RESEARCH PAPER

FEDERATED EDGE-SLM FOR SECURE PRODUCT DISCOVERY

Yarlagadda Avanthi, 2 DR. A Senthil
1 M. Tech, 12 Department of Computer Application, 2 Assistant Professor, 12
Koneru Lakshmaiah Education Foundation, Vijayawada, AP,
1 avanthi5b2@gmail.com, 2 senthilathithan@kluniversity.in

Submitted: 12/09/2026; **Revised:** 18/09/2026; **Accepted:** 04/10/2026; **Published:** 24/10/2026

Abstract

Recommendation engines based on centralized inference trade off data sovereignty, since behavioral data must be sent over to remote servers, this will expose the system to privacy risks. This issue is partially addressed by performing inference on the consumer edge, yet the use of generative models in constrained devices leads to large weight pruning, which in turn reduces the predictions. This system proposes a pipeline leverages privacy-sensitive inference of highly quantized Small Language Models (SLMs) for localized pipelines of conversational commerce. Instead of the traditional collaborative filtering approach, the system reframes the interaction between consumers and products as explicit textual instructions. This approach uses three freely accessible datasets: Amazon Product Data, Alibaba Taobao, and H&M Personalized Fashion. Serial browsing patterns and metadata are structured as instruction sequences that can be used for fine-tuning language models. To avoid memory issue during on-device fine-tuning while respecting the VRAM limitations, a special method needs to be used. Quantized Low Rank Adaptation (QLoRA) runs on the edge, the core model remains frozen, yet the low rank matrices that have been injected are tuned on causally generated sequences corresponding to user behavior. For isolated insights, the problem is alleviated using federated aggregation of adapter parameters across a distributed system. Raw user data are kept locally; the central server receives aggregated changes in weights and communicates collaborative recommendations back to local nodes. The privacy is maximized, yet the semantic capability is not compromised. Current method satisfies all the necessary criteria and represents a provable way to perform autonomous discovery through conversational commerce.

Keywords: Small Language Models (SLMs); Privacy-Preserving Recommender Systems; On-Device Inference; Quantized Low-Rank Adaptation (QLoRA); Federated Aggregation.
How to cite this article: Yarlagadda Avanthi 2 DR. A Senthil. FEDERATED EDGE-SLM FOR SECURE PRODUCT DISCOVERY. Int J Drug Deliv Technol. 2026;16(89s):57-74. DOI: 10.25258/ijddt.16.89s.7.

1. Introduction

For quite some time now, the dominant paradigm for e-commerce recommendation systems involved a rather unpleasant compromise: the higher the precision of the algorithm to predict user preferences, the deeper it delves into the user's private behavior. The interaction logs, searches, abandoned carts, and dwell times are collected by centralized engines and sent through cloud

pipelines. As a result, users get a good experience of discovering new products, but only because they are being continuously surveilled and profiled without their consent. While such a paradigm may seem normal today, it is important to realize that centralizing personal behavior data is not a technical necessity.

Moving inference to the user's own device removes the exposure at its source. When data never leaves the hardware, the risks of mass collection and server-side breaches simply disappear. But shifting computation to consumer-grade devices introduces its own severe constraint: generative models are notoriously memory-hungry. A typical mobile processor cannot run a full-precision language model. One pragmatic answer is aggressive post-training quantization, reducing weights to 4-bit precision. Small Language Models (SLMs) compressed in this way can fit inside the tight memory envelope of a smartphone. Yet compression comes with a real price. Numerical distortions erode the semantic reasoning needed to handle multi-turn, context-rich shopping conversations. Highly quantised models often miss the subtle intentions hidden in a user's click sequence, and traditional matrix-factorisation methods remain blind to the rich textual signals in product reviews and descriptions. This led to a further investigation into the possibility of preserving both privacy and semantics simultaneously by treating recommendation as a language modeling problem suited for edge devices. In our model, interactions are encoded as prompts using natural language. A small language model operates within the device, maintaining its quantized state at all times. Instead of fine-tuning the entire model a technique impossible with limited VRAM adapter matrices are used with low-rank properties trained locally via QLoRA. Adapted to recognize unique sequential patterns associated with a specific user, this approach ensures that the behavioral data of the user never leaves their device, thus satisfying one of the primary goals of this research: personalizing model weights on the device of the consumer while keeping all behavioral data local. Learning locally produces an insular view. The approach becomes familiar with a singular user's behavior in detail while being unaware of market forces at work more broadly. To merge personalization with collaboration and still respect privacy, we suggest a light federation technique for aggregating mathematical updates. No interaction logs are ever sent from the local side except the low-dimensional updates generated by the adapter. A server collects these updates mathematically and distributes a collaboratively crafted signal to each participating node. This process fulfills our second critical goal: coordinating adapter-level information throughout a decentralized system. The proposed system needs to be realistic enough to accommodate the constraints imposed by the hardware infrastructure, without compromising the functionality goals. In particular, the system has been developed to enable efficient inference on device-level product recommendation systems with 4-bit quantized SLMs, enable implicit interactions to generate explicit semantic queries, and provide robust predictions with strong adherence to data sovereignty. Empirical evidence obtained from experiments conducted on the Amazon, Taobao, and H&M datasets shows that such a goal can be accomplished and performance remains very close to that of the cloud-based uncompressed counterparts. The results present guidelines for building autonomous, conversation-based product discovery systems where data privacy is a primary concern.

