

MixEmb: Importance-Aware Quantization and Caching Co-Design for Industrial Decentralized Embedding Training

Zhida Jiang¹, Zhen Chen^{1*}, Zhaolong Xing^{1*}, Jiaying Wang¹, Saixing Gao¹,
Baopeng Yuan¹, Hao Wang², Yin Xu², Lin Liu¹, Dongyue Wang¹, Ke Zhang¹

¹JD.com

²University of Science and Technology of China

{chenzhen48, xingzhaolong1}@jd.com

Abstract

Industrial recommendation training increasingly relies on decentralized multi-GPU infrastructure, yet sparse embedding tables remain a major bottleneck due to significant storage, lookup, and communication overhead. Most existing works typically optimize only one of these bottlenecks or are designed for parameter server architectures, making them less effective in industrial decentralized training stacks. To tackle these challenges, we propose MixEmb, a practical framework for efficient decentralized embedding training through importance-aware quantization and caching. We first design a novel importance metric, *i.e.*, embedding score, which can distinguish the contribution of embedding vectors based on spatial heterogeneity and temporal dynamics. Guided by embedding scores, MixEmb employs mixed precision quantization and hierarchical caching at fine-grained vector level. Besides, our key finding is that quantization and caching decisions are tightly coupled. Instead of naive combination, we delicately integrate them into a unified training framework through coupled co-design. A theoretical convergence analysis is also provided for MixEmb. We implement and evaluate MixEmb in real-world training scenarios. Across three workloads, MixEmb attains up to $1.55\times$ speedup and $3.54\times$ compression ratio compared to state-of-the-art methods.

1 Introduction

Mirroring the breakthroughs of large language models (LLMs), both academia and industry handle numerous features to validate the scaling law in the recommendation domain (Yan et al., 2026). Despite the evolution of dense layers, recommendation systems (Behdin et al., 2025; Team, 2026; Yu et al., 2026) represent heterogeneous inputs (such as user queries, item descriptions, categorical attributes, and semantic IDs) as latent embedding

vectors. Sparse embedding tables remain a critical component to capture the semantic relationships between features (Chakraborty et al., 2025).

However, embedding training still encounters prohibitive resource overhead in industrial settings. **(1) Enormous Storage Requirements.** As the number and categories of features increase, the memory requirements of embedding tables may grow to unprecedented sizes, accounting for more than 99% of the total model capacity (Matam et al., 2024). **(2) Massive Lookup Overhead.** The activated indices retrieve the corresponding representation vectors from embedding tables (Zhang et al., 2025). Frequent memory accesses place significant pressure on CPU-GPU bandwidth, leading to I/O bottlenecks. **(3) Significant Communication Latency.** During forward and backward propagation, massive parameters require frequent communication to maintain consensus, which is further exacerbated by limited inter-node bandwidth.

Most prior studies improve embedding training efficiency for standalone architecture or parameter server (PS) architecture (Miao et al., 2025; Yang et al., 2020), which suffer from significant single-point bottlenecks. With recent advancements in hardware infrastructure and high-speed interconnects, there has been a shift towards fully decentralized peer-to-peer paradigm (Zhao et al., 2020). Mainstream industrial systems widely adopt decentralized multi-GPU architecture. Due to different training logic and communication patterns, existing PS-based methods cannot be deployed on decentralized multi-GPU architecture. Other solutions (Deng et al., 2021; Zhou et al., 2024; Li et al., 2023; Kurniawan et al., 2023; Xie et al., 2022; Song et al., 2023) only reduce one source of overhead while leaving the others largely unexplored. This motivates us to study the following question: *How can we simultaneously reduce storage, lookup, and communication overhead under decentralized multi-GPU architecture?*

*Corresponding Authors.

To tackle this problem, we propose MixEmb, an efficient decentralized training framework with importance-aware embedding quantization and caching. It is essential to identify the differences in convergence contributions of individual embedding vectors for fine-grained optimization. Compared to existing heuristic metrics considering single dimensions or static features, we design a dual-perspective importance metric, *i.e.*, *embedding score*, which captures spatial heterogeneity and temporal dynamics. Guided by this metric, MixEmb assigns different quantization levels to embedding vectors for mixed precision quantization, which reduces memory footprint and communication payload. Then, the majority of embedding vectors are stored and transmitted in low-precision format. In addition, MixEmb re-evaluates the utility of quantized parameters as training proceeds and dynamically adjusts caching hierarchy, thereby avoiding frequent CPU-GPU lookup operations.

More importantly, we highlight that quantization and caching decisions are tightly *coupled* and must be co-designed. The naive combination of existing embedding quantization and caching solutions will degrade training efficiency and accuracy (Yang et al., 2020), which is also empirically verified in Section 5. In comparison, MixEmb delicately unifies embedding quantization and caching into a single training framework through coupled co-design. The quantization levels align with cache hierarchy of embedding vectors, while cache replacement policy is closely tied to the quantization representation. To improve training stability, MixEmb further performs periodic cache updates instead of reacting to every cache miss. Overall, the main contributions of this paper are as follows:

- We propose an industrial-grade embedding training framework called MixEmb, which reduces storage, lookup, and communication overhead of decentralized architecture. We give the corresponding convergence analysis.
- We design a spatio-temporal embedding importance metric for cross-feature fairness comparison and dynamic importance evolution.
- We reveal the tight coupling between mixed precision quantization and hierarchical caching, and co-design them in a single training framework through joint optimization.
- Extensive experiments in real-world training scenarios demonstrate that MixEmb achieves up to $1.55\times$ speedup and $3.54\times$ compression ratio while maintaining comparable accuracy.

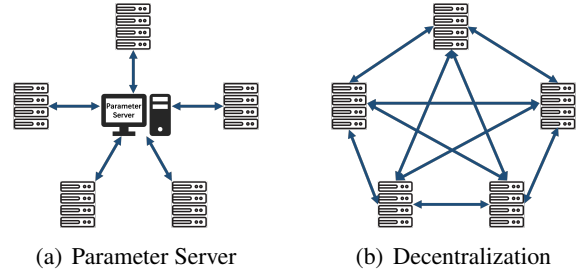


Figure 1: Different distributed training architectures.

2 Preliminaries

2.1 Industrial Training Setting

As shown in Figure 1, prior efforts rely on the PS architecture to scale embedding training. The CPU-hosted servers maintain global embedding tables while GPU-equipped workers process data batches (Miao et al., 2021). This star-shaped design suffers from scalability bottlenecks due to frequent transmission over bandwidth-limited networks.

Driven by advances in hardware infrastructure, mainstream industrial systems tend to deploy decentralized multi-GPU architecture (Zhao et al., 2020), which eliminates the dependency on the PS. Model parallelism partitions the embedding tables across GPU-equipped workers without overlap. Collective communication libraries enable GPU-GPU communication for synchronization. However, as the size of embedding tables experiences explosive growth, we still face prohibitive resource overhead, hindering efficient sparse training.

2.2 Limitations of Prior Works

Although some PS-based approaches (Miao et al., 2025; Zeng et al., 2024; Miao et al., 2021; Matam et al., 2024; Yang et al., 2020) have been proposed for efficient embedding training, their communication topology is always star-shaped and cannot be transferred to decentralized multi-GPU architecture commonly used in industrial systems. To alleviate the resource overhead, previous works have employed compression techniques, which are mainly divided into three categories: row-based compression methods (*e.g.*, hashing) (Zhang et al., 2020; Shi et al., 2020; Liu et al., 2025; Lai et al., 2023), column-based compression methods (*e.g.*, pruning) (Tran et al., 2025; Zhao et al., 2021; Luo et al., 2026; Lyu et al., 2022; Zhang et al., 2024), and value-based compression methods.

