

PEAM: On-device Communication Agent for Personalized, Efficient and Autonomous Mobile Network Configuration

Haowen Wan^{1,2}, Shangzhuo Xie², Syue-Ju Syue³, Xianye Wu³, Runhua Li³, Qianqian Yang²

¹Polytechnic Institute, Zhejiang University

²College of Information Science and Electronic Engineering, Zhejiang University

³OPPO

Abstract

The growing deployment of compact large language models on smartphones is enabling autonomous on-device decision-making, such as emerging mobile GUI agents. However, many mobile decisions, such as network configuration when users switch apps or connectivity conditions change, must satisfy stringent latency constraints to preserve user experience. In this paper, we propose PEAM, an on-device LLM-based communication agent that autonomously generates personalized network-configuration rules executable within milliseconds. Specifically, PEAM first fine-tune the on-device LLM with a dataset constructed from manually annotated real-user data to integrate the communication knowledge into the local model. PEAM then operates in two phases. In the rule-generation phase, which is performed when the device is idle, such as overnight, PEAM analyzes historical network-configuration records in batches and reflects across batches to consolidate consistent evidence, reduce local bias, and derive robust configuration rules. To enable more accurate and stable rule generation over time, PEAM further constructs a user profile from high-confidence rules to capture the user’s network preferences and incorporates this profile into each rule-generation cycle. In the rule-execution phase, PEAM is invoked when the communication context changes, such as when users switch apps or connectivity conditions change, and searches the rule library to identify and execute the appropriate rule. Experiments demonstrate that PEAM, based on a fine-tuned 4B model, achieves 99.13% correct network configuration decisions, while 92.50% of the rules generated by PEAM are valid, outperforming the traditional expert-rule based method by 4.41% and 84.50%, respectively.

Introduction

Recent advances in large language models (LLMs) have significantly improved their reasoning, planning, and autonomous decision-making capabilities. Meanwhile, the growing deployment of compact models on mobile and edge devices has made LLM-based intelligence increasingly accessible in resource-constrained environments (Jiang et al. 2020; Chen et al. 2025; Zhang et al. 2024), as demonstrated by emerging mobile GUI agents (Wen et al. 2025; Zhang et al. 2025b; Yang et al. 2025). These trends open up new

Qianqian Yang is responsible for correspondence regarding this paper. Email: qianqianyang20@zju.edu.cn.

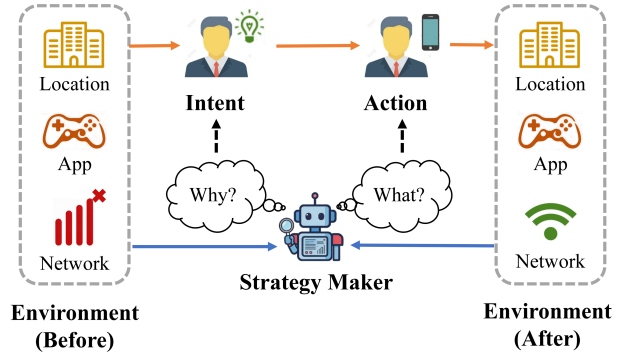


Figure 1: Modeling personalized mobile network configuration from historical decision records. PEAM infers why a user prefers a particular configuration and determines what configuration action should be executed.

opportunities for using on-device LLMs to support complex decision-making tasks in mobile systems. One representative task is mobile network configuration, which dynamically selects and adjusts communication resources and settings, such as access networks, WiFi access points, and SIM cards, according to application requirements, network conditions, device states, and user contexts. Unlike many of-line decision-making tasks, mobile network configuration directly affects user experience and often lies on a latency-critical execution path. A practical solution should therefore be **personalized**, because different users may prefer different configurations under similar communication conditions; **efficient**, because configuration decisions must often be completed within stringent latency constraints; and **autonomous**, because the device should learn and apply user preferences without repeated manual intervention.

Existing network-configuration methods mainly fall into three categories: expert-rule (ER)-based, supervised-learning (SL)-based, and reinforcement-learning (RL)-based methods. ER-based methods remain widely used in practical systems. They encode communication knowledge into hand-crafted constraints over metrics such as signal strength, uplink/downlink latency, and quality of service (QoS), and then map such conditions to switching actions (Sunita, Soni, and Baghla 2015; Miyim, Umar, and Kiyawa 2021; Wu et al.

2022). Although ER-based methods are efficient at runtime, their strategies are primarily determined by expert-designed heuristics, making them difficult to personalize according to individual user behavior. To improve adaptability, recent studies have explored SL-based and RL-based methods. SL-based methods learn correlations between communication observations and user actions from labeled data (Masi and Wac 2020; Paropkari, Thantharate, and Beard 2022; Peng et al. 2023), whereas RL-based methods formulate network switching as a sequential decision process and optimize policies through reward-driven interaction (Tan et al. 2010; Wang et al. 2024; Xie, Zhu, and Li 2023; Lu et al. 2025; Wu et al. 2024; He et al. 2023). However, in these methods, user preferences are still implicitly encoded in hand-crafted rules, training labels, reward functions, or model parameters. As a result, they are difficult to interpret, update, and continually adapt as user behavior evolves.

More recently, researchers have begun to apply LLMs to network-configuration tasks. Existing studies have explored LLM-based solutions for WiFi roaming (Lee, Lu, and Doppler 2025; Nagai et al. 2026) and cellular handover (Li et al. 2026; Jeong et al. 2026), demonstrating the potential of LLMs to interpret heterogeneous communication observations and generate corresponding decisions. However, these approaches largely employ LLMs as reactive decision engines that directly map current observations to configuration actions. They do not systematically distill long-term user behavior into persistent, interpretable configuration rules, and they still rely on LLM inference during the decision-making path. Consequently, existing LLM-based methods struggle to simultaneously achieve personalization, efficiency, and autonomy for on-device mobile network configuration.

To address these limitations, we propose PEAM, an on-device LLM-based communication agent for personalized, efficient, and autonomous mobile network configuration. The key idea of PEAM is to decouple complex LLM reasoning from latency-critical execution. Instead of invoking the LLM for every online configuration decision, PEAM performs LLM-based reasoning offline when the device is idle, distills historical user behavior into persistent condition-action rules, and executes these rules online through lightweight retrieval. As illustrated in Figure 1, PEAM models personalized network configuration as two coupled problems: understanding why a user prefers a particular configuration from historical records and determining what configuration action should be executed under the current communication context.

This design introduces two key challenges on resource-constrained devices. First, **how can a compact on-device LLM reliably infer user intents and generate configuration rules from long and heterogeneous historical records?** General-purpose LLMs lack communication-specific knowledge. Moreover, on-device compact models usually have limited context capacity. PEAM addresses this challenge by constructing a task-specific dataset from manually annotated real-user network-switching records and performing supervised fine-tuning to integrate communication knowledge into the local model. PEAM then partitions long histories into coherent batches, analyzes each batch independently, and performs cross-batch reflection over the inter-

mediate outputs. This divide-and-consolidate process keeps each inference within the model’s context window while integrating distributed evidence, reducing locally biased conclusions, and producing more reliable configuration rules.

Second, **how can we continually capture evolving user preferences and improve the accuracy and stability of rule generation over time?** Static rules cannot adequately represent preferences that change with user behavior. PEAM therefore updates rule confidence according to whether the user accepts or rejects the configuration proposed by the retrieved rule, and discards rules whose confidence falls below a predefined threshold. Beyond maintaining individual rules, PEAM constructs a semantic user profile from high-confidence rules. This profile summarizes both scenario-specific preferences, such as preferred configurations under particular applications, times, locations, and network conditions, and broader behavioral tendencies, such as sensitivity to data usage and preference for connection quality. Rather than directly inheriting the previous profile, PEAM periodically regenerates the profile from the latest reliable rules and incorporates it into the next rule-generation cycle, providing an up-to-date preference context for more accurate and stable personalization.

The main contributions of this paper are summarized as follows:

- We propose PEAM, an on-device communication agent that distills offline LLM reasoning into persistent condition-action rules and executes them through lightweight online retrieval. This design decouples complex deliberation from latency-critical execution, providing a practical architecture for LLM-based decision-making on resource-constrained devices.
- We design a divide-and-consolidate reasoning mechanism that decomposes long, heterogeneous interaction histories into coherent batches and consolidates distributed evidence across batches, enabling compact on-device LLMs to generate personalized rules beyond a single context window.
- We develop a rule-based hybrid memory that combines executable rule memory with semantic preference memory. By reconstructing user profiles from high-confidence rules and feeding them into subsequent reasoning cycles, PEAM supports interpretable and behavior-based personalization over time.
- We construct benchmarks based on real-user and simulated-user records for comprehensive evaluation. Extensive results, together with a smartphone deployment of a quantized 4B model, demonstrate PEAM’s effectiveness and practical on-device feasibility.

Related Work

On-device LLM-based Agent On-device LLM agents have recently attracted increasing attention for enabling autonomous perception, reasoning, and interaction directly on mobile devices. Advances in mobile inference and collaborative model customization provide important technical foundations for such agents. For example, MNN (Jiang et al.

