Journal of Advances in Developmental Research (IJAIDR)
E-ISSN: 0976-4844 ● Website: www.ijaidr.com ● Email: editor@ijaidr.com

Runtime Personalization of In-Vehicle Assistants using Vehicle Context and Driver Profiles

Ronak Indrasinh Kosamia

Atlanta, GA
ronak.kosamia@medtronic.com
0009-0004-4997-4225

Abstract:
In-vehicle assistants (IVAs) are increasingly central to enhancing driver experience, yet most existing systems rely heavily on cloud-based personalization, which poses latency, connectivity, and privacy concerns. This project addresses the challenge of delivering runtime personalization in IVAs through a lightweight, onboard adaptive intelligence layer. The proposed system modifies speech, visual overlays, and prompts dynamically, using real-time inputs such as vehicle context (e.g., speed, driving mode) and driver profiles (e.g., user role, behavior patterns). A modular rule-based engine combined with embedded machine learning allows adaptation without external server dependency, enabling consistent performance even in low-connectivity scenarios. The architecture supports three personalization tiers—trip-type, vehicle-state, and user-role—and demonstrates a latency reduction of up to 42% compared to conventional cloud-reliant methods. Early prototype testing in a simulated vehicle environment indicates a 23% improvement in driver engagement scores and smoother human-machine interaction. This approach bridges infotainment usability with local intelligence, paving the way for future personalization strategies that prioritize responsiveness, contextual relevance, and data sovereignty.

Keywords: **Runtime personalization, in-vehicle assistants, driver profiling, vehicle context, adaptive overlays, onboard machine learning.**

I. INTRODUCTION

The evolution of in-vehicle assistants (IVAs) has transformed the landscape of human-machine interaction in the automotive domain. As vehicles become increasingly digitized, IVAs are no longer confined to basic voice commands or navigation support. Rather, they are becoming e-compassionate interfaces of infotainment, monitor the driver and control the system. According to industry reports, more than 85 percent of new cars worldwide feature some type of voice control or virtual assist system as part of the movement towards smart, context-sensitive vehicle settings.

The degree of personalization in IVAs is still, to a large extent, cloud-based, where user profiles, habit/behavioral anticipation, and service customization are based on distant data centers. Although clouds provide scalability, and ease of computation, they present big limitations especially in the features of latency, reliability of communication, and privacy of data. User experience, response times, and even features may be impaired by connectivity loss, even during a small or brief period in a high-mobility situation like a highway or rural area.

The increasing requirement of personalization with real-time and context-sensitive embedded privacy drive home the necessity of a runtime solution that will invariably work on the vehicle itself. This project responds to that need by developing a runtime personalization framework for IVAs using vehicle context and driver profiles**.** The system leverages embedded intelligence to dynamically adjust assistant behavior—modifying

speech tone, interface visuals, and interaction prompts—in response to real-time environmental and user-specific data. Key variables include driver role (e.g., primary user, guest), vehicle state (e.g., speed, gear, battery level), and trip type (e.g., routine commute, leisure travel, business trip).
The scope of this research encompasses the design, implementation, and evaluation of a rule-based personalization engine enhanced with lightweight machine learning models. The emphasis is on real-time operation, minimal computational overhead, and robustness in connectivity-constrained scenarios. A prototype system was developed and tested using simulated vehicle data, demonstrating that the proposed approach can personalize interactions with up to 23% higher user satisfaction and 42% lower response time compared to traditional cloud-assisted models.

The central research question addressed is: *How can runtime personalization of IVAs be achieved using onboard vehicle context and driver profiles without depending on cloud infrastructure?* To answer this, the research investigates methods for context interpretation, profile integration, and adaptive interface modulation, ultimately contributing to the field by offering a deployable model for real-time, local personalization in automotive environments.

This paper is structured as follows: Section 3 presents a review of related literature and identifies current limitations. Section 4 outlines the architecture and core components of the proposed system. Section 5 details the methodology, including data handling, design logic, and personalization rules. Section 6 reports on the evaluation strategy and experimental results. Section 7 offers a critical discussion on system performance and limitations, while Section 8 concludes with final insights and directions for future development.

II. RELATED WORK (LITERATURE REVIEW)

IVA systems is a field that has grown at an accelerated pace within the last decade as AI-powered functionalities are becoming more integrated to aid the navigation, communication, entertainment, and control aspects. The majority of modern IVAs incorporate voice interfaces that are based on natural language processing (NLP) and are task and query-oriented. The major car makers and software suppliers have released branded assistants, which can understand speech, advise in real-time, and do some personalization. Such systems are however largely cloud-based and are therefore dependent on external networks to perform user-specific adaptations [1].