This paper falls within the value-based compression category, and is orthogonal to the row

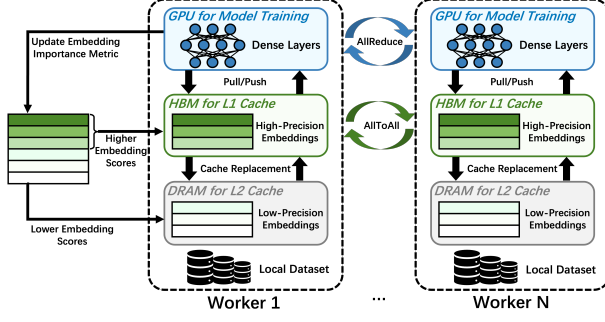


Figure 2: Overview of MixEmb.

and column-based methods. Among value-based methods, post-training quantization (PTQ) (Deng et al., 2021) and quantization-aware training (QAT) (Zhou et al., 2024; Li et al., 2024; Wang et al., 2025) cannot essentially reduce in-training resource consumption. Other works (Li et al., 2023; Xu et al., 2021) directly store parameters using low-precision formats during training, but they ignore heterogeneous embedding importance and adopt identical compression schemes for all embeddings. Besides, these methods require frequent CPU-GPU parameter movement. The significant I/O overhead still holds back the training efficiency.

To improve the lookup performance, a natural solution is to exploit data locality to store the working parameters in cache. Various caching policies (Kurniawan et al., 2023; Xie et al., 2022) focus on improving end-to-end inference performance rather than training. Other studies (Zhao et al., 2019, 2020; Wang et al., 2022; Zhang et al., 2022; Song et al., 2023) are proposed to maintain frequently used embeddings in GPU cache, eliminating the latency of memory access. Unfortunately, most caching schemes do not adopt any parameter reduction techniques. The massive embedding parameters incur exorbitant memory footprint and communication payload. In a nutshell, existing works fail to achieve efficient and scalable embedding training under *decentralized architecture*.

3 Methodology

3.1 Decentralized Training Orchestration

We propose an efficient embedding training framework termed MixEmb, tailored for the unique characteristics of industrial decentralized architecture. As shown in Figure 2, there are N workers collaborating to train in a peer-to-peer manner. Each worker performs mixed precision quantization and hierarchical caching at row granularity. Important embedding vectors reside on GPU HBM with high-

precision to preserve accuracy, while relatively unimportant embeddings are compressed into low-precision formats and offloaded to CPU DRAM.

During each iteration $t \in \{0, 1, \dots, T-1\}$, each worker n processes the current batch and splits keys into local and remote keys. Depending on the application, these keys may correspond to tokens, semantic IDs, or other categorical features derived from textual and behavioral inputs. Each worker retrieves high-precision embeddings indexed by local keys from GPU HBM. If a cache miss occurs, the worker reads low-precision embeddings from CPU DRAM. For remote embeddings belonging to other workers, MixEmb fetches quantized parameters through collective communication primitive AllToAll. Then, each worker receives all the embedding representations required by its own batch and performs forward and backward propagation. To ensure training consistency, MixEmb utilizes AllToAll communication to synchronize embedding parameters. Then, each worker will reassess the importance of updated embeddings based on training feedback (e.g., gradient norms) and periodically update hierarchical cache status. High-importance embeddings are retained or promoted to HBM, while low-importance entries are dumped to DRAM after precision down conversion.

3.2 Importance Evaluation Metric

The importance of embeddings exhibits a power-law distribution (Miao et al., 2021), where a small number of embeddings dominate the training process. The uniform embedding optimization ignores heterogeneous contributions and is unfair to important embeddings. Specifically, “non-hot” embeddings waste limited storage and communication resources, while “hot” embeddings are forced to compromise to insufficient information representation, ultimately resulting in poor recommendation performance. The skewed importance characteristic gives us the opportunity to develop differentiated schemes and allocate more resources to critical embeddings. Unlike existing metrics that rely on single dimensions or static statistics, we design a new spatio-temporal importance metric, *i.e.*, *embedding score*, to continuously evaluate individual embedding vectors using field-normalized access and gradient statistics.

Spatial Heterogeneity. The intensity of power-law distributions exhibits heterogeneity within and across feature fields. The access frequency reflects the popularity of intra-feature semantic represen-

tations. These parameters are frequently involved in model updates and contribute more to model quality. Moreover, gradient information reflects the magnitude of parameter updates. Larger gradient values dominate the optimization trajectory of embeddings and contribute more significantly to accuracy. Our design needs to consider gradient information to prevent aggressive quantization of critical embeddings from causing irreversible accuracy degradation. To this end, an embedding vector with higher access frequencies and/or larger gradient magnitudes indicates greater importance to training convergence. In addition to intra-feature embeddings, inter-feature semantic representations also show heterogeneity in importance. The different embedding dimensions and table sizes across features have an impact on the gradient norms and average access frequencies. It is misleading to directly compare the utility values of inter-feature embeddings (Lai et al., 2023). To ensure fair importance evaluations, each worker assigns a utility value U_i^t for i -th embedding vector $\mathbf{v}_i^t$ as follows:

$$\mu_i^t = \frac{1}{|B_i|} \sum_{m \in B_i} G_m^t A_m^t \quad (1)$$

$$U_i^t = \frac{G_i^t A_i^t - \mu_i^t}{\sqrt{\frac{1}{|B_i|} \sum_{m \in B_i} (G_m^t A_m^t - \mu_i^t)^2 + \epsilon}} \quad (2)$$

G_i^t , A_i^t , and B_i denote the L2-norm of gradients, access frequency, and feature field for $\mathbf{v}_i^t$, respectively. A small constant ϵ prevents division by zero. The rationale behind utility formulation is twofold. The numerator represents the *field relative importance* of embedding vector $\mathbf{v}_i^t$ compared to the mean within respective feature contexts, eliminating systematic biases. The denominator reflects the discriminability of current feature field and offsets scale discrepancies (e.g., embedding dimensions and table sizes) through normalization.

Temporal Dynamics. In real-world industrial scenarios, the popularity of specific feature representations changes dynamically as user preferences shift over time. An embedding vector that initially contributes to convergence may become less important as training progresses. If MixEmb cannot recalibrate the importance of embeddings in real time, irrelevant embeddings occupy excessive resources and introduce additional noise to training. However, focusing only on the latest statistics of embeddings lacks a global perspective. To strike the trade-off between responsiveness and stability,

we introduce a discount factor λ , which pays more attention to recent observations while preserving historical contribution patterns. In summary, the embedding score E_i^t for vector $\mathbf{v}_i^t$ can be calculated as:

$$E_i^t = \lambda E_i^{t-1} + U_i^t \quad (3)$$

It should be stressed that the embedding score plays a crucial role in our framework and demonstrates an implicit balance between spatial heterogeneity and temporal dynamics. The spatial perspective quantifies the importance by jointly considering the underlying logic of quantization and caching decisions. The utility values are further modulated to ensure fair comparisons within and across features. The temporal perspective adds an exponentially decaying factor to capture the training dynamics without overreacting to short-term fluctuations. This dual-perspective evaluation metric effectively distinguishes the embedding vectors that contribute to training and helps to align resource allocation with embedding importance.