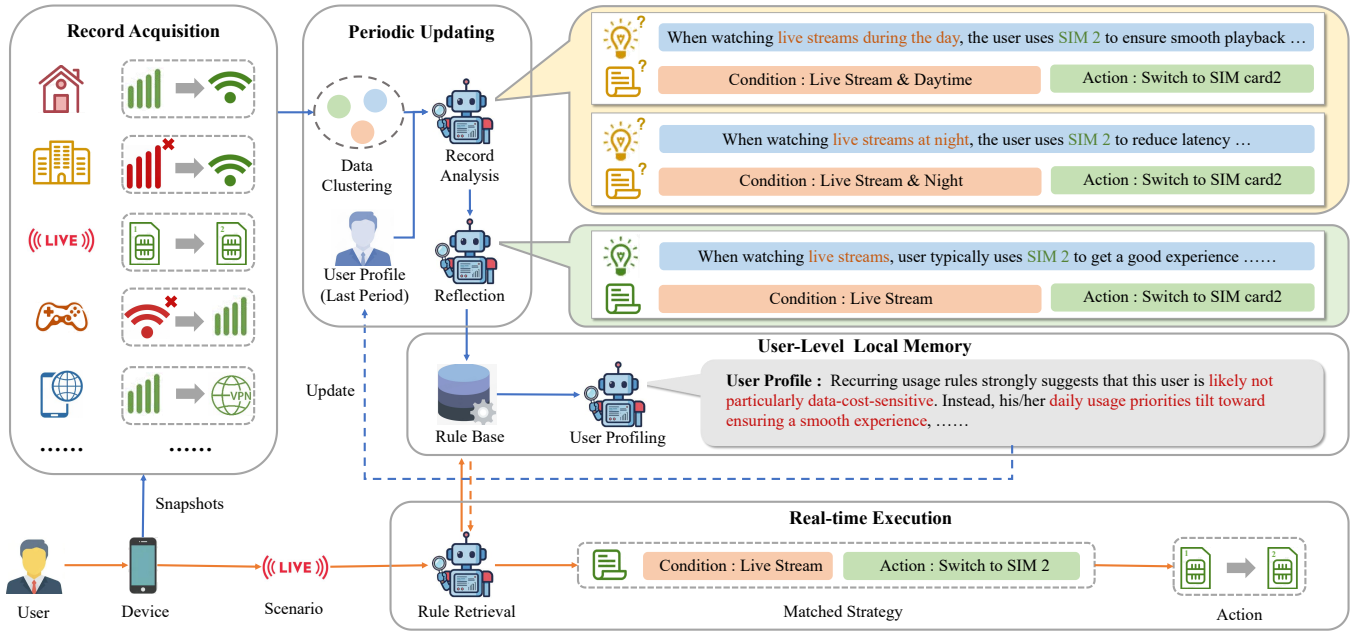


Figure 2: Overview of PEAM. The framework contains four stages: (1) Record Acquisition collects user-initiated configuration records; (2) Periodic Update performs clustering, batch-wise record analysis, and cross-batch reflection to generate personalized rules; (3) User-Level Local Memory maintains the rule base and user profile; and (4) Real-Time Execution retrieves and executes matched rules without invoking the LLM.

2020) supports efficient neural network inference across heterogeneous mobile hardware, while Confidant (Chen et al. 2025) enables collaborative customization of Transformer-based LLMs across multiple mobile devices. At the application level, mobile GUI automation has become a representative direction for on-device agents. AutoDroid (Wen et al. 2025) formulates GUI automation as code generation, allowing an on-device SLM to generate executable programs for multi-step operations. AgentCPM-GUI (Zhang et al. 2025b) develops a mobile-use agent through GUI grounding, supervised fine-tuning, and reinforcement fine-tuning, while Ferret-UI Lite (Yang et al. 2025) explores a compact end-to-end agent for interacting with mobile, web, and desktop interfaces. These studies demonstrate the growing capability of compact language models to support complex autonomous interactions on resource-constrained devices. In contrast, PEAM focuses on latency-sensitive communication decision-making, where on-device LLM reasoning must be translated into executable network-configuration actions.

LLM-based Network Configuration Recent advances in LLMs have also motivated their application to network configuration and mobility management. Lee et al. (Lee, Lu, and Doppler 2025) deployed an on-device LLM to interpret WiFi roaming observations and generate corresponding control actions. Nagai et al. (Nagai et al. 2026) further investigated prompt refinement for real-time WLAN handover and evaluated their method using a quantized Llama-3.1-8B model on edge devices. Beyond WiFi roaming, Li et al. (Li et al. 2026) developed a GPT-2-based framework to model the temporal evolution of reference signal received power, supporting proactive cellular handover. Jeong et al. (Jeong et al.

2026) incorporated visual environmental context through a multi-modal LLM to estimate channel capacity along candidate user paths and assist mobility decisions. These studies show that language models can interpret heterogeneous communication observations and support context-aware network decisions.

Methodology

As illustrated in Figure 2, PEAM consists of four modules: Record Acquisition, Periodic Update, User-Level Local Memory, and Real-Time Execution. We consider personalized mobile network configuration over a sequence of update periods indexed by t . In each period, PEAM first collects user-initiated configuration records, then performs offline LLM-based rule generation when the device is idle, updates the local rule and profile memories, and finally executes matched rules online without invoking the LLM.

Record Acquisition

The *Record Acquisition* module runs continuously in the background and monitors user-initiated network-configuration operations. Once an operation is detected, PEAM records the communication context immediately before the operation, the configuration action selected by the user, and the resulting context after the operation. The records collected during the t -th update period are represented as

$$D_t = \{d_i^t\}_{i=1}^{N_t}, \quad d_i^t = (x_i^{t,-}, a_i^t, x_i^{t,+}), \quad (1)$$

where N_t denotes the number of collected records; $x_i^{t,-}$ and $x_i^{t,+}$ denote the communication contexts immediately be-

fore and after the i -th operation, respectively; and a_i^t denotes the configuration action selected by the user. Each context contains information such as time, location, foreground application, device configuration, and network conditions. All records are stored locally until the next periodic update. The collected record set D_t , together with the user profile p_{t-1} generated in the previous period, serves as the input to Periodic Updating, which is then used to infer the user's underlying decision intents and generate personalized configuration rules.

Periodic Updating

Data Clustering Records collected within an update period are heterogeneous and may exceed the context capacity of the on-device LLM. PEAM therefore groups D_t into batches using feature-weighted HDBSCAN(Campello et al. 2015). For two records d_a and d_b , the dissimilarity of the k -th feature and their overall weighted Gower distance are defined as

$$\delta_k(d_a, d_b) = \begin{cases} \frac{|x_a^k - x_b^k|}{R_k}, & k \in \mathcal{F}_{\text{num}}, \\ \mathbb{I}(x_a^k \neq x_b^k), & k \in \mathcal{F}_{\text{cat}}, \end{cases} \quad (2)$$

$$\text{dist}_{\mathbf{W}}(d_a, d_b) = \frac{\sum_{k=1}^m w_k \delta_k(d_a, d_b)}{\sum_{k=1}^m w_k}.$$

where R_k is the range of the numerical feature k , $\mathcal{F}_{\text{num}}$ and $\mathcal{F}_{\text{cat}}$ denote the numerical and categorical feature sets, and w_k controls the contribution of each feature. The feature weights $\mathbf{W}$ and HDBSCAN hyperparameters Θ are jointly optimized once on labeled training data using the Tree-structured Parzen Estimator(Akiba et al. 2019). Let $\mathbf{c}^*$ denote the ground-truth intent labels and $\hat{\mathbf{c}} \equiv \hat{\mathbf{c}}(\mathbf{W}, \Theta)$ denote the clustering assignments. The optimization objective is

$$Q(\mathbf{W}, \Theta) = \lambda_1 \text{ARI}(\mathbf{c}^*, \hat{\mathbf{c}}) + \lambda_2 \text{NMI}(\mathbf{c}^*, \hat{\mathbf{c}}), \quad (3)$$

$$(\mathbf{W}^*, \Theta^*) = \arg \max_{\mathbf{W}, \Theta} Q(\mathbf{W}, \Theta).$$

where $\lambda_1 + \lambda_2 = 1$. The optimized parameters are fixed after training and reused during deployment. Applying them to D_t yields a cluster set $\mathcal{C}_t = \{C_{t,j}\}_{j=1}^{M_t}$, where each cluster forms an independent analysis batch and sparse observations are treated as noise.

Record Analysis Although clustering reduces the input length and groups records with similar contexts, raw device logs remain difficult for a compact LLM to interpret directly. PEAM therefore converts each cluster into a concise semantic representation before inference. Stable device attributes, such as SIM information and data-usage settings, are presented once as shared configuration context. Dynamic attributes, such as time, location, and network conditions, are summarized using the mean for numerical features and the mode for categorical features. Each contextual transition is then represented as $K : V^- \rightarrow V^+$, where K denotes an attribute and V^- and V^+ denote its values before and after the configuration operation, respectively.