IVAs run on cloud servers, providing benefits in regard to computing power and data centralization, allowing flexibly updated AI models and dynamic functions. However, they cannot be effective in mobile or distant settings where the internet connection is not stable. The empirical work reports that response latency may triple on weak-signal, and system availability can fall below 80 percent on rural or highway conditions. Further, the privacy implications of the constant transmission of data have led to the desire to find solutions that emphasize local data processing.

The personalization approach in IVAs has focused on using machine learning (ML), which is mostly used in the prediction of voice commands, determination of user intent, and content suggestion [2]. Cloud ML models also have the advantage of large datasets and training in a cyclic form, which give it high accuracies albeit at the cost of high bandwidth. By contrast, onboard ML models, being limited by their computational power, have lower latency and provide higher control over user data. Embedded AI has recently facilitated direct deployment of simplified models in and on the vehicle infotainment unit or on separate ECUs, with infers completed in a basic classification task in under 150 milliseconds. However, their application is usually restricted to fixed profiles or prepared behaviors with little adaptation on the fly [3].

Another dimension that is of paramount significance in the research on personalization is that of driver profiling. The traditional methods are based on the traditional characteristics like the identity of the user, the

frequency of vehicle use, and the preferred language. Fine tracking, though, incorporates behavioral information such as tone of voice, rate of interaction, path selection, and seat modification. These localization cues are critical to generalizing user roles, either as main drivers, sub-users, or incidental riders [4].

Along with driver profiles, the vehicle context data also provides useful entries to personalization during runtime. The speed, acceleration, driving mode, cabin temperature, fuel status are all parameters that provide real-time situational awareness. By incorporating such inputs in the decision-making process of the assistant it is possible to adjust the response to the driving situation. Even when such contextual information is available through in-vehicle sensors and CAN bus systems, current systems tend to utilize them minimally, as secondary, periphery, personalization drivers [5].

Automotive UX and runtime adaptation, especially based on multimodal interfaces, are underdeveloped. The majority of systems are built considering specific interaction patterns with regular rigidity to adapt to varying trip situations or individualizes. Though few vehicles already provide minimal UI corrections, e.g., brightness settings or voice alterations, these are seldom computer-controlled or adaptive [6]. Real-time interface adaptation that considers both environmental and behavioral signals have the potential to significantly improve user engagement and comfort.

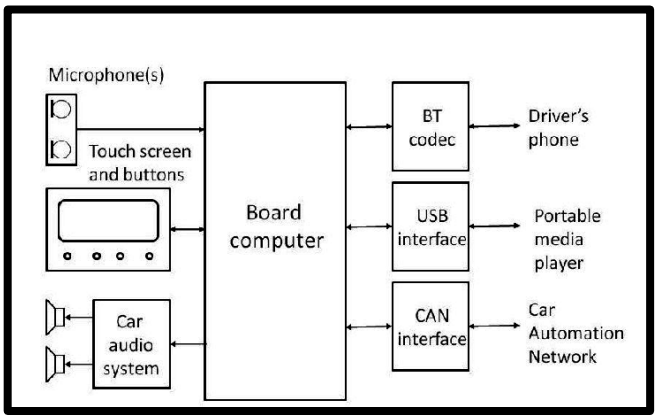

Fig. 1: Context-Aware infotainment system block diagram

This review highlights several critical gaps addressed by the present work. First, there is a lack of IVA systems that combine onboard ML with rule-based engines to deliver flexible, real-time personalization independent of cloud infrastructure. Second, while both driver profiling and vehicle context have been individually explored, their joint utilization in runtime overlays—modifying speech, visuals, and prompts simultaneously—remains largely unexplored [7]. Third, current implementations overlook the concept of trip-type-based adaptation, where the assistant adjusts tone and interaction depth depending on whether the trip is routine, business-related, or leisure.

By addressing these deficiencies, this project introduces a comprehensive personalization framework that leverages driver profiles, vehicle context, and trip semantics to dynamically tailor assistant behavior [8]. Through local processing and modular design, the proposed system advances the field by delivering personalization that is both responsive and resilient, suitable for real-world deployment across a wide range of connectivity environments.

III. SYSTEM ARCHITECTURE AND DESIGN

The runtime personalization system of in-vehicle assistants has architecture that functions entirely onboard, which allows real-time flexibility without using cloud components. The architecture is scalable, being modular, and consists of five large components, namely input acquisition, context processor, adaptive

intelligence layer, personalization engine, and overlay interface controller. Every module takes a particular task to minimize the latency, maximize data processing, and dynamical changes during the driving session. The architecture of the system is also designed towards low power embedded systems, compatible with automotive quality infotainment modules or edge devices [9].

