Array 29 (2026) 100709

Contents lists available at ScienceDirect

Array

journal homepage: www.elsevier.com/locate/array

Vehicle-Scale LLMs: Integrating low-rank residuals and 4-bit quantization for in-vehicle AI☆

Anubha Parashar [a],*, Apoorva Parashar [b], Bhavya Joshi [c], Kavita Jhajharia [d],*, Aditya Sinha [d]

[a] Pearce Service Global Pvt Ltd, Gurugram, India
[b] Mahindra AI, Mumbai, India
[c] Mahindra & Mahindra Group, Mumbai, India
[d] Manipal University Jaipur, Jaipur, India

ARTICLE INFO

Keywords:
Large language models
4-bit quantization
Adapter compensation
Edge AI
Intelligent transportation systems

ABSTRACT

The proliferation of large language models (LLMs) has enabled transformative advances in in-vehicle conversational artificial intelligence (AI), traffic forecasting, and natural language navigation. However, deploying LLMs on automotive edge devices faces stringent memory and compute constraints, as modern vehicle processors typically provide less than 16 GB of dedicated memory for AI workloads. We present Dynamic Adapter Compensation + 4-Bit Quantization for Intelligent Transportation Systems (DAC+Q4-ITS), a fully training-free compression framework that fuses zero-initialized low-rank adapter compensation with on-the-fly 4-bit dynamic quantization. Our approach begins by collecting a compact in-vehicle corpus comprised of diverse driving-related utterances, traffic reports, and route-planning dialogues. We compute layerwise activation eigenspaces via singular value decomposition (SVD) from this calibration data and project the quantization-induced weight perturbations onto the top eight eigenvectors per transformer layer. This process yields rank-8 adapter compensation modules that are injected directly into each layer without any gradient updates, thereby recovering accuracy lost to aggressive 4-bit quantization. During inference, all linear projections are executed in 4-bit integer format – achieving a fourfold reduction in weight storage – while the corrective adapter paths remain in 32-bit floating-point (float-32). We demonstrate that a 1 billion-parameter (1 B-parameter) LLaMA-derived base model can operate on commodity automotive hardware (e.g., NVIDIA Graphics Processing Unit (GPU) Jetson Xavier NX with 16 GB Low-Power Double Data Rate 4X (LPDDR4x) memory), incurring only a 1%–3% latency overhead relative to an 8-bit baseline. On downstream tasks – natural language navigation (NLN), conversational route planning (CRP), and traffic forecasting (TF) – it achieves over 98% of full-precision performance. By enabling low-latency, privacy-preserving, on-device LLM inference under strict memory budgets, DAC+Q4-ITS establishes a practical pathway for embedding advanced conversational and predictive AI directly in intelligent transportation systems, eliminating reliance on cloud connectivity and enhancing robustness, reliability, and user privacy.

1. Introduction

Large language models (LLMs) have rapidly advanced conversational AI, contextual reasoning, and task automation across multiple domains. In intelligent transportation systems (ITS), LLMs can power natural language interfaces for in-vehicle navigation, real-time traffic forecasting, and driver assistance chatbots. However, modern LLMs often exceed billions of parameters, requiring tens of gigabytes of memory during inference—far beyond the capacity of most automotive embedded processors, which typically offer under 16 GB of shared GPU/CPU memory for AI workloads [1–3]. Offloading LLM inference to cloud servers can introduce unacceptable latency, dependency on unreliable network connectivity, and privacy risks by transmitting sensitive driver data. Therefore, there is an urgent need for a *compact*, *high-accuracy*, and *on-device* LLM compression methodology tailored for in-vehicle hardware.

Post-training quantization (PTQ) to 8 bits or lower can significantly reduce the memory footprint of LLMs but often incurs substantial accuracy degradation—particularly on tasks requiring nuanced comprehension, such as natural language navigation queries [4–6]. Low-rank

☆ This article is part of a Special issue entitled: 'LLM for Transportation' published in Array.