As illustrated in Figure 3, the record-analysis prompt combines semantic records with the content constraint Γ_c , the

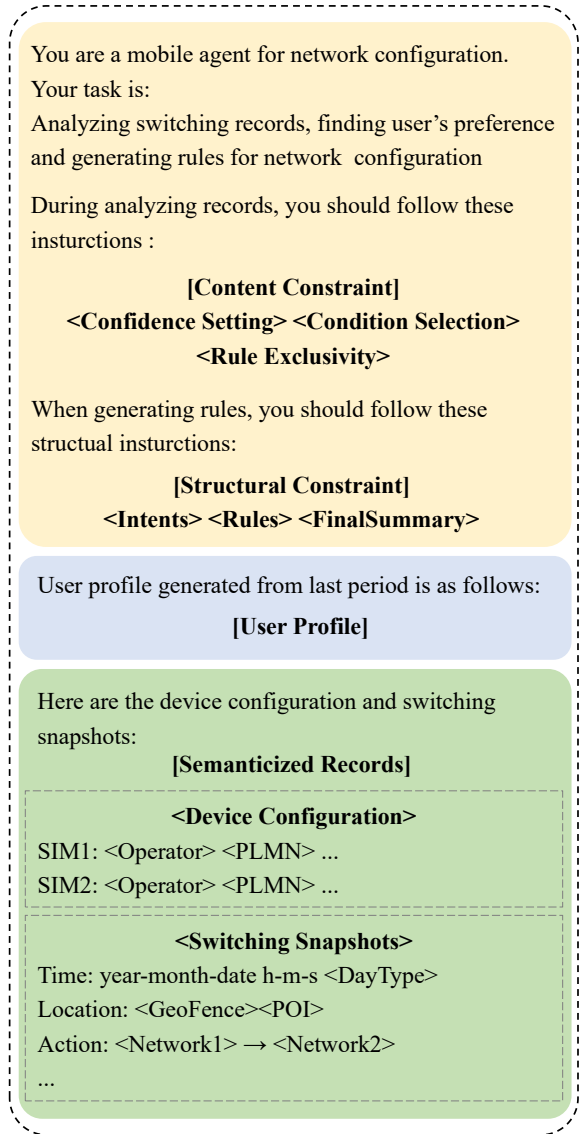


Figure 3: Prompt template for Record Analysis. The yellow boxes contain fixed task and output constraints, the blue box contains the user profile generated in the previous update period, and the green box contains the semantic representations of the current records.

structural constraint Γ_s , and the previous user profile p_{t-1} . The content constraint guides the model to identify shared behavioral patterns, infer the most plausible user intent and rationale, determine corresponding decision conditions, and verify the compatibility of the resulting configuration action. The structural constraint Γ_s enforces a consistent output comprising the inferred intents, confidence scores, rationales, condition-action rules, and a summary. For the j -th cluster, the record-analysis process is formulated as

$$(\hat{\mathcal{I}}_{t,j}, \hat{\mathcal{R}}_{t,j}) = \text{LLM}(\Gamma_c, \Gamma_s, \mathbf{p}_{t-1}, \text{Sem}(C_{t,j})). \quad (4)$$

where $\text{Sem}(\cdot)$ denotes semantic transformation, and $\hat{\mathcal{I}}_{t,j}$ and

$\hat{\mathcal{R}}_{t,j}$ denote the locally inferred intents and candidate configuration rules, respectively.

Reflection Clustering cannot guarantee that all records associated with the same underlying intent fall into a single batch. As a result, batch-level outputs may contain incomplete intents, redundant rules, or conflicting actions caused by limited local evidence. PEAM address this problem through cross-batch reflection (Shinn et al. 2023; Madaan et al. 2023). Guided by a fixed reflection constraint Γ_r , the LLM identifies intents and rules with shared semantics across batches, merges recurring strategies and complementary conditions to consolidate distributed evidence. It further examines candidate rules with overlapping conditions but incompatible actions, suppressing conclusions supported only by isolated batches. Semantically duplicated rules are also eliminated to prevent fragmented observations from being misinterpreted as independent user preferences. The cross-batch reflection process is formulated as

$$(\mathcal{I}_t, \mathcal{R}_t) = \text{LLM} \left(\Gamma_r, \{(\hat{\mathcal{I}}_{t,j}, \hat{\mathcal{R}}_{t,j})\}_{j=1}^{M_t} \right), \quad (5)$$

where $\mathcal{I}_t$ and $\mathcal{R}_t$ denote the consolidated user intents and configuration rules for the current update period.

User-Level Local Memory

PEAM maintains a user-level local memory consisting of a personalized rule base and a semantic user profile. Each rule r contains a condition–action pair and a confidence score q_r . When a rule is retrieved for execution, PEAM asks the user whether to accept the proposed configuration. Acceptance increases the rule confidence, whereas rejection decreases it. Rules whose confidence falls below a predefined threshold are discarded. This feedback mechanism allows PEAM to retain reliable strategies while removing rules that no longer match the user’s preferences. Detailed rule states and transition mechanisms are provided in the Appendix.

To summarize user preferences beyond individual condition-specific rules, PEAM periodically constructs a semantic user profile from high-confidence rules. The profile captures both scenario-specific preferences, such as preferred configurations under particular applications, times, locations, and network conditions, and broader behavioral tendencies, such as sensitivity to data usage and preference for connection quality. Let B_t^{hc} denote the high-confidence rules maintained during the t -th update period. Guided by a fixed profile-generation constraint Γ_p , PEAM generates the profile as

$$p_t = \text{LLM} (\Gamma_p, B_t^{hc}), \quad (6)$$

where p_t denotes the updated user profile. PEAM regenerates the profile from the latest reliable rules and incorporates it into the next offline rule-generation cycle, providing higher-level preference context for subsequent record analysis.

Real-Time Execution

When the communication context changes, PEAM captures the current application, temporal, spatial, device-configuration, and network conditions, and retrieves applicable active rules from the local rule base. Since these rules are

generated in advance during periodic updates, the real-time execution path does not invoke the LLM. Instead, PEAM directly compares the current context with the conditions specified by each rule and retrieves the rules whose conditions are satisfied. If multiple rules match, PEAM resolves the conflict using a two-level priority scheme. It first selects the rule with the largest number of matched condition predicates, since this rule provides the more specific description of the current context. If multiple rules contain the same number of matched conditions, the rule with the highest confidence is selected. Finally, PEAM applies the action associated with the selected rule to the device. This retrieval-based design avoids costly real-time model inference and enables millisecond-level online decision-making.

Evaluation

Datasets

We construct three mutually disjoint datasets for model training and evaluation: a simulated-user dataset, a real-user dataset, and a task-level dataset.

Simulated-User Dataset We first collected one month of network-configuration records from 51 real users, together with additional records obtained in controlled laboratory settings. These records are manually annotated to identify 20 high-frequency communication intents and their corresponding configuration templates. Based on these annotations, we generate 300 simulated users with diverse spatiotemporal contexts, applications, device configurations, and network conditions, and use DeepSeek-V3.2 (Liu et al. 2025) to annotate the simulated data. Among these users, 200 are used exclusively to optimize the feature weights and HDBSCAN hyperparameters, while the remaining 100 are reserved for end-to-end evaluation.

Real-User Dataset We further collect approximately two months of network-configuration records from 82 held-out real users. For each user, the records are chronologically divided into two periods. The first period is used to generate personalized rules, while the second is used to evaluate whether these rules correctly predict subsequent user decisions.

Task-Level Dataset We construct a dialogue dataset covering Record Analysis, Cross-Batch Reflection, and User Profiling. The training set contains 5,000 samples, including 2,000 for Record Analysis, 2,000 for Cross-Batch Reflection, and 1,000 for User Profiling. The test set contains 300 held-out samples, with 100 samples for each task.

Baselines and Implementation Details

Baselines We compare PEAM with two groups of baselines. The first is an expert-rule method consisting of 11 manually designed rules for common network-switching scenarios. Each rule specifies when a switch should be triggered, while the target action for each user is determined by the configuration most frequently observed in that user’s historical records. The second group includes three general-purpose LLMs: Qwen3.5-Flash (Qwen-Team 2026), Gemini-3-Flash (Google-DeepMind 2025), and GPT-5.2 (OpenAI 2025). In the end-to-end evaluation, these models directly

Method	Simulated User Dataset				Real User Dataset			Model Pricing	
	REA	RCR	TRR	JMS	REA	RCR	TRR	Input	Output
Expert Rules	<u>94.72</u>	6.06	8.00	22.15	48.68	2.92	2.56	-	-
Qwen3.5-Flash	86.58	25.93	56.86	51.90	74.99	12.49	51.35 (19/37)	0.028	0.278
Gemini-3-Flash	92.24	54.79	80.19	<u>64.55</u>	<u>72.51</u>	15.47	49.02 (25/51)	0.500	3.000
GPT-5.2	87.56	<u>57.25</u>	<u>87.47</u>	55.72	71.87	<u>20.96</u>	38.89 (14/36)	1.750	14.00
PEAM	99.13	60.58	92.50	81.32	72.34	25.86	38.10 (24/63)	-	-
Improvement	+4.41	+3.33	+5.03	+16.77	-	+4.90	-	-	-

Table 1: End-to-end performance on the simulated-user and real-user datasets, together with API pricing for the cloud-based LLM baselines. REA, RCR, and TRR are reported as percentages. Values in parentheses denote the number of triggered rules over the total number of generated rules. The Improvement row reports the absolute gain over the strongest baseline for each metric. Input and output prices are reported in USD per million tokens. Bold and underlined values denote the best and second-best results, respectively.