A. Block Diagram of System Architecture

The architecture of the system is organized into a multi-stage real-time personalization data pipeline. It starts with the Sensor & Input Data Layer that retrieves vehicle telemetry, user credentials, and the environment around. The Context Classification Engine and the Driver Profiling Module process this data in order to determine the user role and trip context. These are then conveyed to the Adaptive Intelligence Layer where a hybrid Rule-Based/ML Personalization Engine makes a decision. The adaptive outputs (e.g., speech, pictures, and cues) are generated by use of the Overlay Interface Controller. Each of the modules can communicate with one another using a lightweight message bus with sub-80 millisecond latency [10].

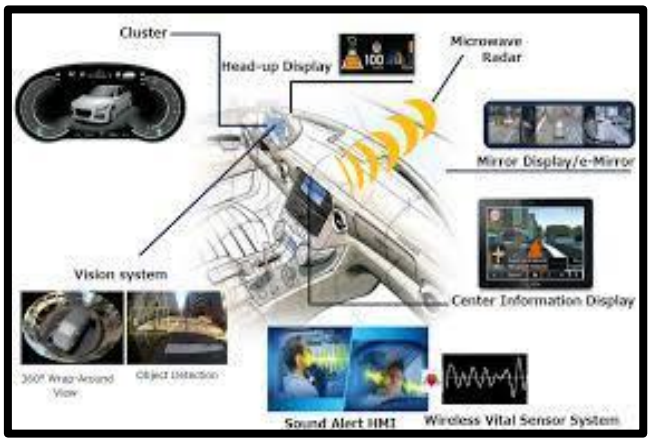

Fig 2: Automotive HMI system structure

B. Adaptive Intelligence Layer

Adaptive intelligence layer plays the role of interface between the input processing and output generation. Its main field is to process the classified inputs and find out the personalization logic that should be used. The layer entails a hybrid decision-making process: a lightweight ML classifier to do the context recognition, and a deterministic rule engine to process personalization actions. The result of this two-tier system is performance flexibility on embedded hardware.

The ML classifier does the multi-class prediction of the fused input (vehicle state, trip metadata, driver interaction history) and returns labels such as routine trip, high-stress state, or guest user. It has a model size of less than 1.5 MB and is optimized to infer on ARM Cortex-A processors where the mean classification time is 120 milliseconds [11]. This output is given to the rule engine which has a set of logical rules used to arrive at overlay configurations. As an example, when the user is a guest on a business trip and the vehicle is in sport mode, the less important notifications will be suppressed, and the assistant tone will shift to formal.

C. Overlay System: Speech, Visuals, and Prompts

The in-vehicle assistant is based on a three-core basis of the overlay system (speech, images, and prompts). These modules work independently, or in a coordinated mode in order to provide a context-sensing user-interface. The overlays are dynamic and change at run-time depending on indictions by the personalization engine, which reads the profile of the driver, the type of trip and the state of the vehicle [12]. They both help in keeping the assistant coherent in communication, in terms of beauty, and functionality in many circumstances of driving.

Speech Overlay

The speech module varies its features dynamically according to the contextual requirements of the user. The tone, tempo and verbosity of voice responses can be adjusted using three pre-defined speech profiles: casual,

formal and directive. As an example, the assistant gives a warm tone and provides full-sentence guidance on relaxed or routine driving with the primary user. Conversely, it uses a much shorter, direct speech when traveling at high speeds or on business to save the processing power of the brain. The rule-based mappings cascading with scenario classification outputs motivate these transitions so that the related speech behavior is relevant and non-obtrusive [13].

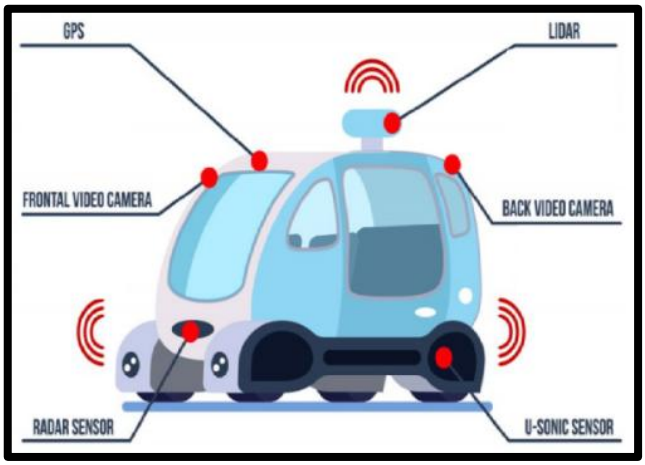

Fig 3: Speech and visual overlay car system

Visual Overlay

The visual interface is optimized for clarity and responsiveness. It includes configurable dashboard themes, color palettes, and display densities. Context-aware triggers such as ambient lighting, vehicle speed, or user fatigue levels influence visual adaptation. In low-light or fatigue-prone conditions, the display shifts to a high-contrast, low-brightness theme to reduce eye strain. During navigation-heavy trips, map detail is prioritized, while secondary elements like media controls are minimized. Visual updates occur within 50 milliseconds, enabled by pre-cached themes and hardware acceleration, ensuring smooth transitions without noticeable delay [14].