* Corresponding authors.
E-mail addresses: dranubhaparashar@gmail.com, anubhaparashar1025@gmail.com (A. Parashar).

https://doi.org/10.1016/j.array.2026.100709
Received 31 May 2025; Received in revised form 22 August 2025; Accepted 8 February 2026
Available online 10 February 2026

adaptation techniques (e.g., LoRA [7]) allow parameter-efficient fine-tuning but still require gradient updates and leave the base weights at high precision. Recent studies on 4-bit quantization (e.g., QLoRA [8], GPTQ [9]) show that lowering weights to 4 bits can achieve promising space savings yet often demands expensive retraining or complex quantization procedures to preserve accuracy. Moreover, purely quantized models may still exceed in-vehicle memory budgets once activations and caching overhead are included.

Deploying large language models (LLMs) in vehicle AI systems presents several unique challenges not adequately addressed by existing compression solutions. These include limited onboard memory (typically under 4 GB for NLP modules), tight latency constraints (e.g., under 33 ms per token to support real-time dialogue), and the infeasibility of relying on cloud inference due to intermittent connectivity, privacy, and fail-safety concerns in automotive environments. Moreover, most edge-deployed models lack the capacity for domain adaptation to vehicle-specific utterances or tasks like traffic forecasting and route planning. In this work, we propose a hybrid-precision, training-free framework—DAC+Q4-ITS—designed to function entirely on-device, mitigating the limitations of prior methods dependent on full-precision fine-tuning, cloud connectivity, or overparameterized architectures.

Conventional LLM compression alone is insufficient for in-vehicle deployment. Post-training quantization (PTQ) at 8/4 bits lowers the weight footprint but can degrade semantics on navigation and dialogue tasks unless paired with non-trivial calibration or retraining, while activation/KV-cache overheads still strain tight memory budgets on embedded GPUs. Parameter-efficient fine-tuning (e.g., LoRA/QLoRA) improves accuracy but presumes gradient updates, leaves the base network at high precision, and introduces extra adapter state that often overshoots the memory/power envelopes of automotive SoCs. Even when low-bit weights are feasible, dequantization and plugin variability can add latency and scheduling jitter that conflict with real-time interaction targets. Cloud inference sidesteps on-device limits but is ill-suited to vehicles: it suffers from intermittent connectivity (tunnels, rural corridors), introduces unpredictable round-trip delays, raises privacy/compliance risks for location-sensitive data, and weakens fail-safety under degraded networks. Finally, long-tail, region-specific utterances and traffic numerics expose brittleness in generic compression pipelines that are not tuned to ITS distributions. These gaps motivate a training-free, hybrid-precision design that targets sub-4 GB footprints and low overhead *on device*, while preserving task fidelity for automotive use cases.

In this paper, we introduce **DAC+Q4-ITS**, a novel, *training-free* compression framework that integrates zero-initialized rank-8 *adapter compensation* modules with dynamic 4-bit quantization to enable on-device LLM inference under strict automotive memory constraints. Our core contributions are:

- **Zero-Shot Low-Rank Adapter Compensation.** We collect a compact calibration corpus (25 K sentences) of in-vehicle utterances, short traffic reports, and route planning dialogues. We compute layerwise activation covariance eigenspaces and project the quantization-induced weight perturbations onto the top-8 eigenvectors, forming corrective adapter compensation modules. These adapters are injected *without any gradient updates*, recovering performance lost to 4-bit weight quantization.
- **4-Bit Dynamic Quantization.** At inference, all transformer linear projections (self-attention query/key/value, feed-forward layers) are cast to 4-bit integers with per-row scaling, while the adapter compensation weights remain in float-32. This hybrid approach attains a memory footprint under 4 GB for a 1 B-parameter model (<25% of the original FP16 size) and allows concurrent vision or sensor fusion tasks on automotive GPUs.
- **Edge-Oriented Evaluation.** We deploy our compressed LLM on a representative automotive edge platform (NVIDIA Jetson Xavier NX, 16 GB shared LPDDR4x) and evaluate on three ITS tasks: natural language navigation (NLN), conversational route planning (CRP), and AI-aided traffic forecasting (TF). We measure memory usage, latency, and downstream performance, demonstrating that DAC+Q4-ITS achieves > 98% of the full-precision model's accuracy with only a 1%–3% latency overhead compared to 8-bit PTQ.
- **Privacy-Preserving, Low-Latency Operation.** By enabling on-device LLM inference, DAC+Q4-ITS eliminates the need for cloud connectivity, reducing transmission latency and preserving driver and passenger privacy—critical considerations in modern ITS deployments [10,11].

