, 2019, pp. 1–6.
- [4] R. Hwang, T. Kim, Y. Kwon, and M. Rhu, “Centaur: A chiplet-based, hybrid sparse-dense accelerator for personalized recommendations,” in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020, pp. 968–981.