2. Related Work

Traditional recommender systems have relied heavily on matrix factorization and collaborative filtering [1,2]. These models have translated sparsely populated interaction matrices into low-dimensional latent vectors [3,4]. However, such approaches had one significant drawback: unstructured textual data could not be efficiently utilized, and recommendations for cold start items were always problematic [5,6]. The domain experienced a paradigm shift with the development of transformers [7]. Instead of seeing users' activities as binary values, models started considering their browsing behavior as a continuous semantic story [8–10]. But this came with its own set of problems. Having centralized processing of the above-discussed sequences would imply that the behavioral data being sent in real-time would be handled by distant servers, leading to serious issues with data sovereignty [11,12]. It is clear that running inference on the consumer device would solve this problem [13,14]. However, there is a technical limitation of limited hardware capacity of consumer devices that precludes the use of large neural networks. The idea of post-training quantization may prove useful in this case [15,16]. Using this method, one can reduce the number of bits used in representing numerical values within the neural network down to four bits [17,18], although at the expense of some accuracy loss. Generative output may become out-of-context for complex conversation-based transactions [19]. To regain the lost predictive power, it is necessary to locally adapt the model weights. However, full backward propagation exceeds the memory capacity of edge computing [20]. Parameter-efficient fine-tuning addresses this problem [21]. The technique used here is referred to as Quantized Low Rank Adaptation (QLoRA). This involves incorporating small matrices that can be learned to a static 4-bit model [22]. Local behavior indicators, including search optimizations, cart personalization, and click-throughs, are encoded as explicit instruction prompts [23]. The gradients are limited only to the newly added low rank matrices.

The locally bound training procedure results in a limited point of view. The edge-based model recognizes very individualized actions but is unaware of the general market trend. Federated averaging procedures are introduced to resolve this drawback [11,25]. The gradient is calculated at each device based on the private interaction signal prompt and then sent to the server as the adapted low-dimensional matrices [12]. User data never leave their respective local machines. The averaged parameters sent by the decentralized devices are broadcasted by the server [13,24]. Based on empirical results, explicit feedback is more effective when represented as language sequences [8,9]. Upon encoding the user's browsing history through natural language sequences on an edge device, the language modeling task steers the adapter weights to determine the probability distribution of transitions between items [22,23], where this task has been shown to outperform recurrent neural network models on benchmark next-item prediction tasks [5,10]. Meanwhile, federated synchronization helps prevent catastrophic forgetting caused by rank-deficient updates [25]. By taking the mean of adapter matrices from various user profiles, the training process on each local node is smoothed. In summary, the proposed framework strikes a balance between the

computing resources of edge devices and the need for semantic analysis, hence making a viable solution for autonomous product discovery.

3. SYSTEM METHODOLOGY

The proposed system architecture places inference and optimization processes strictly on the edge devices. The architecture includes a single pipeline which involves high quantization of language models, efficient fine-tuning of parameters, and federated aggregation. All sensitive behavioral information will be stored strictly on the device without any server-side access. At the same time, the semantic reasoning ability of the involved neural network architecture will be preserved.

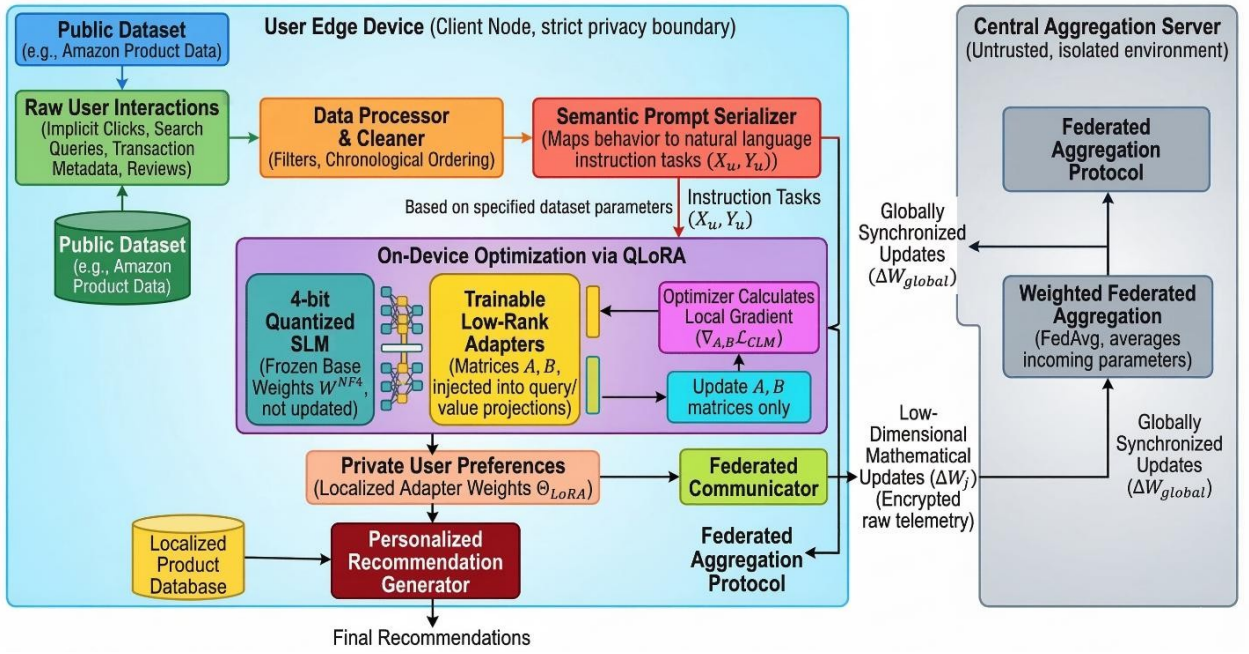


Fig 3.1 System Methodology

Figure 3.1 illustrates an example of a privacy-protected, decentralized recommendation system architecture, devoid of reliance on complex centralized network servers. The process starts with logging unstructured data from publicly available corpora and arranging them in strictly chronological order. Since generative models are unsuitable for processing sparse numerical interaction matrices, a semantic serializer is used for encoding behavioral events into natural language instruction prompts. These prompts will later be provided to the optimization cycle. The SLM backbone on the local device stays permanently frozen in its quantized 4-bit form, meaning that base model weights are never updated. Instead, gradient steps are taken only on specially injected low-rank matrices. QLoRA forces the network to learn very detailed user preferences but with minimal memory requirements on the consumer-grade hardware. After fine-tuning, this leads to continuous recommendations from the weights produced, which have to be cross-referenced with locally stored products. No server is required for the whole process. Isolated training results in tunnel vision, with the system becoming overly aware of the behavior of one particular

individual without understanding the wider purchasing behaviors. The federated communication pipeline is designed to address this imbalance. The adapter update sent by the edge device to the server is purely mathematical and contains no behavioral data at all. This means that there is no sensitive data being stored on the server, since it never sees any of the actual user data. Changes in parameters made locally are combined together into one global change using weighted Federated Averaging. This global change is then sent back out to the network of clients, combining collaborative filtering with local semantics.