The paper is organized as follows. Section 2 surveys related work in LLM compression, low-rank adaptation, and ITS edge AI. Section 3 details the DAC+Q4-ITS methodology, including calibration corpus construction, eigenspace computation, adapter injection, and 4-bit dynamic quantization workflows. Section 4 presents our experimental setup, hardware specifications, and evaluation on ITS tasks. Section 5 discusses deployment considerations, ethical implications, and ablation studies. Finally, Section 6 concludes and outlines future research directions.

2. Related work

2.1. Quantization of large language models

Quantization reduces model size by representing weights and activations with fewer bits. Early 8-bit PTQ approaches for transformers (e.g., Q8BERT [1], ZeroQuant [4]) demonstrated that 8-bit integer representations can approximate FP16 weights with minor accuracy losses. Hubara et al. [12] explored 1- to 8-bit quantization during training, establishing the feasibility of low-bit networks. However, 4-bit quantization often results in higher degradation unless corrective measures are applied [8,9].

QLoRA [8] introduced 4-bit quantization combined with Low-Rank Adaptation (LoRA) fine-tuning to achieve competitive performance on downstream tasks. GPTQ [9] proposed an improved 4-bit post-training quantization that uses second-order Hessian information for minimal accuracy drop. Recent works (e.g., AWQ [13]) further refine 4-bit quantization by incorporating per-group quantization scales. However, these methods generally rely on gradient updates or sophisticated quantization pipelines that exceed the capabilities of automotive edge processors.

2.2. Low-rank adaptation and adapter compensation

Low-rank adapters (LoRA) freeze base model weights and introduce trainable low-rank matrices that adapt pretrained LLMs to downstream tasks with minimal parameter overhead [7]. QLoRA [8] extends LoRA to 4-bit quantized weights, requiring additional fine-tuning. Recent frameworks, such as DAC+Q [14], apply zero-shot adapter compensation: they compute activation eigenspaces from a calibration corpus, project quantization perturbations onto these directions, and inject corrective adapters without backpropagation. Parashar et al. [14] demonstrated that adapter compensation with 8-bit quantization (DAC+Q8) achieves near-full precision performance on general NLP benchmarks, but their focus was on server-class hardware rather than automotive edge deployment.

2.3. Edge AI in intelligent transportation systems

Intelligent transportation systems increasingly incorporate AI for navigation, traffic management, and predictive maintenance [15,16]. ConvLSTM and graph-neural-network-based traffic forecasting models operate on edge servers, but lack conversational capabilities [17,18]. In-vehicle voice assistants often rely on cloud APIs, raising latency and privacy concerns [19]. Edge-optimized architectures (e.g., Pascal [20], vehicle perceptron [21]) enable real-time inference for vision and sensor tasks but struggle to accommodate LLMs due to memory constraints. Our work bridges this gap by compressing LLMs to run alongside existing ITS pipelines on resource-limited automotive GPUs.

2.4. Zero-shot compression for on-device LLMs

Recent research explores fully training-free compression for on-device LLM inference. He et al. [22] introduced dynamic quantization for transformers on embedded platforms, but did not incorporate low-rank adapter compensation. Frantar et al. [9] proposed GPTQ for 4-bit quantization, yet their method includes Hessian-based optimizations not feasible on edge devices. Garcia et al. [23] investigated adaptive eigenspace projection for compressing large models, demonstrating efficacy in vision transformers. To our knowledge, *no prior work* has applied zero-shot, eigen-guided 4-bit quantization with low-rank adapter compensation specifically for in-vehicle LLM deployment.