Method	Record Analysis			Reflection			User Profiling		
	SS	JMS	IT	SS	JMS	IT	SS	JMS	IT
Qwen3.5-Flash	<u>0.7451</u>	74.35	<u>8.73</u>	0.6577	66.91	7.34	<u>0.8726</u>	<u>87.98</u>	2.57
Gemini-3-Flash	0.6973	79.69	4.74	0.6433	64.88	17.31	0.8593	85.54	<u>2.58</u>
GPT-5.2	0.7418	<u>81.06</u>	13.80	<u>0.8377</u>	<u>73.05</u>	<u>9.76</u>	0.8582	89.89	6.59
PEAM	0.8361	86.68	15.64	0.8955	84.12	13.98	0.9112	87.10	5.31
Improvement	+0.0910	+5.62	-	+0.0578	+11.07	-	+0.0386	-	-

Table 2: Task-level performance on Record Analysis, Cross-Batch Consolidation, and User Profiling. IT is reported in seconds. The Improvement row reports the absolute gain over the strongest baseline for each metric. Bold and underlined values denote the best and second-best results, respectively.

process complete historical records and generate personalized configuration rules under the same output format as PEAM.

Implementation Details PEAM uses Qwen3-4B as its default backbone and is fine-tuned with LoRA (Hu et al. 2022) on a single NVIDIA A100 GPU. For the deployment analysis, we additionally train an AndesVL-4B-based variant (AndesVL-Team 2025). Its full-precision version is evaluated on the A100 server, while its W4A16-quantized version is deployed on an OPPO Find X9 Pro smartphone equipped with the MediaTek Dimensity 9500. Detailed configurations are provided in the Appendix.

Evaluation Metrics

We evaluate different methods from three perspectives: rule effectiveness, generation quality, and efficiency.

Rule Effectiveness We use three metrics to evaluate the effectiveness of rules generated from historical records. Rule Execution Accuracy (REA) measures the proportion of correct actions among all triggered rule executions. Rule Coverage Ratio (RCR) measures the proportion of subsequent-period records that trigger at least one generated rule, reflecting the behavioral coverage of the rule set. Triggered Rule Ratio (TRR) measures the proportion of generated rules that are triggered by at least one record in the subsequent evaluation period, indicating whether a method produces excessive inactive rules.

Generation Quality We use Semantic Similarity (SS) and Judge Model Score (JMS) to measure the discrepancy be-

tween generated outputs and their reference labels, where higher scores indicate better generation quality. SS evaluates embedding-level semantic similarity using Qwen3-Embedding-0.6B (Zhang et al. 2025a), while JMS uses DeepSeek-V3.2 to assign a score from 0 to 100 based on their overall consistency. The evaluation criteria for these tasks are summarized in the Appendix.

Efficiency Inference Time (IT) measures the time required to complete a single inference for a given task. For cloud-based LLM baselines, we also report the input and output prices per million tokens. Since PEAM performs inference locally, it incurs no external API cost.

Experiments and Results

Table 1 reports the end-to-end rule performance on the simulated-user and real-user datasets. On the simulated-user dataset, PEAM achieves the best REA, RCR, TRR, and JMS, demonstrating a strong balance among execution accuracy, behavioral coverage, and rule usability. In contrast, the expert-rule method attains high accuracy only within a small set of predefined scenarios, as reflected by its substantially lower RCR and TRR, while the general-purpose LLMs show less consistent performance across metrics. On the real-user dataset, all methods degrade because of more heterogeneous and less repetitive user behaviors. Nevertheless, PEAM achieves the highest RCR of 25.86%, corresponding to a 4.90% improvement over the strongest baseline, while maintaining an REA of 72.34% that remains comparable to the cloud-based LLMs. Its lower TRR should be interpreted

Method	Record Analysis			Reflection			User Profiling			Real-time Execution Retrieval Time
	SS	JMS	IT	SS	JMS	IT	SS	JMS	IT	
PEAM	0.8413	84.06	35.53	0.8563	75.21	32.62	0.8952	84.64	11.87	-
PEAM-Q	0.8132	80.80	33.17	0.8608	72.09	59.29	0.7991	64.56	8.950	0.5730

Table 3: Task-level performance of AndesVL-based PEAM under two deployment configurations. PEAM denotes the full-precision model evaluated on an NVIDIA A100 GPU, whereas PEAM-Q denotes the quantized model deployed on a smartphone. Inference time (IT) is reported in seconds, while retrieval time is reported in milliseconds.

Variant	REA	RCR	TRR	JMS
PEAM	99.13	60.58	92.50	81.32
- w/o Clustering	98.61	49.57	91.79	65.80
- w/o Reflection	92.44	67.96	43.21	70.43
- w/o User Profiling	97.36	61.31	85.26	75.27

Table 4: Ablation results for the major components of PEAM. REA, RCR, and TRR are reported as percentages.

together with the size of the generated rule library: PEAM produces 63 rules, of which 24 are triggered in the subsequent period, nearly matching Gemini-3-Flash despite maintaining a broader candidate set. This lower TRR mainly reflects PEAM’s broader rule exploration, while inactive rules do not affect user experience. Moreover, unlike cloud-based LLMs, PEAM incurs no recurring API cost, yielding a favorable trade-off among accuracy, coverage, and deployment cost.

Table 2 evaluates PEAM on the three offline reasoning tasks: Record Analysis, Cross-Batch Reflection, and User Profiling. PEAM achieves the highest semantic similarity across all three tasks and the best judge scores for Record Analysis and Cross-Batch Reflection, indicating stronger alignment with reference outputs when interpreting communication records and consolidating intermediate results. The clear advantage in Cross-Batch Reflection supports the effectiveness of task-specific fine-tuning for merging distributed evidence and resolving inconsistent rules. For User Profiling, PEAM obtains the best semantic similarity and remains competitive in judge score, suggesting that its generated profiles closely capture reference user preferences. Although PEAM introduces moderate offline inference overhead, these tasks are performed during periodic updates, which occur when the mobile device is idle, and therefore do not affect latency-critical rule execution.

Table 3 compares PEAM under full-precision server inference and W4A16-quantized smartphone deployment. The quantized model largely preserves the quality of Record Analysis and Cross-Batch Reflection, achieving SS values of 0.8132 and 0.8608, respectively. Its reflection similarity is slightly higher than that of the full-precision model, although the corresponding JMS decreases, suggesting that quantization preserves overall semantics but may affect finer-grained generation quality. User Profiling exhibits a more noticeable degradation, indicating that higher-level preference abstraction is more sensitive to quantization than record-level analysis and rule consolidation. Offline inference latency also varies across tasks, showing that quantization does not pro-

vide a uniform speedup across the entire pipeline. Most importantly, PEAM-Q completes rule retrieval in only 0.5730 ms on the smartphone. This result directly validates effectiveness of PEAM’s separation of offline LLM deliberation and online rule execution in real-time network configuration.

Table 4 evaluates the contribution of each major component in PEAM. Removing clustering reduces RCR from 60.58% to 49.57% and decreases JMS from 81.32 to 65.80, while REA and TRR remain relatively high. This is due to that clustering primarily improves evidence organization, behavioral coverage, and generation quality by grouping coherent records before LLM inference. Removing Cross-Batch Reflection increases RCR to 67.96%, but substantially decreases REA, TRR, and JMS to 92.44%, 43.21%, and 70.43, respectively. The increased coverage therefore comes at the cost of many redundant, conflicting, or weakly supported rules, confirming that reflection acts as an essential reliability filter. Removing User Profiling leaves coverage nearly unchanged, but reduces REA, TRR, and JMS to 97.36%, 85.26%, and 75.27. This pattern shows that user profiling mainly improves the consistency and personalization of generated rules rather than simply increasing their number. Overall, the three components provide complementary benefits and jointly contribute to PEAM’s performance.