3.3 Mixed Precision Quantization

Most compression schemes assign identical quantization levels to heterogeneous embedding vectors (Li et al., 2023). Guided by embedding scores, MixEmb assigns different quantization levels l_i^t to each vector $\mathbf{v}_i^t$ and performs mixed precision quantization. Due to hardware friendliness and computational efficiency (Li et al., 2023), MixEmb adopts symmetric quantization and calculates the scaling factor as:

$$s_i^t = \frac{\|\mathbf{v}_i^t\|}{2^{l_i^t-1} - 1} \quad (4)$$

where $\|\mathbf{v}_i^t\|$ is the L2-norm of embedding vector $\mathbf{v}_i^t$. For j -th element $\mathbf{v}_i^t(j)$ in embedding vector $\mathbf{v}_i^t$, the scaled value $|\mathbf{v}_i^t(j)|/s_i^t$ falls in the interval $[l, l+1]$, where l is an integer representing the quantized grid point. A higher quantization level l_i^t means a finer-grained grid so that the original values can be expressed more accurately.

Naively, one can round the scaled value to the nearest quantized grid point. Although deterministic rounding produces smaller errors in individual operations, the errors during training are not isolated. From the long-term perspective, intermediate values become persistently biased toward a certain direction, deviating the overall expectation from the true value. In contrast, stochastic strategy can be interpreted as introducing controlled noise during

training and statistically maintains the expectation of original values. To avoid the accumulated bias, we draw a random number p from a Bernoulli distribution with parameter $|\mathbf{v}_i^t(j)|/s_i^t - l$ to determine whether to round up or down:

$$\mathbf{r}_i^t(j) = \begin{cases} l + 1, & \text{if } p = 1 \\ l, & \text{otherwise} \end{cases} \quad (5)$$

After stochastic rounding, each scaled value can be encoded by $l_i^t - 1$ bits. We additionally need 1 bit for the sign and a full-precision number for the scaling factor s_i^t . Given a d -dimension embedding vector $\mathbf{v}_i^t$, the total number of bits to encode the quantized vector is $dl_i^t + 32$, which is much fewer than $32d$ bits representing the original values. According to the encoded information, the compressed values for $\mathbf{v}_i^t(j)$ is defined as:

$$Q(\mathbf{v}_i^t(j)) = \mathbf{r}_i^t(j) \cdot s_i^t \cdot \text{sign}(\mathbf{v}_i^t(j)) \quad (6)$$

where $\text{sign}(\cdot) = \{+1, -1\}$ is the sign of each element.

3.4 Quantization and Caching Co-Design

Considering the tightly coupled nature, MixEmb jointly optimizes mixed precision quantization and hierarchical caching rather than independent decisions. On the one hand, quantization levels will be aligned with the storage hierarchy. According to GPU cache size, top-ranked embeddings identified through embedding scores are allocated higher quantization levels, while other parameters are limited in bit width. On the other hand, cache management is driven by quantized training feedback rather than by cache misses alone. Conventional cache mechanisms perform replacement policy triggered by cache misses, evicting GPU-cached embeddings according to certain protocols (e.g., LRU/LFU) (Yang et al., 2020). However, due to the integration of quantization, the triggering time and conditions of cache replacement differ from prior methods. MixEmb delays replacement decisions until model updates. Each worker re-evaluates the utility of embeddings based on quantized representations and then determines cache location.

Furthermore, MixEmb uses periodic cache updates due to the inherent coupled nature. Since the quantization level is aligned with the cache location, short update interval P implies frequent adjustments to quantization levels, thereby inducing training noise. Periodic cache updating can

suppress parameter oscillations and enable progressive adaptation to quantization level, thus contributing to training stability, as empirically validated in Section 5. Besides, frequently sorting importance scores of massive embeddings is also computationally intensive, which potentially offsets the gains of our design. Periodic update mechanism can amortize algorithm overhead across multiple iterations.

4 Convergence Analysis

None of the existing studies provides a unified convergence analysis on embedding quantization and caching, which becomes more complicated under decentralized architecture. MixEmb aims to minimize the global loss function: $f^* = \min_{\mathbf{w}} [f(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N f_n(\mathbf{w})]$, where $f_n(\mathbf{w})$ is loss function of worker n . $\bar{\mathbf{w}}^t = \frac{1}{N} \sum_{n=1}^N \mathbf{w}_n^t$ is logical average model at iteration t and then the residual error is $\mathbf{R}_n^t = \mathbb{E} \|\mathbf{w}_n^t - \bar{\mathbf{w}}^t\|^2$. The error introduced by quantized parameters $\hat{\mathbf{w}}_n^t$ is defined by $\mathbf{Q}_n^t = \mathbb{E} \|\mathbf{w}_n^t - \hat{\mathbf{w}}_n^t\|^2$.

Theorem 1. Suppose each f_n is L -smooth and the stochastic gradients have bounded variance σ^2 . For any learning rate $\eta < \frac{1}{2L}$, we have the convergence bound:

$$\begin{aligned} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E} \|\nabla f(\bar{\mathbf{w}}^t)\|^2 &\leq \frac{2}{\eta T} (f(\bar{\mathbf{w}}^0) - f^*) \\ &+ \underbrace{\frac{2L^2}{NT} \sum_{t=0}^{T-1} \sum_{n=1}^N \mathbf{R}_n^t}_{\text{Residual Error}} + \underbrace{\frac{2L^2}{NT} \sum_{t=0}^{T-1} \sum_{n=1}^N \mathbf{Q}_n^t}_{\text{Quantization Error}} + 2L\eta\sigma^2 \end{aligned}$$

Remark. The residual error $\mathbf{R}_n^t$ and quantization error $\mathbf{Q}_n^t$ are related to caching and quantization decisions, respectively. We unify them into a single convergence bound, clarifying their impacts on training. MixEmb enables each worker to collect the cached parameters via AllToAll over a complete graph. Due to the sparse nature of feature embeddings, our design is *logically* equivalent to each worker maintaining an identical copy of model parameters, ensuring $\mathbf{R}_n^t = 0$ in a synchronous manner. Moreover, previous literature has empirically and theoretically proven that the quantization error $\mathbf{Q}_n^t$ has an upper bound (Reisizadeh et al., 2019). Ideally, if full-precision embeddings are stored and transmitted, $\mathbf{Q}_n^t$ decreases to 0. Then, we recover the convergence rate of fully synchronous training. The complete proof can be found in Appendix D.

Dataset	Avazu		Criteo		Industrial	
Metric	AUC	Speedup	AUC	Speedup	AUC	Speedup
PaddleBox	0.7523	1.00×	0.8072	1.00×	0.8015	1.00×
UGache	0.7523	1.17×	0.8072	1.20×	0.8015	1.27×
ALPT	0.7520	0.95×	0.8069	1.08×	0.8010	1.12×
MEC	0.7522	0.91×	0.8070	0.97×	0.8011	1.05×
UniEmb	0.7522	1.22×	0.8072	1.29×	0.8013	1.36×
MixEmb	0.7523	1.25×	0.8071	1.37×	0.8013	1.43×

Table 1: Overall training performance.