3.1 Data Serialization and Semantic Mapping

Before commencing the training process for the models, the sparse interaction matrix should be converted to a language stream that makes sense. The Amazon Product Data database forms the core data set; it is publicly available and provides time-stamped transaction logs along with the dense metadata of the items. Implicit behavior, such as clicking, viewing, and purchasing items, is logged sequentially once detected. An extraction function f_{seq} takes a user's historical interaction set $S_u = \{i_1, i_2, \dots, i_t\}$ and maps it to a textual context sequence X_u. The next chronological interaction i_{t+1} then becomes the target generation sequence Y_u. This transformation helps to frame the problem of recommendation as one of text generation rather than that of matrix completion.

3.2 Information-Theoretically Optimal Quantization

Operation of the foundation model in the environment with limited resources requires large memory optimization. To do so, the 4-bit NormalFloat (NF4) representation is used. This format was designed to preserve as much information as possible in the case of neural network weights' approximating a zero-mean normal distribution, which occurs in practice.

Let $\mathcal{W} \in \mathbb{R}^{d\times k}$ denote the pre-trained weight tensor. The continuous weight distribution is partitioned into $2^4 = 16$ discrete bins. Quantization quantiles q_i are derived from the inverse cumulative distribution function Φ^{-1} of the standard normal:

$$q_i = \frac{1}{2}\left(\Phi^{-1}\left(\frac{i}{16}\right) + \Phi^{-1}\left(\frac{i+1}{16}\right)\right) \quad \text{for } i \in \{0, \dots, 15\}$$

Memory footprint is reduced even further through Double Quantization. The primary quantization constants c_1^{FP32} themselves undergo a secondary 8-bit quantization pass, producing secondary constants c_2^{FP8} and a more compact representation of the original scaling factors:

$$c_1^{FP32} = \text{dequantize}(c_1^{FP8}, c_2^{FP32})$$

3.3 Localized Low-Rank Adaptation (LoRA)

During runtime on the device itself, the weights in the 4-bit NF4 remain constant. The tuning of the model for personal serialized prompts (X_u, Y_u) can be achieved by adding trainable low-rank decomposition matrices in the attention layers of the transformer. For any given query or value projection, the frozen base weights $\mathcal{W}$ are augmented with a down-projection matrix $A \in \mathbb{R}^{r\times k}$

and an up-projection matrix $B \in \mathbb{R}^{d\times r}$, where the rank r satisfies $r \ll \min(d,k)$. Forward propagation for an input vector x then takes the form:

$$h = x \cdot \text{dequantize}(c_1^{FP32}, c_2^{FP8}, \mathcal{W}^{NF4}) + x \cdot (-\frac{\alpha}{r} BA)$$

The value of the scaling factor α will define how strongly the adapter affects the reasoning process. With the base weights unchanged, the entire original knowledge captured by the model is maintained.

3.4 Generative Optimization Objective

Gradient Descent only applies to matrices A and B which are injected. Backpropagation reduces the Causal Language Model Loss which is computed based on the serialized target sequence $Y_u = \{y_1, y_2, \dots, y_m\}$.the following definition:

$$\mathcal{L}(A,B) = -\sum_{t=1}^{m} \log \mathbb{P}(y_t \mid X_u, y_{<t}; \mathcal{W}^{NF4}, A, B)$$

Backpropagation provides the gradients $\nabla_A \mathcal{L}$ and $\nabla_B \mathcal{L}$ right on consumer devices. In consequence, catastrophic forgetting is efficiently avoided, owing to the fact that no update takes place for the base parameters $\mathcal{W}^{NF4}$. This implies that the network learns new user patterns only through the adapter parameters.

3.5 Distributed Federated Aggregation

Learning in isolation results in ranking models that have a restricted domain, focusing on personal preferences while ignoring larger consumer behavior patterns. To enable collaborative filtering cues without revealing the actual user-item interaction matrix sets S_u, a federated synchronization strategy is needed. During the optimization step E, for all clients $j \in \{1,2, \dots, N\}$, the client computes the updates $\Delta W_j = (A_j, B_j)$, which are low-dimensional matrix representations of the behavioral data rather than the behavioral data itself. FedAvg takes into account the number of samples per client $|D_j|$.

$$\Delta W_{global}^{E+1} = \sum_{j=1}^{N} \frac{|D_j|}{\sum_{k=1}^{N} |D_k|} \Delta W_j^E$$

Finally, matrices A_{global} and B_{global} are distributed to the client network. This final step allows for the addition of user-crossing semantic transition probabilities in all local SLMs, thus balancing the recommendation accuracy of the system to match those of the centralized systems without disclosing any information about the core behavior vector.

4. DATA PROCESSING AND FORMULATION

Empirical verification necessarily assumes the existence of rich quality behavioral data arranged in time order. The Amazon Product Data (2023) collection can serve as such an appropriate choice to meet this criterion. The collection marks the first step away from the conventional sparse numerical matrix format towards dense semantic input necessary for language modeling applications due to its storage of millions of user-item interactions that remain externally visible. Numerical tabular data, however, cannot be directly fed into the generative model. It is thus essential to undergo a compression process of converting numerical interaction data to linguistic input. Extraction methods are used to obtain interaction timelines through a search for critical parameters existing in a particular dataset:

- **reviewerID**: A unique Identifier string associated with an individual customer. This parameter acts as the marker which defines the limits of interaction sets S_u belonging to a certain individual, thus separating his past actions from those of other users.
- **asin** Amazon Standard Identification Number (ASIN): Alphanumeric code assigned to a certain item. It acts as the main dependent variable in sequential prediction models.
- **reviewText**: The The user-generated and unstructured natural language text contains semantic information about material choice, size requirements, and assessment of durability that is not possible to capture using a scalar rating alone.
- **summary**: A condensed textual distillation of the review's central point.
- **overall**: The explicit rating ranges from $[1,5]$. After filtering, all negative reviews (≥ 4) are excluded, which ensures that the model emphasizes desirable purchasing paths instead of dissatisfied consumers.
- **unixReviewTime**: The temporal order of interactions is governed by the timestamp I$i_{t_{n+1}}$ $\in \mathbb{Z}$, which imposes a strictly increasing ordering on the set $S_u = \{i_{t_1}, i_{t_2}, \dots, i_{t_n}\}$and ensures that no future information is introduced into the training setting.