Conclusion

In this paper, we presented PEAM, an on-device communication agent for personalized mobile network configuration. PEAM decouples LLM-based deliberation from latency-critical execution by distilling historical user decisions into persistent condition-action rules during offline updates and retrieving matched rules online without invoking the LLM. To support compact on-device reasoning over long and heterogeneous histories, PEAM combines batch-wise analysis with cross-batch reflection, while a rule-grounded hybrid memory connects executable rules with semantic user profiles for cross-cycle personalization. Evaluations on task-level, simulated-user, and held-out real-user benchmarks demonstrate competitive performance in rule execution, coverage, triggering, and generation quality, while smartphone deployment of a quantized 4B model confirms its practical on-device feasibility. Overall, PEAM demonstrates that on-device LLM agents can support latency-critical mobile system decisions by shifting complex reasoning offline and executing personalized decisions through efficient local rules.

References

- Akiba, T.; Sano, S.; Yanase, T.; Ohta, T.; and Koyama, M. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2623–2631. ACM.
- AndesVL-Team. 2025. AndesVL Technical Report: An Efficient Mobile-side Multimodal Large Language Model. arXiv:2510.11496.
- Campello, R. J. G. B.; Moulavi, D.; Zimek, A.; and Sander, J. 2015. Hierarchical Density Estimates for Data Clustering, Visualization, and Outlier Detection. *ACM Transactions on Knowledge Discovery from Data*, 10(1): 1–51.
- Chen, Y.; Yan, Y.; Ge, S.; Qin, Y.; Zheng, Y.; Yang, Q.; He, S.; Shi, Z.; Chen, J.; and Shu, Y. 2025. Confidant: Customizing Transformer-based LLMs via Collaborative Training on Mobile Devices. In *Proceedings of the 31st Annual International Conference on Mobile Computing and Networking*, 483–497. ACM.
- Google-DeepMind. 2025. Gemini3 Flash Doc. <https://docs.cloud.google.com/gemini-enterprise-agent-platform/models/gemini/3-flash>. Accessed:2026-7-10.
- He, J.; Xiang, T.; Wang, Y.; Ruan, H.; and Zhang, X. 2023. A Reinforcement Learning Handover Parameter Adaptation Method Based on LSTM-Aided Digital Twin for UDN. *Sensors*, 23(4): 2191–2211.
- Hu, E. J.; Shen, Y.; Wallis, P.; Zhu, Z. A.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2022. LoRA: Low-rank adaptation of large language models. In *Proceedings of the 10th International Conference on Learning Representations*.
- Jeong, S.; Shin, S.; Kim, S.; Wu, J.; and Shim, B. 2026. Large Multimodal Model-Based Environment-Aware Mobility Management. arXiv:2607.09795.
- Jiang, X.; Wang, H.; Chen, Y.; Wu, Z.; Wang, L.; Zou, B.; Yang, Y.; Cui, Z.; Cai, Y.; Yu, T.; et al. 2020. MNN: A Universal and Efficient Inference Engine. *Proceedings of Machine Learning and Systems*, 2: 1–12.
- Lee, J.; Lu, Y.; and Doppler, K. 2025. On-Device LLM for Context-Aware Wi-Fi Roaming. arXiv:2505.04174.
- Li, W.; Chen, W.; Ai, B.; Zhang, Y.; Peng, Y.; and Matthaiou, M. 2026. LLM-Empowered Wireless Channel Quality Prediction for Reliable Cell Handover. *IEEE Wireless Communications*, 1–8.
- Liu, A.; Mei, A.; Lin, B.; Xue, B.; Wang, B.; Xu, B.; Wu, B.; Zhang, B.; Lin, C.; Dong, C.; et al. 2025. Deepseek-v3. 2: Pushing the frontier of open large language models. arXiv:2512.02556.
- Lu, Y.; Qi, J.; Zhang, S.; Luo, G.; Zhu, A.; Wu, J.; and Qian, Z. 2025. ABUV: Adaptive Bitrate and Upsampling for Video Streaming on Mobile Devices. *Computer Networks*, 257: 1–11.
- Madaan, A.; Tandon, N.; Gupta, P.; Hallinan, S.; Gao, L.; Wiegrefe, S.; Alon, U.; Dziri, N.; S., P.; Yang, Y.; Gupta, S.; Majumder, B. P.; Hermann, K.; Welleck, S.; Yazdanbakhsh, A.; and Clark, P. 2023. Self-Refine: Iterative Refinement with Self-Feedback. *Advances in Neural Information Processing Systems*, 36: 46534–46594.
- Masi, A. D.; and Wac, K. 2020. Towards Accurate Models for Predicting Smartphones Applications’ QoE with Data from A Living Lab Study. *Quality and User Experience*, 5(10): 1–18.
- Miyim, A. M.; Umar, K.; and Kiyawa, A. I. 2021. A Fuzzy-Based AHP Approach to Handover Decision for Mobility Management in Heterogeneous Wireless Network. *SLU Journal of Scientific Research and Management*, 2(2): 46–57.
- Nagai, S.; Le, T. M.; Le, N. P.; Nguyen, K.; and Sekiya, H. 2026. Toward Real-Time On-Device WLAN Handover Control via Prompt Refinement. In *Proceedings of IEEE Conference on Computer Communications*, 1–5. IEEE.
- OpenAI. 2025. GPT-5.2 System Card. <https://deploymentsafety.openai.com/gpt-5-2>. Accessed:2026-7-10.
- Paropkari, R. A.; Thantharate, A.; and Beard, C. 2022. Deep-Mobility: A Deep Learning Approach for an Efficient and Reliable 5G Handover. In *Proceedings of International Conference on Wireless Communications Signal Processing and Networking*, 244–250. IEEE.
- Peng, F.; Lu, B.; Song, L.; Xie, R.; Liu, Y.; and Chen, Y. 2023. PACC: Perception Aware Congestion Control for Real-time Communication. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, 978–983. IEEE.
- Qwen-Team. 2026. Qwen3.5: Accelerating Productivity with Native Multimodal Agents. <https://qwen.ai/blog?id=qwen3.5>. Accessed:2026-7-10.
- Shinn, N.; Cassano, F.; Gopinath, A.; Narasimhan, K.; and Yao, S. 2023. Reflexion: Language Agents with Verbal Reinforcement Learning. *Advances in Neural Information Processing Systems*, 36: 8634–8652.
- Sunita, E.; Soni, E. S.; and Baghla, E. S. 2015. Fuzzy Logic Based Handover Decision and Necessity Estimation scheme for Heterogeneous Wireless Network. *International Journal of Scientific Research and Management*, 3(6): 3234–3238.
- Tan, H.; Li, Y.; Chen, Y.; Tan, L.; and Li, Q. 2010. A Novel Access Network Selection Scheme Using Q-learning Algorithm for Cognitive Terminal. In *Proceedings of 5th International ICST Conference on Communication and Networking in China*, 1–5. IEEE.
- Wang, W.; Yang, H.; Li, S.; Liu, X.; and Wan, Z. 2024. Adaptive UE Handover Management with MAR-Aided Multivariate DQN in Ultra-Dense Networks. *Journal of Network and Systems Management*, 33(17): 1–17.
- Wen, H.; Tian, S.; Pavlov, B.; Du, W.; Li, Y.; Chang, G.; Zhao, S.; Liu, J.; Liu, Y.; Zhang, Y.; et al. 2025. Autodroid-v2: Boosting slm-based gui agents via code generation. In *Proceedings of the 23rd Annual International Conference on Mobile Systems, Applications and Services*, 223–235.
- Wu, B.; Li, T.; Luo, C.; Quyang, C.; Du, X.; and Wang, F. 2022. AutoPlex: Inter-Session Multiplexing Congestion Control for Large-Scale Live Video Services. In *Proceedings of ACM SIGCOMM Workshop on Network-Application Integration*, 1–6. ACM.

Wu, W.; Yuan, J.; Ma, S.; and Yang, M. 2024. AP-assisted Adaptive Video Streaming in Wireless Networks with High-density Clients. *Computer Communications*, 219: 53–63.

Xie, J.; Zhu, B.; and Li, C. 2023. Research of 5G HUDN Network Selection Algorithm Based on Dueling-DDQN. *EURASIP Journal on Wireless Communication Networks*, 2023(113): 1–27.

Yang, Z.; Dou, Z.-Y.; Feng, D.; Huang, F.; Nguyen, A.; You, K.; Attia, O.; Yang, Y.; Feng, M.; Zhang, H.; et al. 2025. Ferret-ui lite: Lessons from building small on-device gui agents. arXiv:2509.26539.

Zhang, S.; Ma, Y.; Fang, L.; Jia, H.; D’Alfonso, S.; and Kostako, V. 2024. Enabling On-Device LLMs Personalization with Smartphone Sensing. In *Proceedings of ACM International Joint Conference on Pervasive Ubiquitous Computation*, 186–190. ACM.

Zhang, Y.; Li, M.; Long, D.; Zhang, X.; Lin, H.; Yang, B.; Xie, P.; Yang, A.; Liu, D.; Lin, J.; Huang, F.; and Zhou, J. 2025a. Qwen3 Embedding: Advancing Text Embedding and Reranking Through Foundation Models. arXiv:2506.05176.