5 Evaluation

5.1 Experimental Setup

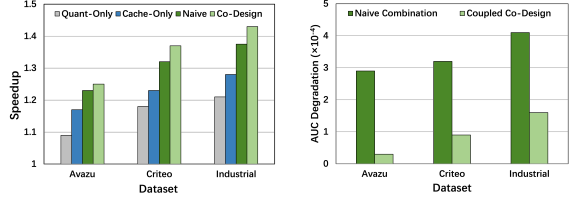
The experimental platform includes 16 NVIDIA A800 GPUs, 4 Intel Xeon Platinum 8338C CPUs, and 4TB of host memory. We conduct experiments over three real-world datasets, *i.e.*, *Avazu* (ava, 2014), *Criteo* (kag, 2014), and *Industrial*. Industrial dataset is sampled from a real-world e-commerce platform. The embedding dimension is 128 by default. For the Avazu and Criteo datasets, the default GPU cache size is set to 20% of the total embedding table size, while that of the Industrial dataset is 10%. Based on our sensitivity analysis of preliminary experiments, the cache update interval P is set to 10, and the discount factor λ is configured as 0.98. We employ two quantization levels for the embedding vectors, *i.e.*, 16 and 8.

5.2 Baselines

Since MixEmb is designed for decentralized architecture that mainstream industrial systems adopt, we exclude PS-based methods whose communication patterns cannot be transferred to the peer-to-peer setting. The row-based (*e.g.*, hashing) and column-based (*e.g.*, pruning) compression methods are omitted because they are orthogonal to MixEmb. Under these circumstances, we compare MixEmb with the following baselines: PaddleBox (Zhao et al., 2020), UGache (Song et al., 2023), ALPT (Li et al., 2023), MEC (Wang et al., 2025), and UniEmb, where UniEmb is a modified version of MixEmb and its core design is similar except that all embedding vectors are assigned the uniform quantization levels.

5.3 Overall Performance

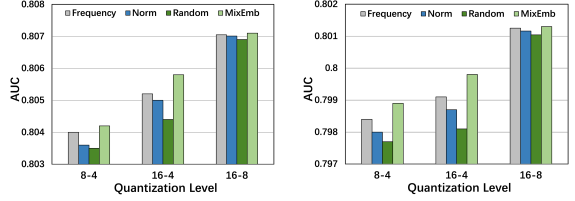
We summarize the final model accuracy and the average training speed to achieve this accuracy in Table 1, where MixEmb significantly accelerates the decentralized training process without compromising accuracy. *Firstly*, MixEmb consistently attains comparable accuracy to the original model.



(a) Training Efficiency

(b) Model Performance

Figure 3: Performance comparison of independent decisions, naive combination and coupled co-design.



(a) Criteo

(b) Industrial

Figure 4: Model accuracy under different importance metrics and quantization levels.

Compared to PaddleBox and UGache, MixEmb achieves 3.25-3.54 $\times$ compression ratio and 1.07-1.43 $\times$ speedup. *Secondly*, although ALPT and MEC may provide a high compression ratio, this does not translate into actual training speedup. For example, the training speed of ALPT is only 76-79% of MixEmb. Moreover, aggressive compression in these methods inevitably hurts model accuracy. *Thirdly*, compared with UniEmb, MixEmb further improves the end-to-end training efficiency by performing fine-grained embedding optimization. UniEmb determines the identical quantization levels for all embeddings regardless of their heterogeneous contributions.

5.4 Ablation Study

To indicate the advantages of core components, we compare the training efficiency and model accuracy of only cache, only quantization, naive combination, and our proposed framework. As illustrated in Figure 3(a), MixEmb always converges faster than independent and naive methods over three datasets, and improves the training speed by 1.35 $\times$ on average. Figure 3(b) plots the degree of AUC degradation compared with the original model. We find that naive combination methods exhibit an obvious decrease in the final model performance. The above findings empirically confirm that coupled co-design is necessary and can unleash the potential of quantization and caching through interactive decision-making.

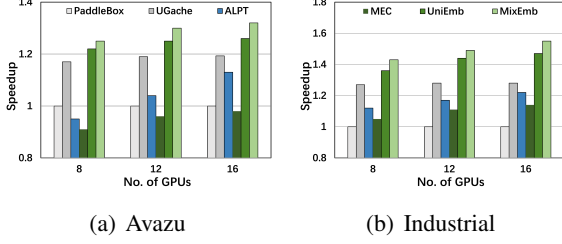


Figure 5: Training efficiency under different scales.

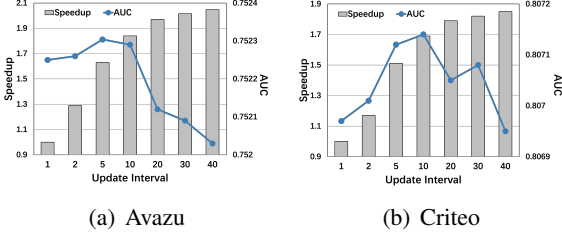


Figure 6: Training performance under different cache update intervals.

5.5 Effectiveness of Embedding Score

We replace different importance metrics while keeping other designs unchanged. Figure 4 depicts the model accuracy under combining various quantization levels. As the precision decreases, all methods suffer from performance degradation. Nevertheless, MixEmb consistently outperforms other metrics among different workloads and demonstrates more pronounced advantages under extreme quantization. For instance, under the quantization levels of 8-4, MixEmb attains the AUC score of 0.7989 for Industrial, while that of *Frequency*, *Norm*, and *Random* is only 0.7984, 0.7980, and 0.7977, respectively. These results emphasize the effectiveness of our proposed importance metric.

5.6 Scalability

We proceed to conduct a scalability study across different numbers of GPUs. Figure 5 shows that our proposed framework offers a clear training speed advantage for varying numbers of GPUs. Furthermore, the performance improvement becomes increasingly significant when deploying in more workers. For instance, as the system scale expands, the training speedup brought by MixEmb rises from $1.43\times$ to $1.55\times$ for Industrial dataset. The explanation for this phenomenon is that GPUs exchange compressed representations to alleviate the communication overhead associated with massive embedding vectors. These experimental results indicate that MixEmb can effectively enhance the efficiency of decentralized embedding training.

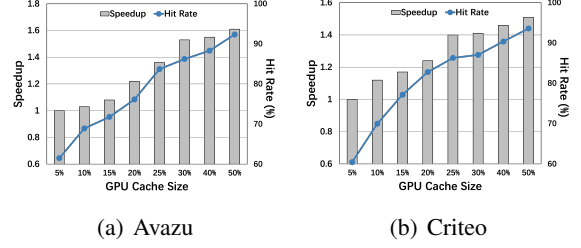


Figure 7: Hit rate and training efficiency under different GPU cache sizes.

5.7 Cache Update Interval

We conduct experiments to analyze the impact of cache update interval P on training performance. As shown in Figure 6, an increased interval can reduce the latency of cache update procedure, achieving up to $2.05\times$ speedup for Avazu and $1.85\times$ speedup for Criteo. Another interesting observation is that updating too frequently may lead to suboptimal convergence performance, while delaying the cache updates even slightly boosts model accuracy. The underlying reason is that periodic update mechanism adjusts the cache and quantization status after the importance distribution becomes stable, which mitigates parameter oscillations.