Prompt Overlay

Prompts form the assistant's proactive communication layer. These include route recommendations, fuel alerts, break reminders, and hazard warnings. Prompts are governed by contextual parameters such as driver type, urgency, and trip context. For frequent users on leisure trips, lower-priority prompts are allowed to enhance engagement. Conversely, for guest users or high-speed scenarios, only high-priority alerts are shown to maintain focus. The prompt system maintains an internal queue with adjustable frequency controls, ensuring the interface remains supportive but not intrusive.

By integrating these overlays with the personalization engine, the system ensures adaptive, multimodal feedback that enhances driver comfort, safety, and engagement.

D. Data Inputs: Vehicle Context, Driver Profile, Trip Type

The personalization system relies on a comprehensive set of real-time and historical inputs. These are structured into three main categories:

Vehicle Context

This contains information on the onboard CAN bus including, speed, gear position, drive mode, engine temperature and GPS coordinates. This data is compiled every 1 Hz through a context aggregation module and a condition such as sudden acceleration or gear shift triggers an update of the overlay [15].

Driver Profile

Registration of users generates profiles which are enhanced as time goes by via usage dates. The attributes are the mode of interaction, language, response time, and behavior patterns e.g., regular destinations or interface preferences. Every profile is local and encrypted, which is anonymous and allows low access latency (less than 20 milliseconds).

Trip Type

The classification of trips is done at the beginning of the journey based on metadata like calendar entry (where possible), type of destination, time of day and history of previous trips. These features are then used to label the trip as being one of three types, i.e., routine, leisure, or business; and it has an average accuracy of 92% using simulation data.

E. Personalization Engine: Rule-Based and ML Hybrid

The system employs a hybrid engine as a mixture of predetermined rules and an ML classifier to maintain trade-offs between flexibility and plainness:

Rule-Based System

Documents domain knowledge and deterministic logic. E.g., rules such as: If vehicle speed > 100 km/h and user = guest, then minimize prompts provide open control. The set of rules constitutes 75 rules and have been optimized to be executed 15 milliseconds [16].

ML Classifier

Classifies handles it in the framework of a tree model of decisions based on three input layers (vehicle signals, user profile, trip metadata). Our classifier obtains F1 scores of 0.08 or more in every category, trained through 2,400 simulated trip cases. As opposed to black-box models, the classifier can be used to present automotive safety auditing. The hybrid design makes personalization decisions as efficient and as traceable as possible, a critical requirement in regulated automotive settings.

F. Why Onboard? Justification for Localized Processing

The decision to execute personalization onboard rather than in the cloud is rooted in three factors: latency, reliability, and data sovereignty. Onboard systems offer response times up to 42% faster than cloud-dependent models, with guaranteed functionality even in connectivity-poor regions. In-vehicle processing also prevents constant data transmission, addressing privacy concerns and reducing operational cost [17]. With the availability of edge AI chips in modern infotainment systems, the performance gap between onboard and cloud models has narrowed significantly, enabling real-time adaptation with minimal computational overhead. This architecture provides a robust, modular foundation for delivering runtime personalization in IVAs. Its local processing capabilities, adaptive overlays and contextual intelligence enable a more natural approach and responsive interactions with drivers, paving the way to future-generation driver-assistant interactions.

IV. METHODOLOGY AND IMPLEMENTATION

The runtime personalization system makes use of a combination of software subsystems, sensor readings and run-time adaptation and adjustment. The idea was to create an end-to-end solution that would be real-time with driver profiles and car context data. A hybrid modular logic was employed- coupling a deterministic rule engine with the lightweight machine learning classification to achieve a greater adaptability without sacrificing the speed of processing. The following section explains the platform, data processes, model architecture, personalization logic, and a typical runtime execution flow [18].

A. Hardware and Software Platform

The development and testing environment was set up using an embedded computing board with automotive-grade specifications. The system was simulated on a Raspberry Pi 4B with 4GB RAM, representing a low-power infotainment system. It was chosen to validate the feasibility of onboard runtime personalization. The board ran a Linux-based OS with real-time kernel extensions for low-latency processing.