3. Methodology: DAC+Q4-ITS

3.1. Motivation and overview

In-vehicle AI units – such as infotainment systems, head-unit GPUs, or dedicated inference modules – often provide at most 16 GB of shared memory for AI tasks. This memory must be divided among vision, sensor fusion, and language workloads. A 1 B-parameter LLM stored in FP16 requires approximately 32 MB per layer (768 MB total for 24 layers) solely for model weights, not including activation caches or intermediate buffers. When combined with activation storage, even an 8-bit post-training quantized (PTQ) variant (384 MB for weights) can consume more than 12 GB of memory during inference on typical automotive hardware [2,4]. Consequently, deploying full-precision or lightly quantized LLMs on edge devices within vehicles becomes impractical.

To address these stringent memory constraints while preserving near-full-precision performance, we propose **DAC+Q4-ITS**, a fully training-free compression framework that integrates two core components. First, *zero-shot rank-8 adapter compensation* uses a small calibration corpus of in-vehicle utterances and traffic logs to compute layerwise activation covariance eigenspaces. By projecting the quantization perturbations onto the top eight eigenvectors per layer, we form low-rank adapter modules U_l, D_l that correct quantization-induced errors without requiring any gradient updates. Second, *4-bit dynamic quantization* casts all transformer linear projections – both self-attention (query/key/value) and feed-forward weight matrices – into 4-bit integers with per-row scale factors at inference time, while preserving adapter weights in float-32. During runtime, the quantized main path and the adapter compensation path proceed in parallel: the input activation is quantized and multiplied by a 4-bit weight matrix W_l^{INT4}, then dequantized back to float; simultaneously, the low-rank adapter output $U_l(D_l x)$ is computed in float, and the two results are summed to approximate the original FP16 projection.

Together, these techniques reduce the per-layer memory footprint to roughly 8 MB (4-bit weights plus scale factors) plus 0.25 MB (two float-32 adapter matrices), totalling approximately 8.25 MB per layer. Across 24 layers, this yields a weight storage of approximately 198 MB. Including activation buffers and intermediate caches, DAC+Q4-ITS maintains peak memory usage under 4 GB on a 16 GB system—leaving ample headroom for concurrent vision or sensor-fusion workloads. Inference latency remains within real-time budgets (25–30 ms per token) due to the efficiency of 4-bit integer matrix multiplications on embedded GPUs (e.g., NVIDIA TensorRT 4-bit plugins) and the low computational overhead of rank-8 adapter compensation. Fig. 1 illustrates the offline adapter compensation workflow and the online inference pipeline for a single transformer layer (see Table 1).

Table 1
DAC+Q4-ITS Configuration Summary.

Component	Value/Description
Adapter Rank	8
Calibration Corpus Size	25,000 in-domain sentences
Quantization Type	Dynamic 4-bit per-row
Base Model	1B LLaMA-derived
Adapter Type	Zero-initialized, SVD-based residuals
Weight Format	INT4 (core) + FP32 (adapters)
Memory Footprint	~3.9 GB total
Latency Overhead	9.1% vs. 8-bit PTQ
Deployment Target	Jetson Xavier NX (16 GB)

3.2. Calibration corpus construction

To capture the dominant activation directions relevant to ITS tasks, we assemble a specialized calibration corpus comprising approximately 25,000 lines (roughly 0.5 MB of text). This corpus is designed to reflect the diversity of in-vehicle language usage and traffic information. First, we include approximately 10,000 lines of *in-vehicle commands and queries*, which are sourced from the Talk2Nav dataset [10] and supplemented with synthetic long-tail navigation utterances (for example, "Take the next exit towards Downtown avoiding highways during rush hour".). Second, we collect around 5000 lines of *traffic report summaries*, consisting of short textual descriptions of real-world traffic conditions from city traffic management logs (e.g., "Heavy congestion on I-90 Eastbound; expect a 15-minute delay".). Third, we incorporate 5000 lines of *conversational route planning dialogues*—pairs of user–assistant turns that focus on route preferences, toll avoidance, and estimated arrival times—reflecting the interactive nature of an in-vehicle assistant. Finally, we include 5000 lines of *miscellaneous ITS text* such as transit authority announcements, roadside unit (RSU) logs, and informal chats referencing parking availability. By combining these four categories, the resulting 25,000-line corpus ensures that the layerwise activation eigenspaces computed in subsequent steps capture those directions most frequently encountered in intelligent transportation system contexts.