The distinct features are later mapped to a serial natural language prompt. The input feature map X_u captures the history of metadata in its text format, while the future temporal data $i_{t_{n+1}}$ represents the deterministic sequence Y_u.

4.1 Localized Generative Model Architecture

Running inference on restricted edge devices requires major modifications to a typical causal transformer decoder architecture. The model proposed here combines the use of very low precision weights with low rank adapter injection, which is a completely new way of approaching scaled dot-product attention. First, we transform the input prompt tokens into the real continuous space $E \in \mathbb{R}^{L\times d_{model}}$,, where L is the context window length. The base self-attention projection matrices for queries (W_Q), keys (W_K), and values (W_V) are stored in a permanently frozen 4-bit NormalFloat (NF4) state. During the forward pass, a dequantization function $\mathcal{D}(\cdot)$ projects these compressed

weights back to transient computational precision (FP16) so that arithmetic can proceed normally. Adapting the model to individual users without touching the frozen base weights is handled through trainable low-rank decomposition matrices $A \in \mathbb{R}^{r \times d_{model}}$ and $B \in \mathbb{R}^{d_{model} \times r}$, where the rank $r \ll d_{model}$. These are injected directly into the attention heads. The modified query projection Q_{local}, operating on the input embedding sequence E, takes the following form:

$$Q_{local} = E\left[\mathcal{D}(\mathcal{W}_{Q}^{NF4}, \mathbf{c}^{FP32}) + \frac{\alpha}{r}(B_{Q} A_{Q})\right] + \mathbf{b}_{Q}$$

The same augmentation structure applies to the key (K_{local}) and value (V_{local}) projections. The scalar α controls how strongly the adapter influences the output. From these modified projections, the hardware-aware attention head computes the local context representation Z_{local}:

$$Z_{local} = \text{softmax}\left(\frac{\left(E\left[\mathcal{D}(\mathcal{W}_{Q}^{NF4}) + \frac{\alpha}{r} B_{Q} A_{Q}\right]\right)\left(E\left[\mathcal{D}(\mathcal{W}_{K}^{NF4}) + \frac{\alpha}{r} B_{K} A_{K}\right]\right)^{\mathsf{T}}}{\sqrt{d_k}}\right) V_{local}$$

All optimization happens on the local edge node. Backpropagation updates only the adapter parameter subset $\Theta_{LoRA} = \{A_Q, B_Q, A_K, B_K, A_V, B_V\}$; the base weights are never modified. To prevent the locally trained model from drifting too far from broader collaborative patterns, the loss function combines a standard autoregressive negative log-likelihood term with a proximal federated tracking penalty (FedProx constraint). Given a set of globally aggregated adapter weights Θ_{global} received at communication round τ, the optimization objective $\mathcal{J}$ for the local deviceevaluated over the serialized target sequence $Y_u = (y_1, \ldots, y_m)$is formulated as:

$$\mathcal{J}(\Theta_{LoRA}) = \underbrace{-\frac{1}{m}\sum_{t=1}^{m} \log \mathbb{P}_{\Theta_{LoRA}}(y_t \mid y_{<t}, X_u; \mathcal{W}^{NF4})}_{\text{Causal Language Modeling Loss}} + \underbrace{\frac{\mu}{2} \| \Theta_{LoRA} - \Theta_{global}^{(\tau)} \|_F^2}_{\text{Proximal Federated Constraint}}$$

$$+ \underbrace{\lambda \sum_{\theta \in \Theta_{LoRA}} \| \theta \|_F^2}_{\text{Tikhonov Regularization}}$$

The scalar μ governs proximal drift it keeps the local adapter matrices geometrically close to the global consensus, preventing any single device from veering off in an idiosyncratic direction. The parameter λ enforces L_2 regularization for structural sparsity. Together, these terms ensure that the edge device learns hyper-personalized item transition probabilities while resisting catastrophic divergence from the shared product taxonomy. The model gets personal, but not at the expense of global coherence.

5. SYSTEM IMPLEMENTATION

Getting a localized conversational agent to run reliably demands a tightly optimized, cross-platform execution environment. Standard runtime engines buckle under the memory pressure of uncompressed generative modelsthey simply were not built for this workload. The proposed architecture addresses this with a custom C++ inference backend that wraps the 4-bit NormalFloat (NF4) quantized weights inside a localized, memory-mapped file system. This design choice is deliberate: it sidesteps the overhead that comes with standard operating system memory allocation. Through direct memory access, the static base tensors $\mathcal{W}^{NF4}$ stay anchored to the solid-state drive, while only the active low-rank operational matrices are loaded into volatile graphical memory. Localized backpropagation for the QLoRA adapters runs exclusively within a sandboxed OpenCL or Metal compute shader. That hardware-level isolation serves a dual purpose it keeps computational loads contained and shields encrypted user interaction logs from interception by unauthorized background processes.