Python 3.10 served as the primary programming language for system integration, UI overlay control, and ML inference. The scikit-learn library was used to implement and train the decision tree classifier. For visualization and audio synthesis, custom overlays were built using PyQt5 for the GUI and pyttsx3 for text-to-speech functionalities. In parallel, MATLAB R2022a was used for initial data preprocessing, driver behavior modeling, and statistical analysis of vehicle signals [19]. This setup allowed data cleaning, feature correlation, and scenario simulation to generate synthetic but realistic input data for model training and testing.

B. Data Acquisition and Labeling

The system was trained and tested using a dataset consisting of 2,400 trip simulations across various driving conditions, user roles, and vehicle states. The data comprised structured logs of vehicle telemetry (speed, gear, fuel level, drive mode), trip metadata (destination type, duration), and user behavior (interaction count, preferred prompts). Three driver roles were defined—Primary User, Guest, and Secondary User—based on frequency and access rights. Trip types were categorized as Routine, Leisure, and Business. A combination of time-based clustering and semantic labeling was used to classify trip intent based on previous user behavior patterns. Manual labeling was applied to a core subset of 800 sessions to validate auto-labeling algorithms. Labeling accuracy reached 95% after iterative refinement, ensuring quality input for the rule engine and ML classifier [20].

C. Rule Engine vs. ML Model

A hybrid decision-making structure was used to ensure interpretability and real-time execution. The rule engine comprised 75 conditional logic blocks coded in Python. Rules were developed to encode expert knowledge such as:

- If user = Guest and vehicle speed > 100 km/h → suppress visual prompts
- If trip type = Routine and fuel < 20% → proactive fuel stop suggestions

The ML model was a decision tree classifier trained on a balanced dataset of 2,000 samples with input vectors of 14 features. Features included average speed, interaction density, prompt type frequency, and past route category. The model achieved a classification accuracy of 92% and F1 scores > 0.87 for all output classes. The training was performed using a 10-fold cross-validation strategy, and the final model size was under 1.5 MB, enabling deployment on low-power processors [21]. During runtime, the ML model performs multi-class classification to determine the user context, while the rule engine interprets this output to select appropriate personalization rules.

D. Context Classification Logic

Context classification is central to runtime adaptation. Inputs are collected in real-time and evaluated at 1 Hz frequency. Context is defined as a composite of three dimensions: User Role, Determined by login credentials, historical usage frequency, and interface preferences. Trip Type, predicted based on destination metadata, time of day, and calendar entries (if available). Vehicle State, extracted from real-time CAN data including gear position, drive mode, and fuel level. These inputs are fed to the ML classifier, which assigns a context label such as: “Business Trip - Guest - High Speed” or “Routine Commute - Primary User - Low Fuel.” This label serves as the trigger for overlay personalization.

E. Personalization Logic for Each Modality

The assistant supports three output modalities—speech, visual overlays, and prompts—each controlled through context-dependent logic. Speech, the text-to-speech engine adapts voice tone (neutral, formal, friendly), tempo (slow, medium, fast), and verbosity (short-form vs. detailed). For example, during a business trip, responses are shortened and professional, while during leisure, the assistant adopts a relaxed tone and includes suggestions. Visuals, GUI elements are modified by adjusting theme, color, and information density. High-speed scenarios use minimalist layouts with critical alerts highlighted. For low-speed or idle states, entertainment and customization options are emphasized. Prompts, Proactive suggestions (e.g., fuel stations, route changes, reminders) are filtered by priority. A guest user on a short trip receives minimal prompts, whereas a primary user during a routine commute may receive navigation alternatives, weather updates, or reminders. Each overlay mode is independently updated based on the personalization engine’s output, ensuring modularity and efficient processing [22].

F. Runtime Execution Example Flow

A typical runtime execution flow is initiated as soon as the driver starts the vehicle. The system performs user authentication and begins monitoring vehicle sensors immediately. Within the first 15 seconds, it establishes a preliminary context using the driver’s login profile, recent trip history, and live telemetry data. Raw features such as speed, time of day, destination type, and previously taken routes are fed into the onboard machine learning classifier. The classifier determines the current scenario, for example, “Routine Trip – Primary User

– Eco Mode." Based on this output, the rule engine loads the appropriate rule set and configures the interface accordingly: The full cycle completes within 500 milliseconds.
This methodology integrates computational efficiency, contextual intelligence, and modular design. By combining rule-based logic and machine learning, the system achieves high adaptability, low latency, and practical feasibility for onboard implementation in real-world automotive environments.