Zhang, Z.; Lu, Y.; Fu, Y.; Huo, Y.; Yang, S.; Wu, Y.; Si, H.; Cong, X.; Chen, H.; Lin, Y.; Xie, J.; Zhou, W.; Xu, W.; Zhang, Y.; Su, Z.; Zhai, Z.; Liu, X.; Mei, Y.; Xu, J.; Tian, H.; Wang, C.; Chen, C.; Yao, Y.; Liu, Z.; and Sun, M. 2025b. AgentCPM-GUI: Building Mobile-Use Agents with Reinforcement Fine-Tuning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 155–180.

Appendix

A Supplementary Experiments

Training Details

We fine-tune Qwen3-4B and AndesVL-4B through multi-task supervised fine-tuning using LoRA on a single NVIDIA A100 GPU, with a rank of 32, a scaling factor of 32, and a dropout rate of 0.1. We train the model for 5 epochs using AdamW with a learning rate of 5×10^{-5} , cosine decay, a warmup ratio of 0.05 and a batch size of 4. We jointly train the model on Record Analysis, Cross-Batch Reflection, and User Profiling. A task-aware sampler enforces a 2:1:1 allocation within each effective batch, corresponding to 2, 1, and 1 samples from the three tasks, respectively. All tasks use equal sample-level loss weights. We further adopt field-aware token weighting to emphasize task-critical outputs. Prompt tokens are excluded from loss computation; within the responses, tokens in the `rules` field of Record Analysis and the `result` field of Cross-Batch Reflection are assigned a weight of 10, while all remaining response tokens retain a weight of 1. The weighted token losses are normalized within each sample before being averaged across the batch. This design emphasizes the structured outputs directly used for rule generation and consolidation without uniformly amplifying the loss of an entire task.

Baseline Details

To fully compare PEAM with different methods, we design an expert-rule method consisting of 11 manually designed rules for common network-switching scenarios. Each rule specifies when a switch should be triggered, while the action for each user is determined by the configuration most frequently observed in that user’s historical records. The detailed expert rules are shown in Table A1.

Intent	Condition
Optimize cellular network usage for smooth meeting scenarios.	<code>app_scene=Meeting, WiFi_QoE < 4, cellular_score ≥ 80</code>
Optimize cellular network usage for smooth video call scenarios.	<code>app_scene=VIDEO_CALL, WiFi_QoE < 4, cellular_score ≥ 80</code>
Optimize cellular network usage for smooth voice call scenarios.	<code>app_scene=VOICE_CALL, WiFi_QoE < 4, cellular_score ≥ 80</code>
Optimize cellular network usage for smooth game-playing scenarios.	<code>app_scene=GAME_PLAYING, WiFi_QoE < 4, cellular_score ≥ 80</code>
Optimize cellular network usage for smooth game lobby scenarios.	<code>app_scene=GAME_LOBBY, WiFi_QoE < 4, cellular_score ≥ 80</code>
Save cellular data in live-streaming scenarios.	<code>app_scene=LIVE_STREAM, WiFi_QoE ≥ 4</code>
Save cellular data in long-video scenarios.	<code>app_scene=LONG_VIDEO, WiFi_QoE ≥ 4</code>
Save cellular data in short-video scenarios.	<code>app_scene=SHORT_VIDEO, WiFi_QoE ≥ 4</code>
Save cellular data in download scenarios.	<code>app_scene=DOWNLOAD</code>
Switch to cellular network when Wi-Fi quality is poor.	<code>WiFi_QoE < 3, cellular_score ≥ 80</code>
Switch to Wi-Fi when cellular network quality is poor.	<code>cellular_score ≤ 60, WiFi_QoE ≥ 4</code>

Table A1: Rule Conditions for Network Switching Intents

For the general-purpose models, we design task-specific prompts and retain each model’s default temperature and input/output token limits to reflect its out-of-the-box performance without model-specific tuning. During evaluation, we disable thinking mode for all models to ensure practical and comparable response latency.

On-Device Deployment

To validate on-device deployment, we ran a PEAM variant based on quantized AndesVL on an OPPO Find X9 Pro (MediaTek Dimensity 9500 processor, 16 GB of RAM, Android 16). The model adopts mixed-precision quantization: the backbone linear layers use W4A16, with 4-bit asymmetric weights and 16-bit symmetric activations; the KV cache uses INT8; RMSNorm, the final normalization layer, and logits use INT16; and the embedding and softmax layers retain floating-point precision. The left side of Figure A1 shows the logs produced when PEAM analyzes a user’s network-switching records and generates personalized strategies. The right side illustrates how PEAM automatically applies a previously generated strategy when the same context recurs, such as when the user launches a particular application. The overall performance results demonstrate the feasibility of on-device inference and real-time rule execution on consumer-grade hardware.

We further evaluated rule-execution efficiency on a device running PEAM. During the experiment, participants first performed several prescribed network switches, allowing PEAM to generate corresponding rules. After rule generation, they triggered the same scenarios again so that PEAM could retrieve and execute the matched rules. As shown in Table A2, the average rule-retrieval latency during real-time execution was 0.573 ms. Applying the selected

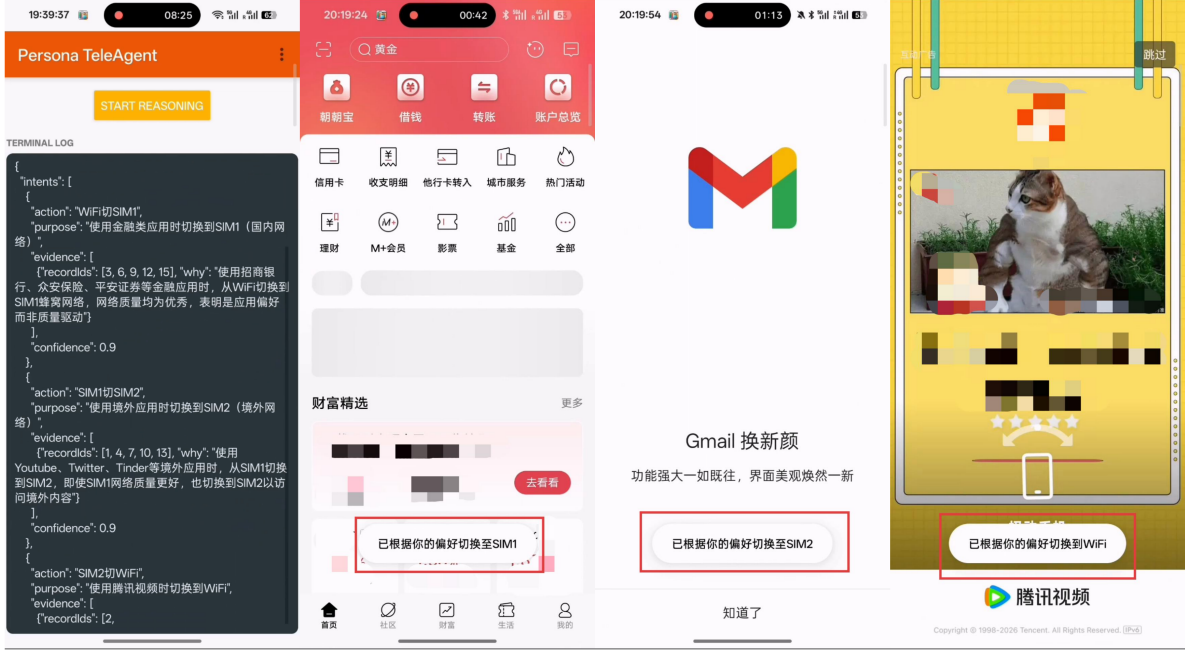


Figure A1: On-device deployment of PEAM.

rule required an average of 614.2 ms for switching between cellular networks and 93.57 ms for switching between cellular and Wi-Fi networks. Overall, PEAM completed rule retrieval and application within one second in all tested scenarios, satisfying users' responsiveness requirements.

Metric	Retrieval Time	Execution Time	
		Cellular-to-Cellular	Cellular-to-Wi-Fi
Time (ms)	0.573	614.2	93.57

Table A2: Runtime of strategy retrieval and execution. All times are reported in milliseconds.

Real Examples

We continuously monitored several users and collected at least one month of data from their mobile devices. In subsequent interviews, we asked the users to explain their motivations for performing network switches. We then compared the interview findings with the results produced by PEAM. The detailed comparison is presented in Tables A3–A8.

Generation Quality Evaluation

For Judge Model Score (JMS), DeepSeek-V3.2 compares each generated output with its reference answer and assigns an overall score from 0 to 100. For Record Analysis and Cross-Batch Reflection, the evaluation considers decision-structure completeness, intent-identification accuracy, and agreement of the generated network decisions with the reference. For User Profiling, it considers semantic consistency and information completeness, factual accuracy of network preferences and scenario descriptions, and naturalness and fluency of expression.