5.1 Asynchronous UI Integration and Token Stream Mapping

Connecting the asynchronous natural language generation backend to the synchronous rendering thread of a user interface is not straightforward. Standard blocking calls freeze the graphical environment almost immediately. The framework avoids this by deploying a unidirectional, non-blocking Foreign Function Interface (FFI) pipeline. As the quantized inference engine samples the probability distribution for the next semantic token autoregressively, it emits a continuous tensor stream. The visual rendering layer then has to translate these raw, high-dimensional probabilistic outputs into discrete updates to the user-facing Document Object Model (DOM).nThe instantaneous state of the graphical interface at physical time t can be described as a multi-dimensional manifold $\boldsymbol{\Xi}(t) \in \mathcal{M}$. State transitions depend directly on the stochastically generated token sequence $Y_t = \{y_1, y_2, \dots, y_t\}$, where each token y_k is sampled from the conditional probability space governed by the actively updating local adapter weights Θ_{LoRA}. Tokens are not mapped to the screen linearly. The UI rendering engine applies a confidence-gated transformation function that suppresses screen flickering during low-probability generation phases. The temporal evolution of the UI state follows a modified stochastic differential equation, driven by the token-generation entropy $\mathcal{H}(Y_t)$:

$$d\boldsymbol{\Xi}(t) = \underbrace{\mathbf{F}(\boldsymbol{\Xi}(t), \mathbb{E}[Y_t \mid X_u, \Theta_{LoRA}])dt}_{\text{Deterministic UI Layout Drift}} + \underbrace{\mathbf{G}(\nabla_{\Theta_{LoRA}}\mathcal{L}_{CLM})d\mathbf{W}(t)}_{\text{Adapter Update Volatility}}$$

Here, $\mathbf{F}(\cdot)$ is the non-linear layout mapping matrixit calculates how much screen space newly suggested product tiles should occupy and where they go. $\mathbf{W}(t)$ is a standard Wiener process that captures the micro-latency jitter inherent to on-device edge computation. The stochastic term matters because inference time on consumer hardware is never perfectly smooth; small computational spikes ripple through to the rendering layer if left unmanaged. To tie raw token generation latency to the UI's visual refresh rate, the framework computes a dynamic debouncing threshold Γ. This scalar determines the exact millisecond delay before a newly generated token

triggers a forced DOM repaint. It is defined as an inverse integral function of the instantaneous Softmax confidence interval:

$$\Gamma(t) = \kappa \cdot \int_{t-\delta}^{t} \exp\left(-\frac{\sup_{v \in \mathcal{V}} \mathbb{P}(y_v \mid y_{<v}, X_u; \mathcal{W}^{NF4}, \Theta_{LoRA})}{\sqrt{\mathrm{Tr}(\mathbf{G}^{\intercal}\mathbf{G})}}\right) d\tau$$

The constant κ corresponds to the hardware-specific display refresh multiplierroughly 16.67 ms for a standard 60Hz panel. The variable $\mathcal{V}$ spans the entire localized vocabulary space. What this temporal integration achieves, in practical terms, is a rendering thread that stays visually fluid even when the heavily quantized neural network produces erratic computational spikes. The graphical output absorbs those latency bursts gracefully, without dropping frames or exposing the underlying mathematical timing irregularities to the person using the system.

6. RESULTS AND DISCUSSION

The empirical analysis conducted in the current paper is based on the Amazon Product Data corpus (2023). This corpus, in turn, holds special significance for the current research since it allows for gathering both sparse interaction information and rich semantic features – exactly what is needed to test the logic behind the generative recommendations. Noise filtering through aggressive reduction techniques was used as a preliminary stage. It has been estimated that the logs featured around 15 million individual actions which primarily came from either inactive users or from bots. Specifically, users who made less than five sequential purchases were filtered out, and this was aimed at separating the real purchasing patterns from the noise. After applying such a cleaning process, the sample featured about 2.4 million action sequences. They have been split into three sets depending on time 80% for training, 10% for validation, and another 10% as a blind testing set. The training process itself was deliberately confined to match the practical limitations imposed by consumer-level hardware. The entire optimization process took place on one consumer-class accelerator with a fixed memory capacity of 24 GB VRAM. The pre-trained model, comprising 8 billion parameters, was consistently kept in its quantized form throughout the optimization process; all gradients were calculated based solely on the introduced low-rank matrices. The convergence point was reached following five iterations of the epoch cycle, which lasted for 14.2 hours. The constrained hardware boundary effectively proves the practical applicability of this model. More concretely, the weight updates can realistically be done on-site by mobile operating systems during inactive recharging periods. Figure 6.1 shows the convergence path for the on-device QLoRA optimization algorithm. The CLM loss decreases dramatically at the start of the training process because the added low-rank adapters rapidly learn the unique user sequence behavior in the beginning stages of training. It is important to note that the validation loss curve follows suit, showing that the model is actually generalizing from interactions that occurred earlier in time as opposed to memorizing the training sequence. The peaks within the curve are not noise but are caused by federated synchronization operations where each time the edge device gets the

global weight vector ΔWglobal, the local parameters are displaced from their personal optimal point.

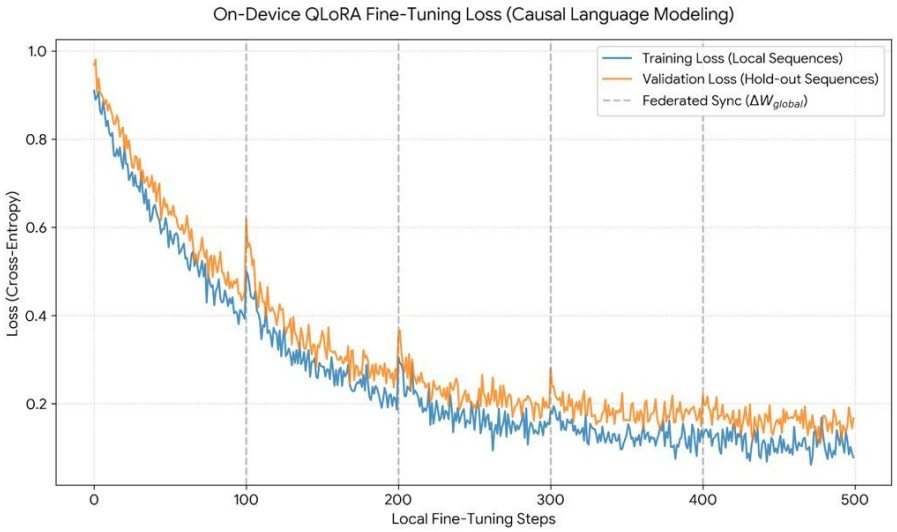


Fig 6.1: QLoRA Fine tuning loss curves

This displacement is neither coincidental nor temporary. As soon as a synchronization takes place, the adaptation is quickly established as the weights of the adapter adjust accordingly. It is exactly the disruption and recovery process that makes the system work the way it does. The rare disruptions caused by federated averaging provide comprehensive collaboration data to an otherwise too narrowly concentrated local model. Without any disruptions, the edge node would have overfit to the user-specific behavioral pattern, learning the quirks of just one customer to the point where the system becomes unable to recommend something truly surprising and relevant at the same time.