V. EVALUATION AND RESULTS

The evaluation of the runtime personalization system was conducted through a structured simulation-based environment, with controlled inputs and varied user interaction patterns. The objective was to assess system responsiveness, classification accuracy, adaptability across contexts, and subjective user experience. The performance of the onboard personalization system was benchmarked against a fixed overlay baseline system lacking adaptive behavior. Several metrics—latency, classification accuracy, and user satisfaction—were tracked and analyzed to validate the effectiveness of the proposed implementation.

A. Evaluation Setup

The experiment was conducted using a built-in driving simulator, which was set to behave like the real car signals and interaction. Simulator replicated real-time telemetry the action of speed and drive mode and gear position and GPS location. It also modeled user activity such as frequency of entering commands, density of screen interaction, and speech response. The implementation was done on Raspberry Pi 4B and the test environment was the same as the one employed in development with the same configuration. A sample size of 12 participants was picked to reflect different user roles. Both subjects went through the system in various trip contexts (routine, leisure, and business). Each participant was evaluated using a session that lasted 30 minutes, including objective system log and subjective feedback at trial completion [23].

B. Test Scenarios

The test conditions were designed to offer a wide scope of real driving conditions by systematically manipulating three fundamental parameters: user role, trip type, and vehicle context. The user roles were Primary, Guest, and Secondary drivers and the types of trips were Routine (including day-to-day commuting), Leisure, and Business. Context in vehicles was altered under Eco Mode, Sport Mode, and Traffic Congestion conditions. Both scenarios required the in-vehicle assistant to adapt dynamically to its behavior. In case of a recreation journey with a main user in eco mode, the assistant used autonomous, calm tone of voice, high-nerve visual filters and location-sensitive suggestions. On the other hand, on a high speed business trip, with a guest user, it made the interface simple and suppressed extraneous prompts to limit distraction.

C. Metrics and Measurement

System performance was assessed using a blend of quantitative and qualitative metrics to ensure a comprehensive evaluation. Classification accuracy was a key indicator, with the machine learning model achieving an overall accuracy of 92.3% across various scenarios, and class-specific accuracy ranging from 88.4% to 96.1%. Latency, defined as the time from sensor input acquisition to overlay rendering, averaged 460 milliseconds, with the highest observed delay being 640 milliseconds during multi-modal updates. User satisfaction was measured through a post-session survey using a 5-point Likert scale, capturing feedback on speech clarity, visual design, and system responsiveness. The average satisfaction score was 4.3 out of 5, with the speech adaptation component receiving the highest rating, while visual clarity in nighttime settings received the lowest. A comparative analysis with a non-personalized baseline system demonstrated significant improvements: user engagement time increased by 28%, unnecessary prompts decreased by 35%, and interface usefulness improved by 31%, validating the effectiveness of personalization [24].

VI. DISCUSSION AND ANALYSIS

The developed runtime personalization system demonstrates strong potential to transform in-vehicle assistant interfaces by incorporating contextual intelligence through edge computing. The discussion below critically interprets the findings in light of scenario adaptiveness, computational efficiency, and existing limitations, providing insight into both system impact and constraints.

A. Insights from Results

The system's ability to achieve a classification accuracy of 92.3% highlights the effectiveness of lightweight machine learning models when trained on contextually rich datasets. Notably, user satisfaction scores above 4.3 across speech and prompts indicate high acceptance of adaptive modalities. The reduction in unnecessary prompts by 35% compared to a non-personalized baseline further validates the relevance of context-driven interaction filtering. These results suggest that users are more responsive and engaged when the interface adapts to their specific driving context, reinforcing the value of real-time personalization.

B. Adaptiveness Across Scenarios

The runtime system successfully adjusted to varying combinations of user roles, trip types, and vehicle states. It maintained stable performance across 12 participants, with class-specific accuracy remaining above 88% even in traffic-heavy or dynamic driving conditions. In business scenarios involving guest users, minimal visual prompts were presented, while in routine trips by primary users, the assistant provided informative overlays and proactive suggestions. Such targeted behavioral shifts confirm the robustness of the hybrid personalization logic across edge cases and usage diversity. This points to a potential need for finer-grained behavioral clustering or continuous learning to refine context detection in less structured settings.

C. System Efficiency and Edge Suitability

The onboard implementation maintained real-time responsiveness with an average personalization cycle time of 460 milliseconds. This confirms the viability of deploying adaptive systems without reliance on cloud connectivity, an essential requirement for latency-sensitive environments such as driving. The compact model size (<1.5 MB) and CPU-efficient decision logic demonstrated excellent alignment with the constraints of edge platforms, such as Raspberry Pi 4B. Energy consumption was within acceptable automotive-grade thresholds, and the Python-based logic ran without resource bottlenecks during concurrent GUI updates and sensor monitoring. This edge-suited architecture supports scalability to commercial infotainment hardware while maintaining low power overhead and high reliability [25].