5.8 Impact of Cache Size

We investigate how MixEmb performs under different GPU cache sizes. As depicted in Figure 7, the hit rate increases from 60.5% to 93.6% with cache size varying from 5% to 50% for Criteo dataset. Consequently, we observe that MixEmb provides up to $1.61\times$ speedup for Avazu and $1.51\times$ speedup for Criteo. This is because more important embeddings can be directly accessed from GPU HBM without fetching from CPU DRAM, thus reducing considerable lookup overhead. The marginal benefit diminishes as the GPU cache expands. This phenomenon aligns with the inherent power-law distribution of categorical features.

6 Conclusion

We design and implement an embedding training framework, called MixEmb, tailored for the unique characteristics of industrial decentralized architecture. We introduce an embedding importance metric considering spatial heterogeneity and temporal dynamics. Guided by embedding scores, MixEmb coordinates mixed precision quantization and hierarchical caching. Extensive experiments suggest that MixEmb significantly accelerates embedding training without compromising accuracy.

Limitations

While MixEmb delivers substantial efficiency improvements for industrial decentralized embedding training, it remains inherently constrained by synchronous training paradigms with full GPU-GPU connectivity. Our design does not address the unique challenges of asynchronous training scenarios, where parameter consistency guarantees are weaker. Future efforts could extend MixEmb to support these distributed paradigms by adapting the quantization and caching strategies to tolerate partial synchronization. In addition, under a non-complete graph, remote embeddings would require multi-hop routing, neighbor-based propagation, or periodic global synchronization. In these cases, MixEmb may preserve synchronous semantics but incur additional communication latency, which varies with specific network topologies.

References

2014. Click-through rate prediction. <https://www.kaggle.com/c/avazu-ctr-prediction>.
2014. Display advertising challenge. <https://www.kaggle.com/c/criteo-display-ad-challenge>.
- Kayhan Behdin, Ata Fatahi Baarzi, Qingquan Song, Yun Dai, Aman Gupta, Zhipeng Wang, Hejian Sang, Shao Tang, Gregory Dexter, Sirou Zhu, Siyu Zhu, Tejas Dharamsi, Vignesh Kothapalli, Zhoutong Fu, Yihan Cao, Pin-Lun Hsu, Fedor Borisjuk, Natesh S. Pillai, Luke Simon, and Rahul Mazumder. 2025. *Scaling down, serving fast: Compressing and deploying efficient llms for recommendation systems*. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025 - Industry Track, Suzhou, China, November 4-9, 2025*, pages 1687–1702. Association for Computational Linguistics.
- Shramona Chakraborty, Shashank Mujumdar, Nitin Gupta, Sameep Mehta, Ronen Kat, Itay Etelis, Mohamed Mahameed, Itai Guez, and Rachel Tzoref-Brill. 2025. *Group, embed and reason: A hybrid LLM and embedding framework for semantic attribute alignment*. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025 - Industry Track, Suzhou, China, November 4-9, 2025*, pages 1703–1710. Association for Computational Linguistics.
- Li Chen, Wei Liu, Yunfei Chen, and Weidong Wang. 2024. *Communication-efficient design for quantized decentralized federated learning*. *IEEE Trans. Signal Process.*, 72:1175–1188.
- Zhaoxia Deng, Jongsoo Park, Ping Tak Peter Tang, Haixin Liu, Jie Yang, Hector Yuen, Jianyu Huang, Daya Shanker Khudia, Xiaohan Wei, Ellie Wen, Dhruv Choudhary, Raghuraman Krishnamoorthi, Carole-Jean Wu, Nadathur Satish, Changkyu Kim, Maxim Naumov, Sam Naghshineh, and Mikhail Smelyanskiy. 2021. *Low-precision hardware architectures meet recommendation model inference at scale*. *IEEE Micro*, 41(5):93–100.
- Daniar Heri Kurniawan, Ruipu Wang, Kahfi S. Zulkifli, Fandi A. Wiranata, John Bent, Ymir Vigfusson, and Haryadi S. Gunawi. 2023. *Evstore: Storage and caching capabilities for scaling embedding tables in deep recommendation systems*. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2, ASPLOS 2023, Vancouver, BC, Canada, March 25-29, 2023*, pages 281–294. ACM.
- Fan Lai, Wei Zhang, Rui Liu, William Tsai, Xiaohan Wei, Yuxi Hu, Sabin Devkota, Jianyu Huang, Jongsoo Park, Xing Liu, Zeliang Chen, Ellie Wen, Paul Rivera, Jie You, Chun-cheng Jason Chen, and Mosharaf Chowdhury. 2023. *Adaembed: Adaptive embedding for large-scale recommendation models*. In *17th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2023, Boston, MA, USA, July 10-12, 2023*, pages 817–831. USENIX Association.
- Shiwei Li, Huifeng Guo, Lu Hou, Wei Zhang, Xing Tang, Ruiming Tang, Rui Zhang, and Ruixuan Li. 2023. *Adaptive low-precision training for embeddings in click-through rate prediction*. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pages 4435–4443. AAAI Press.
- Shiwei Li, Zhuoqi Hu, Fuyuan Lyu, Xing Tang, Haozhao Wang, Shijie Xu, Weihong Luo, Yuhua Li, Xue Liu, Xiuqiang He, and Ruixuan Li. 2024. *Mixed-precision embeddings for large-scale recommendation models*. *CoRR*, abs/2409.20305.
- Zirui Liu, Hailin Zhang, Boxuan Chen, Zihan Jiang, Yikai Zhao, Yangyu Tao, Tong Yang, and Bin Cui. 2025. *CAFE+: towards compact, adaptive, and fast embedding for large-scale online recommendation models*. *ACM Trans. Inf. Syst.*, 43(3):61:1–61:42.
- Qinyi Luo, Penghan Wang, Wei Zhang, Fan Lai, Jiachen Mao, Xiaohan Wei, Jun Song, Wei-Yu Tsai, Shuai Yang, Yuxi Hu, and Xuehai Qian. 2026. *FIITED: fine-grained embedding dimension optimization during training for recommender systems*. *IEEE Trans. Computers*, 75(2):448–461.
- Fuyuan Lyu, Xing Tang, Hong Zhu, Huifeng Guo, Yingxue Zhang, Ruiming Tang, and Xue Liu. 2022. *Optembed: Learning optimal embedding table for click-through rate prediction*. In *Proceedings of the 31st ACM International Conference on Information*

- & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022, pages 1399–1409. ACM.
- Kiran Kumar Matam, Hani Ramezani, Fan Wang, Zeliang Chen, Yue Dong, Maomao Ding, Zhiwei Zhao, Zhengyu Zhang, Ellie Wen, and Assaf Eisenman. 2024. [Quickupdate: a real-time personalization system for large-scale recommendation models](#). In *21st USENIX Symposium on Networked Systems Design and Implementation, NSDI 2024, Santa Clara, CA, April 15-17, 2024*, pages 731–744. USENIX Association.
- Xupeng Miao, Hailin Zhang, Yining Shi, Xiaonan Nie, Zhi Yang, Yangyu Tao, and Bin Cui. 2021. [HET: scaling out huge embedding model training via cache-enabled distributed framework](#). *Proc. VLDB Endow.*, 15(2):312–320.
- Xupeng Miao, Hailin Zhang, Yining Shi, Xiaonan Nie, Zhi Yang, Yangyu Tao, Jie Jiang, and Bin Cui. 2025. [Efficient and scalable huge embedding model training via distributed cache management](#). *VLDB J.*, 34(3):27.
- Amirhossein Reisizadeh, Aryan Mokhtari, Hamed Hassani, and Ramtin Pedarsani. 2019. [An exact quantized decentralized gradient descent algorithm](#). *IEEE Trans. Signal Process.*, 67(19):4934–4947.
- Hao-Jun Michael Shi, Dheevatsa Mudigere, Maxim Naumov, and Jiyan Yang. 2020. [Compositional embeddings using complementary partitions for memory-efficient recommendation systems](#). In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pages 165–175. ACM.
- Xiaoniu Song, Yiwen Zhang, Rong Chen, and Haibo Chen. 2023. [UGACHE: A unified GPU cache for embedding-based deep learning](#). In *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023, Koblenz, Germany, October 23-26, 2023*, pages 627–641. ACM.
- OneRec Team. 2026. [Onereason technical report](#). *CoRR*, abs/2606.06260.
- Hung Vinh Tran, Tong Chen, Guanhua Ye, Quoc Viet Hung Nguyen, Kai Zheng, and Hongzhi Yin. 2025. [On-device content-based recommendation with single-shot embedding pruning: A cooperative game perspective](#). In *Proceedings of the ACM on Web Conference 2025, WWW 2025, Sydney, NSW, Australia, 28 April 2025- 2 May 2025*, pages 772–785. ACM.
- Kefan Wang, Hao Wang, Kenan Song, Wei Guo, Kai Cheng, Zhi Li, Yong Liu, Defu Lian, and Enhong Chen. 2025. [A universal framework for compressing embeddings in CTR prediction](#). In *Database Systems for Advanced Applications - 30th International Conference, DASFAA 2025, Singapore, Singapore, May 26-29, 2025, Proceedings, Part II*, volume 15987 of *Lecture Notes in Computer Science*, pages 84–100. Springer.
- Zehuan Wang, Yingcan Wei, Minseok Lee, Matthias Langer, Fan Yu, Jie Liu, Shijie Liu, Daniel G. Abel, Xu Guo, Jianbing Dong, Ji Shi, and Kunlun Li. 2022. [Merlin hugectr: Gpu-accelerated recommender system training and inference](#). In *RecSys '22: Sixteenth ACM Conference on Recommender Systems, Seattle, WA, USA, September 18 - 23, 2022*, pages 534–537. ACM.
- Minhui Xie, Youyou Lu, Jiazhen Lin, Qing Wang, Jian Gao, Kai Ren, and Jiwu Shu. 2022. [Fleche: an efficient GPU embedding cache for personalized recommendations](#). In *EuroSys '22: Seventeenth European Conference on Computer Systems, Rennes, France, April 5 - 8, 2022*, pages 402–416. ACM.
- Zhiqiang Xu, Dong Li, Weijie Zhao, Xing Shen, Tianbo Huang, Xiaoyun Li, and Ping Li. 2021. [Agile and accurate CTR prediction model training for massive-scale online advertising systems](#). In *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, pages 2404–2409. ACM.
- Bencheng Yan, Shilei Liu, Zhiyuan Zeng, Zihao Wang, Yizhen Zhang, Yujin Yuan, Langming Liu, Jiaqi Liu, Di Wang, Wenbo Su, Pengjie Wang, Jian Xu, and Bo Zheng. 2026. [Unlocking scaling law in industrial recommendation systems with a three-step paradigm based large user model](#). In *Proceedings of the Nineteenth ACM International Conference on Web Search and Data Mining, WSDM 2026, Boise, ID, USA, February 22-26, 2026*, pages 798–807. ACM.
- Jie Amy Yang, Jianyu Huang, Jongsoo Park, Ping Tak Peter Tang, and Andrew Tulloch. 2020. [Mixed-precision embedding using a cache](#). *CoRR*, abs/2010.11305.
- Qihang Yu, Kairui Fu, Zheqi Lv, Shengyu Zhang, Xinhui Wu, Chen Lin, Feng Wei, Bo Zheng, and Fei Wu. 2026. [Thinkrec: Thinking-based recommendation via LLM](#). In *Proceedings of the ACM Web Conference 2026, WWW 2026, Dubai, United Arab Emirates, originally scheduled for April 13-17, 2026, rescheduled for June 29 - July 3, 2026*, pages 5698–5709. ACM.
- Chaoliang Zeng, Xudong Liao, Xiaodian Cheng, Han Tian, Xinchun Wan, Hao Wang, and Kai Chen. 2024. [Accelerating neural recommendation training with embedding scheduling](#). In *21st USENIX Symposium on Networked Systems Design and Implementation, NSDI 2024, Santa Clara, CA, April 15-17, 2024*. USENIX Association.
- Buyun Zhang, Liang Luo, Yuxin Chen, Jade Nie, Xi Liu, Shen Li, Yanli Zhao, Yuchen Hao, Yantao Yao, Elie Dingqiao Wen, Jongsoo Park, Maxim Naumov, and Wenlin Chen. 2024. [Wukong: Towards a scaling law for large-scale recommendation](#). In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pages 59421–59434. PMLR / OpenReview.net.

Caojin Zhang, Yicun Liu, Yuanpu Xie, Sofia Ira Ktena, Alykhan Tejani, Akshay Gupta, Pranay Kumar Myana, Deepak Dilipkumar, Suvadip Paul, Ikuhiro Ihara, Prasang Upadhyaya, Ferenc Huszar, and Wenzhe Shi. 2020. [Model size reduction using frequency based double hashing for recommender systems](#). In *RecSys 2020: Fourteenth ACM Conference on Recommender Systems, Virtual Event, Brazil, September 22-26, 2020*, pages 521–526. ACM.

Han Zhang, Yunjiang Jiang, Mingming Li, Haowei Yuan, Yiming Qiu, and Wen-Yun Yang. 2025. [pebr: A probabilistic approach to embedding based retrieval](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025 - Industry Track, Suzhou, China, November 4-9, 2025*, pages 2332–2342. Association for Computational Linguistics.

Yuanxing Zhang, Langshi Chen, Siran Yang, Man Yuan, Huimin Yi, Jie Zhang, Jiamang Wang, Jianbo Dong, Yunlong Xu, Yue Song, Yong Li, Di Zhang, Wei Lin, Lin Qu, and Bo Zheng. 2022. [PICASSO: unleashing the potential of gpu-centric training for wide-and-deep recommender systems](#). In *38th IEEE International Conference on Data Engineering, ICDE 2022, Kuala Lumpur, Malaysia, May 9-12, 2022*, pages 3453–3466. IEEE.

Weijie Zhao, Deping Xie, Ronglai Jia, Yulei Qian, Ruiquan Ding, Mingming Sun, and Ping Li. 2020. [Distributed hierarchical GPU parameter server for massive scale deep learning ads systems](#). In *Proceedings of the Third Conference on Machine Learning and Systems, MLSys 2020, Austin, TX, USA, March 2-4, 2020*. mlsys.org.

Weijie Zhao, Jingyuan Zhang, Deping Xie, Yulei Qian, Ronglai Jia, and Ping Li. 2019. [Aibox: CTR prediction model training on a single node](#). In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019*, pages 319–328. ACM.

Xiangyu Zhao, Haochen Liu, Wenqi Fan, Hui Liu, Jiliang Tang, Chong Wang, Ming Chen, Xudong Zheng, Xiaobing Liu, and Xiwang Yang. 2021. [Autoemb: Automated embedding dimensionality search in streaming recommendations](#). In *IEEE International Conference on Data Mining, ICDM 2021, Auckland, New Zealand, December 7-10, 2021*, pages 896–905. IEEE.