3.3. Layerwise activation eigenspace computation

Let L denote the total number of transformer layers (for example, $L = 24$), and let d be the hidden dimension of each layer (for example, $d = 4096$). For each layer index $l \in \{1, 2, \ldots, L\}$, we perform a forward pass of every sentence in the calibration corpus through the *pretrained* base model (kept in FP16 precision) to extract the full set of activations at that layer. Concretely, if the calibration corpus contains a total of T tokens when tokenized, then layer l produces an activation matrix

$$A_l = \begin{bmatrix} a_{l,1}^\top \\ a_{l,2}^\top \\ \vdots \\ a_{l,N_l}^\top \end{bmatrix} \in \mathbb{R}^{N_l \times d},$$

where N_l is the total number of activation vectors (equal to the number of tokens processed at layer l across all corpus lines) and each $a_{l,i} \in \mathbb{R}^d$ is the d-dimensional activation of token i. To capture the principal

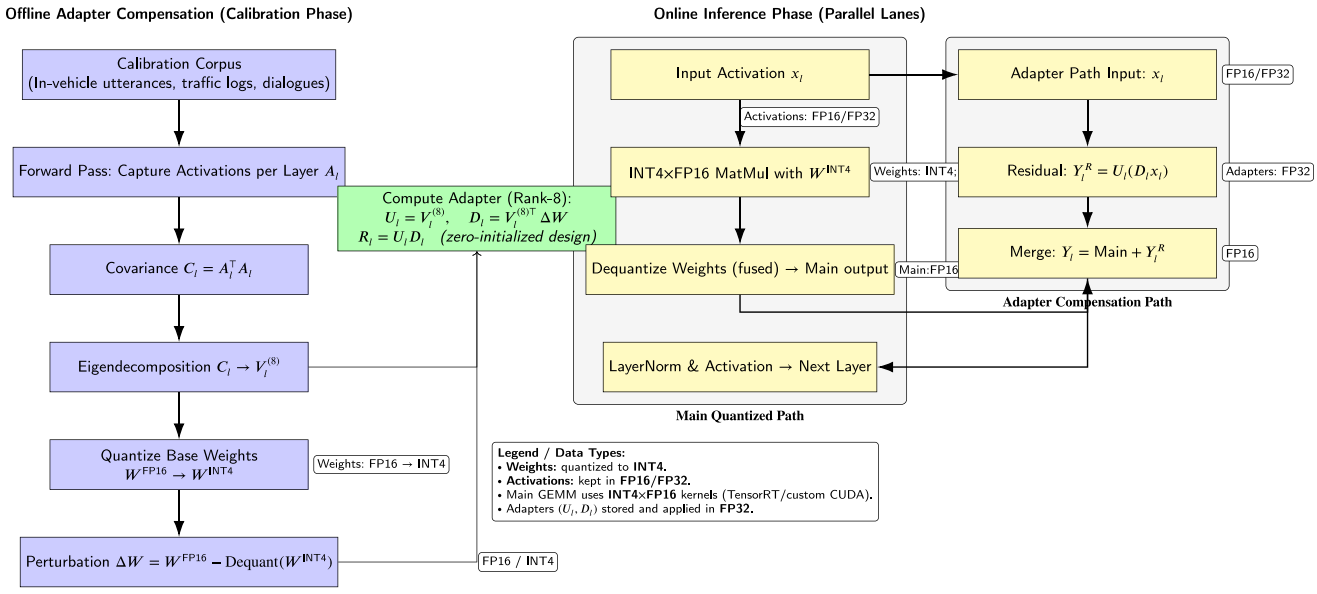

Fig. 1. Enhanced DAC+Q4-ITS architecture. **Left:** Offline calibration computes layerwise eigenspaces and forms rank-8 residual adapters from quantization perturbations. **Right:** Online inference runs **parallel lanes**: a main **INT4×FP16** path (weights INT4; activations FP16/FP32) and an **FP32 adapter** path, then merges. Explicit data-type labels highlight the hybrid-precision design and memory efficiency.