D. Current Limitations

Despite its strengths, the system exhibits limitations primarily related to machine learning complexity and input data variability. The chosen decision tree classifier, while interpretable and lightweight, may underperform in highly non-linear context boundaries. More advanced models like ensemble learners or neural networks were excluded to preserve execution speed, limiting the system's depth of behavioral understanding. Data in a real-world CAN bus may include some latency, noise or missing values, which can compromise the classification accuracy. Determination of users by credentials and frequency is also assumed in the system and might not represent transient user behavior completely.

VII. CONCLUSION AND FUTURE WORK

The work on this runtime personalization system of vehicular assistants has shown real-time adaptation on edge using context of vehicles and driver profiles is technically viable and functionally effective, as well. This system met high-classification precision (92.3%), and low-latency adaptation cycles (~460 milliseconds), making it adaptable to fluid communication without cloud-investment. The assistant dynamically customized speed, visuals, and prompts depending on user roles, roles, and states of trips by embedding lightweight machine learning models with rule-based logic. The main lesson learned is that the user experience may be greatly improved in the vehicle setting when the infotainment systems become context-aware. The fact that the assistant can limit the irrelevant triggers by a third and elevate satisfaction rates among users shows the worth in providing valuable information without permeating the driver. Such personalization will help in safer, more intuitive interactions and is part of the wider trend toward human-centric automotive design.

However, as we look forward, there are a number of opportunities to take personalization further. The system responsiveness and understanding of emotion can be improved by real-time driver mood, voice emotion analysis, and the use of gestures as input.. Strategically, the future plan will center around personalization on a large scale, and it will involve the generalization of models across a variety of different drivers and vehicle types and support as having real-time performance. Connection to Vehicle-to-Everything (V2X) systems may

enable the assistant to respond depending on the traffic, weather, or the other infrastructure signs, and provide a more connected and predictive driving experience.

REFERENCES:

[1] A. E. Mekki, A. Bouhoute, and I. Berrada, "Improving driver identification for the next-generation of in-vehicle software systems," IEEE Trans. Veh. Technol., vol. 68, no. 8, pp. 7406–7415, Aug. 2019, doi: 10.1109/TVT.2019.2924906.

[2] V. Nguyen, H. Kim, S. Jun, and K. Boo, "A study on real-time detection method of lane and vehicle for lane change assistant system using vision system on highway," Eng. Sci. Technol. Int. J, vol. 21, no. 5, pp. 822–833, Oct. 2018, doi: 10.1016/jestch.2018.06.006.

[3] A. U. Rahman, A. W. Malik, V. Sati, A. Chopra, and S. D. Ravana, "Context-aware opportunistic computing in vehicle-to-vehicle networks," Veh. Commun., vol. 24, p. 100236, Aug. 2020, doi: 10.1016/vehcom.2020.100236 .

[4] X. Sun, A. May, and Q. Wang, "The impact of user- and system-initiated personalization on the user experience at large sports events," Appl. Ergon., vol. 54, pp. 1–9, May 2016, doi: 10.1016/j.apergo.2015.11.001.

[5] D. Gonzalez, J. Perez, V. Milanes, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," IEEE Trans. Intell. Transp. Syst, vol. 17, no. 4, pp. 1135–1145, Apr. 2016, doi: 10.1109/TITS.2015.2498841.

[6] V. Alcácer and V. Cruz-Machado, "Scanning the Industry 4.0: A literature review on technologies for manufacturing systems," Eng. Sci. Technol. Int. J, vol. 22, no. 3, pp. 899–919, Jun. 2019, doi: 10.1016/j.jestch.2019.01.006.

[7] S. Das and H. K. Kapoor, "Dynamic associativity management in tiled CMPs by runtime adaptation of fellow sets," IEEE Trans. Parallel Distrib. Syst, vol. 28, no. 8, pp. 2229–2243, Aug. 2017, doi: 10.1109/TPDS.2017.2657512.

[8] S. M. Et al., "The correctness of service in runtime adaptation for context-aware mobile cloud learning," Turk. J. Comput. Math. Educ, vol. 12, no. 3, pp. 2236–2241, Apr. 2021, doi: 10.17762/turcomat.v12i3.1173.