Yang Zhou, Zhen Dong, Ellick Chan, Dhiraj Kalamkar, Diana Marculescu, and Kurt Keutzer. 2024. [DQRM: deep quantized recommendation models](#). *CoRR*, abs/2410.20046.

A Design Challenges

It is non-trivial to delicately integrate embedding quantization and caching within a unified decentralized training framework. We need to overcome the following obstacles.

Challenge #1: *How to evaluate the importance of individual embedding vectors?* A natural question is what kind of semantic representation is helpful for training performance. The importance of embedding vectors exhibits heterogeneity within and across features. Intra-feature representations have skewed access patterns due to different popularity, while inter-feature representations contribute unequally to model convergence due to different vector dimensions and table sizes. Besides, the embedding importance evolves dynamically over time. The metric design must account for heterogeneity and dynamics to maintain model performance (Section 3.2).

Challenge #2: *How to address the tightly coupled problem between mixed precision quantization and hierarchical caching?* The decisions for embedding quantization and caching are interactive. The quantization levels need to align with the cache hierarchy of embeddings, and the cache replacement policy requires adaptive adjustments based on compressed representations. Independent decisions or naive combinations of existing methods fail to coordinate resource utilization between two components. To cope with the tightly coupled problem, we need to jointly optimize quantization and caching decisions, which imposes additional difficulties to framework design (Section 3.4).

Challenge #3: *Does embedding training still converge after integrating quantization and caching under decentralized architecture?* None of existing studies investigates the quantitative relationship among embedding compression, caching, and training performance. It remains unclear how quantization and caching strategies affect the final convergence. The complexity is further amplified when training sparse models across multiple workers in a decentralized manner (Section 4).

B Pseudo Code of MixEmb

The training process of MixEmb is presented in Algorithm 1.

C Baselines

We compare MixEmb against five baselines. For a fair comparison, all baselines are implemented on the same platform as MixEmb.

- **PaddleBox** (Zhao et al., 2020) is a hierarchical training framework, which leverages excellent computing and communication capa-

Algorithm 1: Training Process of MixEmb

```

1 for Each iteration  $t = 0, 1, \dots, T - 1$  do
2   for Each worker  $n = 1, 2, \dots, N$  do
3     Sample a batch of data from local
       dataset  $\mathcal{D}_n$ ;
4     Split into local and remote keys;
5     Pull local quantized embeddings
       from hierarchical cache;
6     Pull remote quantized embeddings
       from other workers;
7     Model forward and backward
       propagation;
8     Synchronize parameters with
       AllToAll;
9     Update the embedding score
        $E_i^t = \lambda E_i^{t-1} + U_i^t$ ;
10    if  $t \bmod P = 0$  then
11      Update cache status by sorting
         $E_i^t$ ;
12      Perform cache replacement;
13      Mixed precision quantization
        based on cache hierarchy;
14    else
15      Write back to L1/L2 cache
        based on status of  $t - 1$ ;

```

bilities of modern GPUs to enable large-scale embedding training.

- **UGache** (Song et al., 2023) provides a unified multi-GPU cache mechanism that efficiently utilizes high-speed interconnects to accelerate memory accesses.
- **ALPT** (Li et al., 2023) is an adaptive low-precision training method designed to compress embedding tables during training, thereby achieving a trade-off between model accuracy and overhead.
- **MEC** (Wang et al., 2025) is a model-agnostic embedding compression framework that adopts popularity-weighted regularization and contrastive learning to improve the quality of quantized codes.
- **UniEmb** is a modified version of MixEmb, and its core design is similar except for quantization strategy. All embedding vectors are assigned the uniform quantization levels to enhance efficiency.

D Proof of Theorem 1

Local loss function is $f_n(\mathbf{w}) = \mathbb{E}_{\xi \sim \mathcal{D}_n} F_n(\mathbf{w}; \xi)$ where ξ denotes training samples from local dataset $\mathcal{D}_n$. Each function f_n is L -smooth, i.e., $\|\nabla f_n(\mathbf{x}) - \nabla f_n(\mathbf{y})\| \leq L \|\mathbf{x} - \mathbf{y}\|, \forall n, \forall \mathbf{x}, \forall \mathbf{y}$. The stochastic gradients have bounded variance, i.e., $\mathbb{E} \|\nabla F_n(\mathbf{x}; \xi) - \nabla f_n(\mathbf{x})\|^2 \leq \sigma^2, \forall n, \forall \mathbf{x}$. These assumptions hold for typical models and are widely used in previous studies (Chen et al., 2024). For the sake of presentation, we introduce the following notations:

$$\begin{aligned}\widehat{\mathbf{W}}^t &= [\hat{\mathbf{w}}_1^t, \hat{\mathbf{w}}_2^t, \dots, \hat{\mathbf{w}}_N^t] \\ \overline{\nabla F}(\widehat{\mathbf{W}}^t) &= \frac{1}{N} \sum_{n=1}^N \nabla F_n(\hat{\mathbf{w}}_n^t; \xi_n^t) \\ \overline{\nabla f}(\widehat{\mathbf{W}}^t) &= \frac{1}{N} \sum_{n=1}^N \nabla f_n(\hat{\mathbf{w}}_n^t) \\ \nabla f(\overline{\mathbf{w}}^t) &= \frac{1}{N} \sum_{n=1}^N \nabla f_n(\overline{\mathbf{w}}^t)\end{aligned}$$

Due to the smoothness of the loss function, we have:

$$\begin{aligned}& \mathbb{E} f(\overline{\mathbf{w}}^{t+1}) \\ & \leq \mathbb{E} f(\overline{\mathbf{w}}^t) + \mathbb{E} \langle \nabla f(\overline{\mathbf{w}}^t), -\eta \overline{\nabla F}(\widehat{\mathbf{W}}^t) \rangle \\ & \quad + \frac{L}{2} \mathbb{E} \left\| -\eta \overline{\nabla F}(\widehat{\mathbf{W}}^t) \right\|^2 \\ & = \mathbb{E} f(\overline{\mathbf{w}}^t) - \eta \mathbb{E} \langle \nabla f(\overline{\mathbf{w}}^t), \overline{\nabla f}(\widehat{\mathbf{W}}^t) \rangle \\ & \quad + \frac{L\eta^2}{2} \mathbb{E} \left\| \overline{\nabla F}(\widehat{\mathbf{W}}^t) - \overline{\nabla f}(\widehat{\mathbf{W}}^t) + \overline{\nabla f}(\widehat{\mathbf{W}}^t) \right\|^2 \\ & \leq \mathbb{E} f(\overline{\mathbf{w}}^t) - \eta \mathbb{E} \langle \nabla f(\overline{\mathbf{w}}^t), \overline{\nabla f}(\widehat{\mathbf{W}}^t) \rangle \\ & \quad + L\eta^2 \mathbb{E} \left\| \overline{\nabla F}(\widehat{\mathbf{W}}^t) - \overline{\nabla f}(\widehat{\mathbf{W}}^t) \right\|^2 \\ & \quad + L\eta^2 \mathbb{E} \left\| \overline{\nabla f}(\widehat{\mathbf{W}}^t) \right\|^2 \\ & = \mathbb{E} f(\overline{\mathbf{w}}^t) - \eta \mathbb{E} \langle \nabla f(\overline{\mathbf{w}}^t), \overline{\nabla f}(\widehat{\mathbf{W}}^t) \rangle + L\eta^2 \mathbb{E} \left\| \overline{\nabla f}(\widehat{\mathbf{W}}^t) \right\|^2 \\ & \quad + \frac{L\eta^2}{N^2} \mathbb{E} \left\| \sum_{n=1}^N (\nabla F_n(\hat{\mathbf{w}}_n^t; \xi_n^t) - \nabla f_n(\hat{\mathbf{w}}_n^t)) \right\|^2 \\ & \leq \mathbb{E} f(\overline{\mathbf{w}}^t) - \eta \mathbb{E} \langle \nabla f(\overline{\mathbf{w}}^t), \overline{\nabla f}(\widehat{\mathbf{W}}^t) \rangle + L\eta^2 \mathbb{E} \left\| \overline{\nabla f}(\widehat{\mathbf{W}}^t) \right\|^2 \\ & \quad + \frac{L\eta^2}{N} \sum_{n=1}^N \mathbb{E} \left\| \nabla F_n(\hat{\mathbf{w}}_n^t; \xi_n^t) - \nabla f_n(\hat{\mathbf{w}}_n^t) \right\|^2 \\ & = \mathbb{E} f(\overline{\mathbf{w}}^t) - \frac{\eta}{2} \mathbb{E} \left\| \nabla f(\overline{\mathbf{w}}^t) \right\|^2 - \frac{\eta}{2} \mathbb{E} \left\| \overline{\nabla f}(\widehat{\mathbf{W}}^t) \right\|^2 \\ & \quad + \frac{\eta}{2} \mathbb{E} \left\| \nabla f(\overline{\mathbf{w}}^t) - \overline{\nabla f}(\widehat{\mathbf{W}}^t) \right\|^2 + L\eta^2 \mathbb{E} \left\| \overline{\nabla f}(\widehat{\mathbf{W}}^t) \right\|^2 \\ & \quad + \frac{L\eta^2}{N} \sum_{n=1}^N \mathbb{E} \left\| \nabla F_n(\hat{\mathbf{w}}_n^t; \xi_n^t) - \nabla f_n(\hat{\mathbf{w}}_n^t) \right\|^2 \\ & \leq \mathbb{E} f(\overline{\mathbf{w}}^t) - \frac{\eta}{2} \mathbb{E} \left\| \nabla f(\overline{\mathbf{w}}^t) \right\|^2 - \frac{\eta}{2} (1 - 2\eta L) \mathbb{E} \left\| \overline{\nabla f}(\widehat{\mathbf{W}}^t) \right\|^2 \\ & \quad + \frac{\eta}{2} \mathbb{E} \left\| \nabla f(\overline{\mathbf{w}}^t) - \overline{\nabla f}(\widehat{\mathbf{W}}^t) \right\|^2 + L\eta^2 \sigma^2 \quad (7)\end{aligned}$$

Then, we give the bound of the term $\mathbb{E} \left\| \nabla f(\bar{\mathbf{w}}^t) - \nabla \bar{f}(\widehat{\mathbf{W}}^t) \right\|^2$:

$$\begin{aligned}
& \mathbb{E} \left\| \nabla f(\bar{\mathbf{w}}^t) - \nabla \bar{f}(\widehat{\mathbf{W}}^t) \right\|^2 \\
&= \frac{1}{N^2} \mathbb{E} \left\| \sum_{n=1}^N (\nabla f_n(\bar{\mathbf{w}}^t) - \nabla f_n(\hat{\mathbf{w}}_n^t)) \right\|^2 \\
&\leq \frac{1}{N} \sum_{n=1}^N \mathbb{E} \left\| \nabla f_n(\bar{\mathbf{w}}^t) - \nabla f_n(\hat{\mathbf{w}}_n^t) \right\|^2 \\
&\leq \frac{L^2}{N} \sum_{n=1}^N \mathbb{E} \left\| \bar{\mathbf{w}}^t - \hat{\mathbf{w}}_n^t \right\|^2 \\
&\leq \frac{2L^2}{N} \sum_{n=1}^N \mathbb{E} \left\| \bar{\mathbf{w}}^t - \mathbf{w}_n^t \right\|^2 + \frac{2L^2}{N} \sum_{n=1}^N \mathbb{E} \left\| \mathbf{w}_n^t - \hat{\mathbf{w}}_n^t \right\|^2 \\
&\leq \frac{2L^2}{N} \sum_{n=1}^N (\mathbf{R}_n^t + \mathbf{Q}_n^t) \tag{8}
\end{aligned}$$

By substituting Eq. (8) into Eq. (7), we obtain:

$$\begin{aligned}
& \mathbb{E} f(\bar{\mathbf{w}}^{t+1}) \\
&\leq \mathbb{E} f(\bar{\mathbf{w}}^t) - \frac{\eta}{2} \mathbb{E} \left\| \nabla f(\bar{\mathbf{w}}^t) \right\|^2 - \frac{\eta}{2} (1 - 2\eta L) \mathbb{E} \left\| \nabla \bar{f}(\widehat{\mathbf{W}}^t) \right\|^2 \\
&\quad + \frac{\eta L^2}{N} \sum_{n=1}^N (\mathbf{R}_n^t + \mathbf{Q}_n^t) + L\eta^2 \sigma^2 \tag{9}
\end{aligned}$$

Arranging the above inequality, we have:

$$\begin{aligned}
& \mathbb{E} \left\| \nabla f(\bar{\mathbf{w}}^t) \right\|^2 \\
&\leq \frac{2}{\eta} (\mathbb{E} f(\bar{\mathbf{w}}^t) - \mathbb{E} f(\bar{\mathbf{w}}^{t+1})) - (1 - 2\eta L) \mathbb{E} \left\| \nabla \bar{f}(\widehat{\mathbf{W}}^t) \right\|^2 \\
&\quad + \frac{2L^2}{N} \sum_{n=1}^N (\mathbf{R}_n^t + \mathbf{Q}_n^t) + 2L\eta\sigma^2 \tag{10}
\end{aligned}$$

If the learning rate $\eta < \frac{1}{2L}$, summing over $t \in \{0, 1, \dots, T-1\}$ and averaging on both sides yield:

$$\begin{aligned}
& \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E} \left\| \nabla f(\bar{\mathbf{w}}^t) \right\|^2 \\
&\leq \frac{2}{\eta T} (f(\bar{\mathbf{w}}^0) - f(\bar{\mathbf{w}}^T)) - \frac{1 - 2\eta L}{T} \sum_{t=0}^{T-1} \mathbb{E} \left\| \nabla \bar{f}(\widehat{\mathbf{W}}^t) \right\|^2 \\
&\quad + \frac{2L^2}{NT} \sum_{t=0}^{T-1} \sum_{n=1}^N (\mathbf{R}_n^t + \mathbf{Q}_n^t) + 2L\eta\sigma^2 \\
&\leq \frac{2}{\eta T} (f(\bar{\mathbf{w}}^0) - f^*) + 2L\eta\sigma^2 \\
&\quad + \underbrace{\frac{2L^2}{NT} \sum_{t=0}^{T-1} \sum_{n=1}^N \mathbf{R}_n^t}_{\text{Residual Error}} + \underbrace{\frac{2L^2}{NT} \sum_{t=0}^{T-1} \sum_{n=1}^N \mathbf{Q}_n^t}_{\text{Quantization Error}} \tag{11}
\end{aligned}$$