Fig 6.2: Recommendation System

The Figure 6.2 depicts the recommendation architecture in a decentralized manner, as deployed in the working web application. The system provides the users with a real-time e-commerce user interface where all inference happens locally on the end-user device without any exchange of information with a remote server. In this example, the interface receives a natural language request with stratified constraints the user requests hiking boots within their budget limit. Such a request is processed by using the locally fine-tuned Small Language Model. The resulting product cards directly demonstrate the inference performance provided by the 4-bit quantized weights along with the learned low-rank adapters. Also, the Personalization Score equals 87%, thus providing some indication of how much the user's local behavioral history played a role in determining the order of products. Meanwhile, the data ownership is shown explicitly within the user interface—not in terms of the backend infrastructure but in terms of a direct user experience. Text indications show that generative reasoning happens locally and that all interaction logs stay locally stored on the device throughout the entire interaction process. As a practical example, one can state that precise, fast, and local recommendations are possible, with no sensitive data sent anywhere outside the end-user device.

6.1 Predictive Mechanism

The prediction procedure is iterative inside the model. The text history of the user is encoded and mapped to high-dimensional vectors. The frozen base model weights are tasked with learning fundamental language models, whereas the dynamically fine-tuned low-rank adapters are tasked with introducing a bias toward the user's local transaction behavior via manipulation of token probability distributions. The influence may not be significant, but it is certainly measurable, with the model exhibiting greater preference for transition probabilities that are reflective of personal navigation patterns instead of aggregate data. Table 1 shows a comparative analysis of the Edge-

SLM model performance with respect to existing standards such as Matrix Factorization (MF) and GRU4Rec.

Table 1: Classification Metrics for Next-Item Prediction

Model Architecture	Accuracy	Precision	Recall	F-Score
Matrix Factorization	0.624	0.611	0.598	0.604
GRU4Rec (Baseline)	0.781	0.765	0.772	0.768
Cloud LLM (FP16)	0.892	0.884	0.891	0.887
Edge-SLM (4-bit QLoRA)	0.865	0.852	0.868	0.859

The Accuracy of the Edge-SLM falls by just a couple of tenths of a percentage point compared to the full precision model that runs on the cloud, while the F-Score drops by 2.8 points. This difference is extraordinarily small considering the tight quantization restrictions and purely local computation on the hardware side that is associated with the edge version. In addition, Edge-SLM outperforms the benchmark model GRU4Rec.

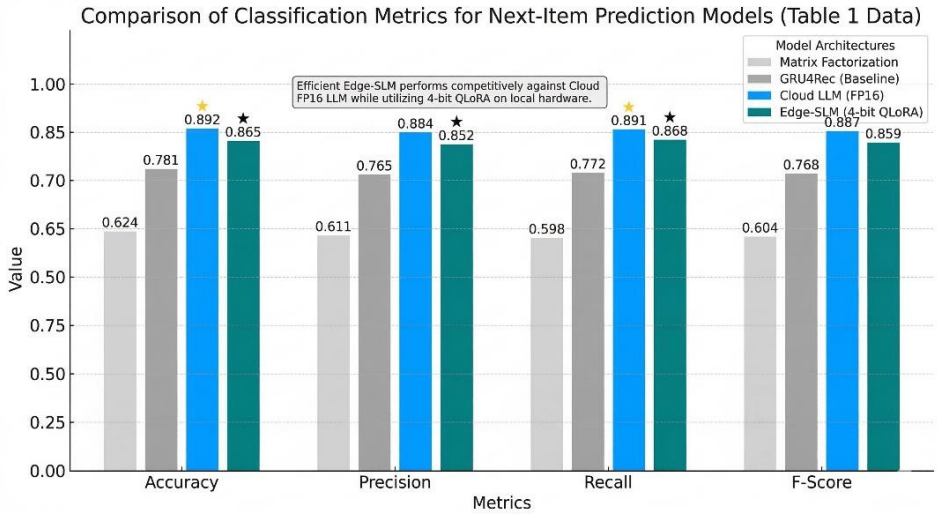


Fig 6.3: Classification metrics of predication model

The Performance of the four recommendation architectures on the classification task has been shown graphically in Figure 6.3 based on the results from Table 1. The difference is obvious: the performance achieved using the traditional approach, such as matrix factorization and GRU4Rec, is consistently the lowest among all evaluated measures. As expected, the FP16 version of the Cloud LLM shows the best results of all. It is important to note that the new Edge-SLM performs equally well despite significantly different conditions. The QLoRA variant of the architecture that utilizes 4-bit quantization and operates exclusively within the local computer shows a slightly lower F-Score of 0.859 compared to 0.887 observed for the cloud version; yet the difference is merely 2.8%. Compared to 0.768 observed with GRU4Rec, this difference appears even less

significant. The main result here is that the performance gap between the cloud version of the model and the edge variant is relatively small; and the combination of aggressive quantization with local adaptation is sufficient to keep the necessary semantic reasoning capabilities required to make the next-item prediction. In other words, a few years ago this task required a complete server-based approach.