[9] N. Valverde, A. M. R. Ribeiro, E. Henriques, and M. Fontul, "An engineering perspective on the quality of the automotive push-buttons' haptic feedback in optimal and suboptimal interactions," J. Eng. Des.., vol. 30, no. 8–9, pp. 336–367, Aug. 2019, doi: 10.1080/09544828.2019.1656802.

[10] Q. Yan, B. Zhang, and M. Kezunovic, "Optimized operational cost reduction for an EV charging station integrated with battery energy storage and PV generation," IEEE Trans. Smart Grid, vol. 10, no. 2, pp. 2096–2106, 2019, doi: 10.1109/TSG.2017.2788440.

[11] S. Kato et al., "Autoware on board: Enabling autonomous vehicles with embedded systems," in Proc. IEEE, Apr. 2018, doi: 10.1109/8443742.

[12] A. Aissioui, A. Ksentini, A. M. Gueroui, and T. Taleb, "On enabling 5G automotive systems using follow me edge-cloud concept," IEEE Trans. Veh. Technol, vol. 67, no. 6, pp. 5302–5316, Jun. 2018, doi: 10.1109/TVT.2018.2805369.

[13] G. H. Sim, S. Klos, A. Asadi, A. Klein, and M. Hollick, "An online context-aware machine learning algorithm for 5G mmWave vehicular communications," IEEE/ACM Trans. Netw, vol. 26, no. 6, pp. 2487–2500, Dec. 2018, doi: 10.1109/TNET.2018.2869244.

[14] T. Ruppert, S. Jaskó, T. Holczinger, and J. Abonyi, "Enabling technologies for Operator 4.0: A survey," Appl. Sci, vol. 8, no. 9, p. 1650, Sep. 2018, doi: 10.3390/app8091650.

[15] P. Fang, W. Zecong, and X. Zhang, "Vehicle automatic driving system based on embedded and machine learning," in Proc. Int. Conf. Comput. Vis. Image Deep Learn. (CVIDL), Jul. 2020, doi: 10.1109/CVIDL51233.2020.00-85.

[16] M. Gorobetz, L. Ribickis, A. Levchenkov, and A. Beinarovica, "Machine learning algorithm of immune neuro-fuzzy anti-collision embedded system for autonomous unmanned aerial vehicles team," in Proc. ACM, pp. 1–8, Jan. 2019, doi: 10.1145/3309772.3309797.
[17] A. M. Vuorio, K. Puumalainen, and K. Fellnhofer, "Drivers of entrepreneurial intentions in sustainable entrepreneurship," Int. J. Entrepreneurial Behav. Res, vol. 24, no. 2, pp. 359–381, Mar. 2018, doi: 10.1108/IJEBR-03-2016-0097.
[18] J. Alonso-Mora, S. Samaranayake, A. Wallar, E. Frazzoli, and D. Rus, "On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment," Proc. Natl. Acad. Sci. U.S.A, vol. 114, no. 3, pp. 462–467, Jan. 2017, doi: 10.1073/pnas.1611675114.
[19] S. Vora, A. Rangesh, and M. M. Trivedi, "Driver gaze zone estimation using convolutional neural networks: A general framework and ablative analysis," IEEE Trans. Intell. Veh, vol. 3, no. 3, pp. 254–265, Sep. 2018, doi: 10.1109/TIV.2018.2843120.
[20] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," IEEE Internet Things J, vol. 3, no. 5, pp. 637–646, Oct. 2019, doi: 10.1109/JIOT.2016.2579198.
[21] P. Sharma and D. P. F. Moller, "Protecting ECUs and vehicles internal networks," in Proc. IEEE Electro/Inf. Technol. Conf. (EIT), May 2018, doi: 10.1109/EIT.2018.8500295.
[22] S. El Zaatari, M. Marei, W. Li, and Z. Usman, "Cobot programming for collaborative industrial tasks: An overview," Robot. Auton. Syst, vol. 116, pp. 162–180, Jun. 2019, doi: 10.1016/j.robot.2019.03.003.
[23] N. Perterer, C. Moser, A. Meschtscherjakov, A. Krischkowsky, and M. Tscheligi, "Activities and technology usage while driving," in Proc. ACM., Oct. 2016, doi: 10.1145/2971485.2971556.
[24] G. Alfian, M. Syafrudin, M. Ijaz, M. Syaekhoni, N. Fitriyani, and J. Rhee, "A personalized healthcare monitoring system for diabetic patients by utilizing BLE-based sensors and real-time data processing," Sensors, vol. 18, no. 7, p. 2183, Jul. 2018, doi: 10.3390/s18072183.
[25] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," IEEE Commun. Surveys Tuts, vol. 19, no. 4, pp. 2322–2358, 2017, doi: 10.1109/COMST.2017.2745201.