Intent	Objective Evaluation	Subjective Evaluation
<i>Interview Results</i>		
When using Douyin at two regular home locations, prefer connecting to the home Wi-Fi network; switch to cellular data when Wi-Fi is unavailable.	—	—
When using Nike Run Club at home, prefer connecting to Wi-Fi.	—	—
At home, prefer connecting to the home Wi-Fi network regardless of the app being used.	—	—
<i>PEAM Analysis Results</i>		
Switch to the cellular network when using Douyin or watching short videos at specific locations such as POI-1 and POI-2 to ensure a smooth viewing experience.	Fully Compliant (55)	Fully Compliant
Prefer connecting to a specific Wi-Fi network, SSID-1, at specific locations such as POI-1 and POI-3 to save cellular data or obtain a better network experience.	Fully Compliant (64)	Fully Compliant

Table A3: Comparison Between User Interview Results and PEAM Analysis Results for User 001

Intent	Objective Evaluation	Subjective Evaluation
<i>Interview Results</i>		
Use the China Mobile app for faster login; switch to the corresponding carrier SIM when opening the app, and switch back to Wi-Fi after login.	—	—
Connect to Wi-Fi when downloading apps from the app store to save cellular data.	—	—
Switch to the SIM 1 cellular network before using Alipay to ensure network stability.	—	—
Connect to the commonly used Wi-Fi network when entering a regular location.	—	—
Switch back to a more stable cellular network when Wi-Fi quality is poor.	—	—
Prefer using the cellular network when using the TeamTalk app at home.	—	—
<i>PEAM Analysis Results</i>		
Prefer connecting to SSID-1 at home, POI-1, on holiday afternoons.	Fully Compliant (7)	Partially Compliant
Prefer connecting to the company Wi-Fi network, SSID-2, at the office location POI-2 on weekday afternoons.	Fully Compliant (13)	Partially Compliant
Switch to the cellular network when the Wi-Fi user experience score is poor or relatively low to obtain a stable network connection.	Fully Compliant (31)	Fully Compliant
Connect to SSID-1 at POI-3 on holiday mornings.	Non-Compliant (0)	Partially Compliant
Connect to SSID-1 when using Sina Weibo on holiday afternoons.	Partially Compliant (1)	Not Mentioned

Table A4: Comparison Between User Interview Results and PEAM Analysis Results for User 002

Intent	Objective Evaluation	Subjective Evaluation
<i>Interview Results</i>		
When using Xiaohongshu in the first regular area at home, prefer manually connecting to the home Wi-Fi network.	—	—
When using Bilibili in the first regular area at the office, prefer manually connecting to an available Wi-Fi network; switch to cellular when the network quality is poor.	—	—
<i>PEAM Analysis Results</i>		
Connect to the cellular network when entering the company location.	Partially Compliant (2)	Not Mentioned
Habitually switch to Wi-Fi when using Xiaohongshu at the company location POI-1.	Partially Compliant (1)	Not Mentioned
Habitually switch to SIM 1 when using system tools such as Performance Monitor at POI-2.	Partially Compliant (1)	Not Mentioned

Table A5: Comparison Between User Interview Results and PEAM Analysis Results for User 003

Intent	Objective Evaluation	Subjective Evaluation
<i>Interview Results</i>		
Prefer using SIM 1 for Douyin short videos, possibly because SIM 1 has sufficient cellular data.	—	—
Switch to SIM 2 for Douyin and TeamTalk when the SIM 1 network quality is poor.	—	—
Switch to the cellular network when Wi-Fi quality is poor while using apps such as Youdao Dictionary and WeChat.	—	—
Proactively switch to the cellular network before leaving a regular location, likely to avoid lag caused by weak signals near the edge of Wi-Fi coverage.	—	—
Connect to another phone's hotspot for internet access during phone data backup or data migration.	—	—
Prefer using Wi-Fi when updating or downloading apps to save cellular data.	—	—
<i>PEAM Analysis Results</i>		
Habitually switch to the cellular network when using Douyin at night.	Mostly Non-Compliant (0)	Partially Compliant
Prefer using the cellular network when using WeChat on weekday mornings.	Fully Compliant (11)	Not Mentioned
Prefer connecting to SSID-1 at home, POI-1.	Fully Compliant (9)	Not Mentioned
Prefer using the cellular network when using TeamTalk on weekday afternoons.	Fully Compliant (10)	Partially Compliant

Table A6: Comparison Between User Interview Results and PEAM Analysis Results for User 004

Intent	Objective Evaluation	Subjective Evaluation
<i>Interview Results</i>		
When cellular data becomes stuck, prefer switching to a Wi-Fi network with better signal quality to continue using apps.	—	—
Manually connect to the in-building Wi-Fi network at POI-1.	—	—
Prefer using the company Wi-Fi network when using TeamTalk at the office.	—	—
Use cellular data to access Tencent News when Wi-Fi is unavailable.	—	—
When playing DNF Mobile, switch to cellular data if the connected Wi-Fi network has poor quality.	—	—
<i>PEAM Analysis Results</i>		
Switch to the SIM 1 cellular network when the Wi-Fi user experience score is poor or relatively low to obtain a stable network connection.	Fully Compliant (4)	Partially Compliant

Table A7: Comparison Between User Interview Results and PEAM Analysis Results for User 005

Intent	Objective Evaluation	Subjective Evaluation
<i>Interview Results</i>		
When using Douyin, switch to cellular data if the connected Wi-Fi network becomes stuck.	—	—
<i>PEAM Analysis Results</i>		
Switch to the SIM 2 cellular network when the Wi-Fi user experience score is poor or relatively low to obtain a stable network connection.	Fully Compliant (4)	Partially Compliant
Switch to the cellular network when watching Douyin live streams on holiday mornings to ensure smooth viewing and avoid lag.	Partially Compliant (2)	Partially Compliant
Connect to SSID-1 at POI-1.	Fully Compliant (9)	Not Mentioned

Table A8: Comparison Between User Interview Results and PEAM Analysis Results for User 006

B Dataset

Real-World Record Acquisition

To characterize the communication contexts in which users perform network switches, we recruited more than 1,00 participants and, with their consent, installed a data-collection application on their smartphones. To comprehensively describe each communication context, we collected multiple types of data covering spatiotemporal information, foreground applications, network access states, cellular network quality, and Wi-Fi network quality, as detailed in Table A9. It is worth noting that all collected records are conducted with the participants’ explicit consent and authorization. Records are strictly encrypted both at rest and in transit to safeguard user privacy and prevent unauthorized disclosure.

Category	Features
Spatiotemporal Context	Timestamp, POI, geofence
Foreground Application	App package name, app scene, app foreground dwell time, app-scene foreground dwell time
Network Access Status	Network access type, data SIM slot, PLMN, roaming status
Cellular Network	Cellular network type, RSRP, RSRQ, SINR, cellular signal level, network quality score
Wi-Fi Network	SSID, BSSID, frequency band, bandwidth, RSSI, QoE score, Wi-Fi signal level
Others	Traffic configuration, VPN configuration, battery status

Table A9: Contextual Features Used for Network Intent Modeling

The data-collection application operates similarly to the Record Acquisition module in PEAM. When a user initiates a network switch, the application records the buffered data from the preceding $\delta(T)$ seconds at one-second granularity and continues collecting data for $\delta(T)$ seconds after the switch. The two temporal windows are combined into a snapshot of a single switching event and stored locally on the device. After operating for a predefined collection period, the application automatically encrypts and uploads the accumulated snapshots, which constitute the user’s network-switching records for that period.

Data Preprocessing

Because the communication context is recorded at one-second granularity, even a single switching record can result in a lengthy textual representation. Meanwhile, each per-second snapshot provides only an instantaneous view, making it difficult for an LLM to directly identify temporal changes. We therefore statistically summarize the raw data before presenting it to the model. We categorize the collected fields into three types: stable fields, continuously varying fields (e.g., RSRP and RSRQ), and discretely varying fields (e.g., SSID). Stable fields mainly include temporal and location-related information, which typically exhibits no significant change around a network switch. We therefore use the value observed at the switching moment as the representative value for the entire snapshot. For continuously varying fields, we compute the mean separately over the pre-switch and post-switch windows. For discretely varying fields, we use the mode within each window. Because the foreground application provides the most direct indication of user–device interaction, we preserve the complete foreground-application trace throughout the observation window rather than aggregating it. Figure A2 illustrates an example of the resulting standardized data representation.

The statistically summarized data still cannot be directly provided to the model, as compact LLMs often struggle to interpret raw numerical communication metrics. We therefore apply a semantic transformation to make the underlying context and its changes more explicit. For each record, the transformation program first reads the pre-switch and post-switch values and then processes each field individually. For the temporal field, it extracts the switching time and determines whether the date is a working day and which period of the day the event belongs to. Location is represented by combining geofence and point-of-interest (POI) information. For foreground applications, the program maps package names to human-readable application names and lists the application changes over time line by line. Numerical network-quality metrics are converted into qualitative descriptions, allowing the model to interpret changes in network conditions more intuitively. Following this procedure, each field transition is expressed in the form $A \rightarrow B$, producing a semantic representation of the complete switching process. An example of the standardized semantic representation is shown below.