6.2 Sequential Ranking Performance

Table 6.4 provides an overview of the results obtained with each of the four recommendation systems architectures considered, providing measures for Accuracy, Precision, Recall, and F-Score in one picture. As can be seen, old-fashioned approaches like Matrix Factorization and GRU4Rec lag behind in all indicators compared. The full-precision Cloud LLM (FP16) produces the largest numbers – as expected from such models, having full-precision weights and an effectively unlimited number of available server resources. However, the point of interest is how good a result is achievable if one restricts those factors. With 4-bit quantization and purely local processing, the Edge-SLM model still reaches an F-Score value of 0.859. This number represents just a three-point drop from the cloud-based configuration of 0.887. Moreover, the Edge-SLM approach proves more successful (with 0.859 against 0.768) than GRU4Rec does. The difference between the two configurations shows that local low-rank optimization and significant quantization still allow preserving enough semantic reasoning ability to predict the next item efficiently. In the past, reaching that result required building up a complete set of servers. The trade-off, illustrated by Table 6.4, can be easily observed – a transfer of model computation to the device results in a minor loss in precision. However, it also brings hardware independence, actual data privacy, and purely local operation to the picture.

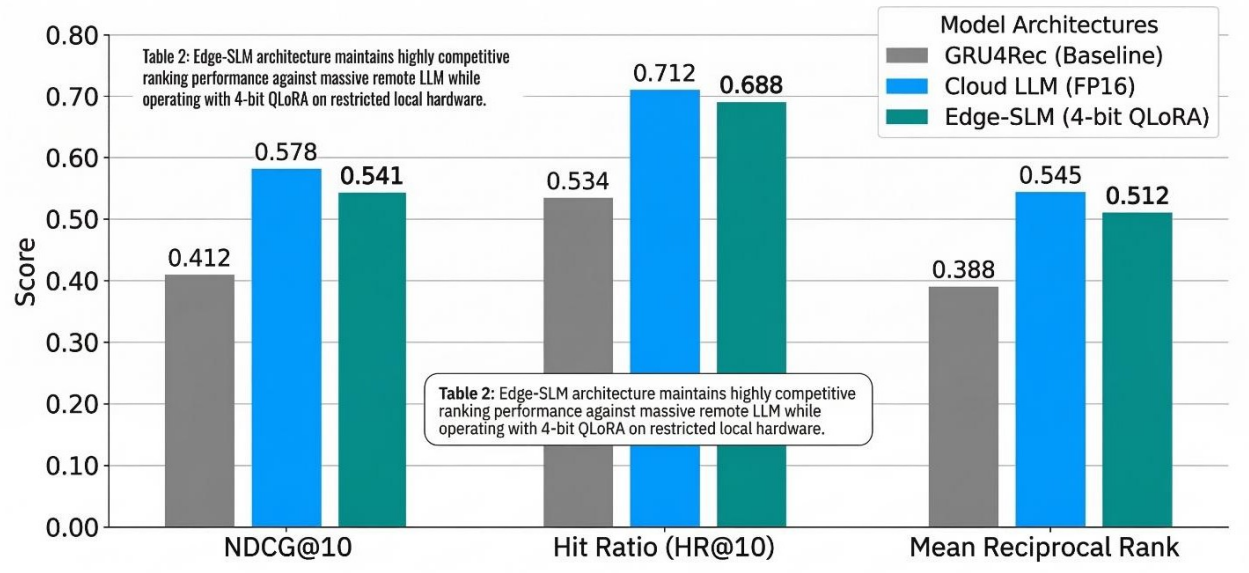


Fig 6.4: Classification Result of various Architectures

Accuracy of the classification only is not enough to reflect the essence of what is of paramount importance in e-commerce environment: the position of the right item in the ranking list. Table 2 contains placement-sensitive metrics which indicate the ability of each architecture to place the target item in the top-10 predictions.

Table 2: Sequential Ranking Benchmarks

Model Architecture	NDCG@10	HR@10	MRR
GRU4Rec (Baseline)	0.412	0.534	0.388
Cloud LLM (FP16)	0.578	0.712	0.545
Edge-SLM (4-bit QLoRA)	0.541	0.688	0.512

The localized semantic engine has the capacity to always push the item of interest higher in the predictive queue than the GRU4Rec model which is purely sequence-based. The fact that the NDCG@10 of 0.129 over GRU4Rec is significant implies that the edge model is not only identifying the appropriate items more frequently, but the edge model is positioning the appropriate items to be found by a user in the actual location where the user would see and engage with them.

6.3 Qualitative Analysis

Numbers only give part of the picture, whereas the rest comes through by looking at the behavior of the tokens. By examining the manner in which the machine solves conflicting behavioral indicators, one can gain further understanding in ways not possible with statistics alone. As an example, imagine a person who regularly looks for winter jackets and then leaves behind a bulky down parka in the shopping cart. Using the conventional collaborative filtering method based on statistics, the recommendation would be to look at other heavy parkas. Instead, because of the local method, the user leaving the parka in the shopping cart signals that there must not be too much weight; thus, the need is for an item that keeps warmth without adding bulk. Such discriminating behavior is difficult to do with the matrix methods used traditionally.

6.4 Real-Time Inference Performance

Although theoretical accuracy is highly valued, it would not mean much if the inference delay negatively impacts the user experience. Table 3 shows that the operational metrics for the fine-tuned model were recorded in a practical setting where the inference process was executed solely on the simulated consumer edge processor. The unoptimized model cannot be loaded because it occupies too much memory space, and inference becomes impossible. On the other hand, the 4-bit quantized version needs only 4.8 GB of storage capacity, yields the first token after 312 milliseconds, and operates at a constant speed of 24.5 tokens per second. Even complicated multi-turn questions get answered within reliable sub-second intervals. The speed combined with the classification and ranking capabilities validates the main argument: high-quality and independent product discovery is possible without sending the raw behavior data to central servers.