Simulation

We construct simulated-user records from templates derived from real-world network-switching samples, together with user-specific contexts and annotated intent rules. The simulated users vary in their spatiotemporal contexts, applications, device configurations, and network conditions. Each record represents a before–after network-state transition associated with a condition–action rule, retaining the general structure of the collected records. The resulting data provide explicit intent and rule annotations for optimizing clustering on the simulated training users and evaluating personalized rule generation on the disjoint simulated test users.

```

"before": {
  "data_times": "2026-06-21 00:36:45.743",
  "poi": "POI-X",
  "geo_fence_id": "001",
  "latitude": x x x ,
  "longitude": x x x ,
  "app_details": {
    "pkg_name1": {
      "name": null, "duration": 6, "time_range": "1-6"
    },
    "pkg_name2": {
      "name": "app_A", "duration": 3, "time_range": "7-9",
      "app_scene1": { "name": "COMMON", "duration": 3 }
    },
    "pkg_name3": {
      "name": "app_B", "duration": 4, "time_range": "10-
13",
      "app_scene1": { "name": "COMMON", "duration": 4 }
    },
    },
    "way_to_access_internet": "CELLULAR",
    "fps": 45.0,
    "delay": 120.0,
    "sim1Plmn": x x x x x ,
    "sim2Plmn": x x x x x ,
    "sim1Roaming": 0.0,
    "sim2Roaming": 0.0,
    "sim1_network_type": "LTE",
    "sim2_network_type": "NR_SA",
    "data_slot_id": 1,
    "trad_score": 62.0,
    "sim1_rsrp": -104.2,
    "sim2_rsrp": -108.0,
    "sim1_rsrq": -12.0,
    "sim2_rsrq": -17.0,
    "sim1_snr": 6.8,
    "sim2_snr": -2.0,
    "wifi_ssid": null,
    "wifi_bssid": null,
    "wifi_frequency": null,
    "wifiBandwidth": null,
    "wifiQoeLevel": null,
    "wifi_rssi": null,
    "wifiSignalLevel": null,
    "battery_level": 23.0,
    "batteryStatus": 3.0
  },

```

```

"after": {
  "data_times": "2026-06-21 00:37:00.603",
  "poi": "POI-X",
  "geo_fence_id": "001",
  "latitude": x x x ,
  "longitude": x x x ,
  "app_details": {
    "pkg_name1": {
      "name": "app_C",
      "duration": 10,
      "time_range": "1-10",
      "app_scene1": {
        "name": "COMMON",
        "duration": 10
      }
    },
    },
    "way_to_access_internet": "WIFI",
    "fps": 58.0,
    "delay": 35.0,
    "sim1Plmn": x x x x x ,
    "sim2Plmn": x x x x x ,
    "sim1Roaming": 0.0,
    "sim2Roaming": 0.0,
    "sim1_network_type": "LTE",
    "sim2_network_type": "NR_SA",
    "data_slot_id": 1,
    "trad_score": 100.0,
    "sim1_rsrp": -101.9,
    "sim2_rsrp": -96.0,
    "sim1_rsrq": -11.0,
    "sim2_rsrq": -12.0,
    "sim1_snr": 9.2,
    "sim2_snr": 5.0,
    "wifi_ssid": "XXXXXXXXXX",
    "wifi_bssid": "xx:xx:xx:xx:xx:xx",
    "wifi_frequency": 2412.0,
    "wifiBandwidth": 0.0,
    "wifiQoeLevel": 5.0,
    "wifi_rssi": -60.2,
    "wifiSignalLevel": 4.0,
    "battery_level": 23.0,
    "batteryStatus": 3.0
  }

```

Figure A2: An example of the resulting standardized data representation

```
## Log 1
Time: 2026-06-21 00:36:45.743 (Rest, Early Morning)
Location Label: 001 (POI_A)
Network Decision: SIM2→WiFi (xx:xx:xx:xx:xx:xx)
** Foreground Applications **
00s-06s: None
07s-09s: App_A (System) (General)
10s-13s: App_B (System) (General)
Network Switch Occurs
16s-25s: App_C (General)
Frame Rate: 45.0→58.0
Latency: 120.0→35.0
** Cellular Network **
Network Type: NR_SA
Algorithm Score: Fair→Good
RSRP: Poor→Fair
RSRQ: Fair→Good
SINR: Poor→Fair
** Data Usage **
SIM1 Data Usage: Unknown
SIM2 Data Usage: Unknown
** WiFi Network **
Frequency: None→2412.0
Bandwidth: None→0.0
Signal Level: None→FairUser
Experience Score: None→Good
RSSI: None→Good
** VPN **
VPN Status: None
VPN Name: None
** Battery **
Battery Level: 23.0%
```

Figure A3: An example of the standardized semantic representation.

C Clustering Algorithm

We employ a Tree-structured Parzen Estimator (TPE)-based hyperparameter optimization strategy on the simulated training users to jointly optimize the feature weights and HDBSCAN parameters. The clustering objective combines the Adjusted Rand Index (ARI) and Normalized Mutual Information (NMI) with weights of 0.7 and 0.3, respectively. TPE iteratively searches for the parameter configuration that maximizes the average clustering score across users.

We then apply the optimized parameters to the simulated test users to evaluate clustering performance. The results show that the resulting clustering configuration can effectively separate records associated with different latent user intents, thereby improving input coherence and reducing noise in the data provided to the model.

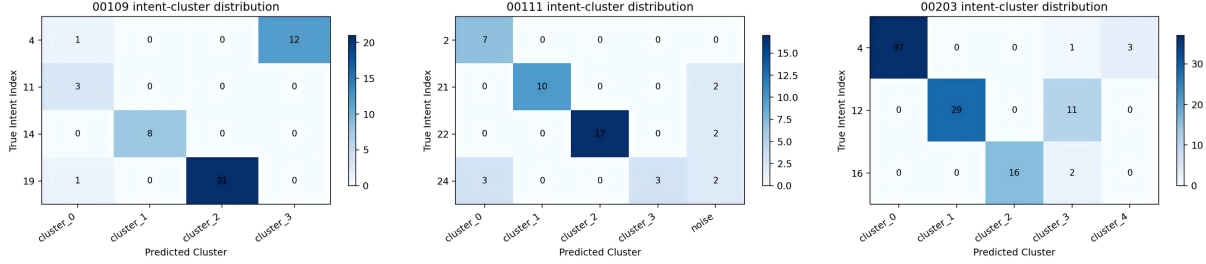


Figure A4: Clustering results of optimized parameters on simulated users.

D User-Level Local Memory Management

Each rule r is represented by a condition–action pair, an LLM-assigned confidence score q_r , and a state $s_r \in \{\text{silent}, \text{active}, \text{invalid}\}$. Although the LLM provides an initial confidence score q_r^0 during rule generation, every newly generated rule enters the rule base in the silent state. A silent rule can participate in condition matching but cannot recommend or execute its action. Instead, when its condition is satisfied, PEAM passively observes the network configuration selected by the user without presenting a recommendation. If the user’s choice agrees with the rule action, the observation is regarded as positive evidence; otherwise, it is treated as negative evidence. A new rule is activated only after two consecutive positive observations, which prevents rules supported by incidental or noisy records from being immediately applied.

PEAM updates rule confidence using fixed positive and negative steps. Let $y_r \in \{0, 1\}$ denote the feedback associated with a matched rule, where $y_r = 1$ indicates positive evidence and $y_r = 0$ indicates negative evidence. The confidence update is defined as

$$q_r \leftarrow \begin{cases} q_r + \Delta^+, & y_r = 1, \\ q_r - \Delta^-, & y_r = 0, \end{cases} \quad (\text{A1})$$

where Δ^+ and Δ^- denote the confidence increase and decrease steps, respectively. For a silent rule, y_r is determined by whether the user independently selects the same network configuration as the rule action. Any inconsistent observation resets the consecutive-match counter. For an active rule, PEAM presents the retrieved configuration to the user and asks whether it should be accepted. Acceptance is treated as positive feedback, whereas rejection or the absence of feedback is treated as negative feedback. The configuration is executed only after explicit acceptance.

Active rules are eligible for real-time recommendation and execution. When the confidence of an active rule falls below the activation threshold τ_a , the rule returns to the silent state and is subsequently evaluated through passive observations. Unlike newly generated rules, a previously active rule can be reactivated once its confidence recovers to τ_a , without requiring two additional consecutive matches. If the confidence falls below the invalidation threshold τ_i , the rule becomes invalid and is permanently excluded from subsequent matching, execution, and user-profile construction. Thus, active and silent states are mutually reversible, whereas the invalid state is irreversible.

When multiple active rules match the current context, PEAM first selects the rule whose condition contains the largest number of matched fields. If several rules have the same number of matched fields, the rule with the highest confidence is selected. Only the selected rule receives the current user feedback and updates its confidence. The user profile is periodically reconstructed using active rules only, ensuring that it is grounded in validated and currently reliable configuration preferences.