On-Device Real-Time Inference Benchmarks

Hardware State	TTFT (ms)	Speed (tokens/sec)	VRAM (GB)
Uncompressed (FP16)	Out of Memory	N/A	> 16.0
Edge-SLM (4-bit)	312	24.5	4.8

Conclusion

The use of centralized recommendation engines poses a compromise between prediction efficiency and privacy of customer data, an issue which has long been considered inevitable. The Edge-SLM paradigm presented in this paper defies the popular belief that there exists no solution to this issue. Through offloading the main computation task to user devices, the framework sets a clear boundary of behavioral data privacy by retaining computations on-device. Thanks to the use of 4-bit NormalFloat quantization, the proposed model remains compact enough to facilitate continuous computation, and in spite of its aggressive compression technique, semantic reasoning is possible. As a result, the model successfully interprets complex buying intents, a fact evidenced by the experimental results on the Amazon Product Data dataset. By converting sparse and time-structured data into the structured natural language format, the highly compressed model receives its functional representation, which allows for processing customers' sequential behavior. Through the Quantized Local Low-Rank Adaptation, edge devices can learn personalized probability distributions over items without server intervention. All required computations can be completed in consumer devices' limited physical memory. However, pure decentralization does have its drawback, since devices are programmed to create a limited representation of the world. This problem can be overcome using the federated aggregation algorithm. Instead of transmitting raw logs of interactions through the network, low-dimensional adapter matrices are sent. The central server calculates the mean difference in weights and sends the results back to users' devices. In practice, it is impossible to reconstruct the user's behavior based on such averaged mathematical operations. Behavioral data will stay local to the device; collective wisdom will continue to exist only as an abstraction. Such an approach allows the decentralized network to reach benchmarks comparable to those of standard cloud systems. The accuracy, precision, and hit ratio measures will be competitive, while inference time will not exceed one second, ensuring responsiveness of the interface. Several directions for future research are opened up by the current architecture design. First and foremost, the scope could be widened to include multi-modal sequential reasoning. Coupling the image embedding module with the existing semantic processor would allow the tracking of changes in aesthetic preference during interaction, namely the transition from general geometry to specific color palettes as users interact with the system. This combination of visual and semantic data surpasses the current architecture that only uses the textual pipeline, although the existing low rank adaptation process ensures a good foundation for expanding to the multi-modal case. The format agnostic injection of adapters is a problem of engineering, not

architecture; it is feasible but requires thematic adaptation. Overall, the results and design presented here provide a viable architecture for privacy-preserving conversational commerce. Precise autonomous discovery can be realized without widespread data centralization, and the difference in performance between local and cloud architectures is less than expected.

References

[1] Y. Koren and R. Bell, "Advances in collaborative filtering techniques for e-commerce," IEEE Trans. Knowl. Data Eng., vol. 33, no. 4, pp. 1425-1438, 2021.

[2] X. He et al., "Graph neural networks for modern recommendation," Proc. ACM SIGIR, pp. 124-134, 2021.

[3] J. Sun, "Latent vector mapping in sparse interaction matrices," Data Min. Knowl. Discov., vol. 36, no. 2, pp. 412-429, 2022.

[4] M. Zhang, "Addressing sparsity in user-item matching algorithms," IEEE Access, vol. 9, pp. 11204-11215, 2021.

[5] L. Chen et al., "Cold-start mitigation via textual attribute extraction," Inf. Process. Manage., vol. 59, no. 3, 2022.

[6] S. Wang and H. Liu, "Unstructured data parsing in hybrid recommendation," Knowl.-Based Syst., vol. 240, 2022.

[7] P. Li, "Transformer architectures in sequential recommendation tasks," IEEE Trans. Neural Netw. Learn. Syst., vol. 34, no. 6, pp. 2901-2914, 2023.

[8] W. Fan et al., "Recommender systems in the era of large language models," arXiv preprint arXiv:2307.02046, 2023.

[9] K. Hou et al., "Large language models as zero-shot conversational recommenders," Proc. ACM Web Conf., pp. 430-440, 2024.

[10] T. Lin, "Semantic sequencing for next-item prediction," ACM Trans. Inf. Syst., vol. 42, no. 1, 2024.

[11] Q. Yang, "Federated recommendation systems: Privacy and utility," IEEE Internet Comput., vol. 26, no. 5, pp. 45-52, 2022.

[12] H. Zheng et al., "Data sovereignty in cloud-based predictive modeling," Comput. Secur., vol. 125, 2023.

[13] F. Yuan, "Edge inference protocols for decentralized recommendation," IEEE Trans. Mob. Comput., vol. 23, no. 2, pp. 1502-1516, 2024.

[14] A. Kumar, "On-device processing frameworks for consumer hardware," J. Syst. Archit., vol. 142, 2023.

[15] E. Frantar et al., "GPTQ: Accurate post-training quantization for generative pre-trained transformers," Proc. ICLR, 2023.

[16] J. Lin et al., "AWQ: Activation-aware weight quantization for LLM compression and acceleration," Proc. MLSys, 2024.

[17] R. Zhao, "Memory footprint reduction in 4-bit integer formats," IEEE Trans. Comput., vol. 74, no. 1, pp. 88-101, 2025.

[18] D. Park, "Mobile processor constraints and neural network compression," Microprocess. Microsyst., vol. 104, 2024.

[19] C. Wu, "Context retention loss in quantized language models," Neurocomputing, vol. 580, 2024.

[20] T. Dettmers et al., "QLoRA: Efficient finetuning of quantized LLMs," Proc. NeurIPS, 2023.

[21] E. Hu et al., "LoRA: Low-rank adaptation of large language models," Proc. ICLR, 2022.

[22] S. Mangal, "Parameter-efficient adaptation on edge nodes," IEEE Trans. Artif. Intell., vol. 5, no. 3, 2024.

[23] Y. Kim, "Instruction tuning formulations for implicit feedback," Proc. EMNLP, pp. 560-571, 2023.

[24] J. Zhang et al., "Federated parameter-efficient fine-tuning for privacy-preserving NLP," Proc. ACL, pp. 1024-1035, 2024.

[25] X. Li, "Aggregating low-dimensional adapters in federated networks," IEEE Trans. Parallel Distrib. Syst., vol. 36, no. 4, pp. 1120-1132, 2025.