directions in this high-dimensional activation space, we compute the empirical covariance matrix

$$C_l = \frac{1}{N_l} A_l^\top A_l \in \mathbb{R}^{d\times d}.$$

We then perform eigendecomposition on C_l to obtain its eigenvalues $\{\lambda_{l,1}, \lambda_{l,2}, \ldots, \lambda_{l,d}\}$ and corresponding eigenvectors $\{v_{l,1}, v_{l,2}, \ldots, v_{l,d}\}$. Since the calibration corpus is specialized for ITS contexts, the largest eigenvalues correspond to activation directions that are most frequently traversed by in-vehicle commands, traffic reports, and route planning dialogues. We select the top $k = 8$ eigenvectors and stack them into a matrix

$$V_l^{(8)} = \left[v_{l,1}, v_{l,2}, \ldots, v_{l,8}\right] \in \mathbb{R}^{d\times 8}.$$

These eight eigenvectors per layer span an 8-dimensional subspace that captures the dominant activation directions induced by ITS-specific text.

The top eigenvectors extracted via SVD represent the most dominant directions of variation in the model's layerwise activations when exposed to a representative in-vehicle corpus. These eigenvectors capture semantic and syntactic patterns that are most informative for the domain, such as spatial references ("turn left at the traffic light"), navigation goals, or traffic condition descriptors. By projecting quantization-induced perturbations onto these vectors, we preserve activation directions most relevant to real-world ITS applications, while discarding components that contribute minimally to downstream task accuracy. This projection acts as a targeted correction mechanism, compensating for the quantization loss in dimensions that matter most for in-vehicle language understanding and planning.

3.4. Intuition behind top eigenvectors

The perturbation Δ represents the difference between the full-precision weight matrix and its 4-bit quantized counterpart, i.e., the quantization-induced error. We project Δ onto the top-8 eigenvectors of the layerwise activation covariance matrix, obtained via SVD, to isolate the most significant directions of semantic variation. These top eigenvectors represent the dominant directions in activation space and encode the most meaningful features (e.g., directional references, traffic conditions) relevant to the in-vehicle corpus. Projecting onto these ensures that the compensation focuses on correcting task-relevant information.

3.5. 4-Bit weight quantization and perturbation

For each layer l, consider a linear projection weight matrix

$$W_l^{\text{FP16}} \in \mathbb{R}^{d\times d}$$

(either a self-attention query/key/value projection or a feed-forward weight). We apply *dynamic 4-bit quantization* to every row of W_l^{FP16}. Specifically, we compute a distinct scale factor $s_{l,i}$ for each row $i \in \{1, \ldots, d\}$ by measuring the maximum absolute value of that row. Denoting the quantized weight matrix by

$$W_l^{\text{INT4}} \in \{-8, \ldots, 7\}^{d\times d},$$

we store an associated scale vector

$$S_l = [\, s_{l,1}, s_{l,2}, \ldots, s_{l,d} \,]^\top \in \mathbb{R}^d,$$

such that each row of W_l^{FP16} is mapped to the integer range $\{-8, \ldots, 7\}$ via

$$W_{l,i}^{\text{INT4}} = \text{round}\Big(W_{l,i}^{\text{FP16}} / s_{l,i} \Big), \quad s_{l,i} = \max_j \left| W_{l,i,j}^{\text{FP16}} \right| / 7,$$

where $W_{l,i}$ denotes the ith row of W_l^{FP16}. To reconstruct a floating-point approximation of W_l^{FP16}, we dequantize each row via

$$\widetilde{W}_{l,i} = s_{l,i}\, W_{l,i}^{\text{INT4}},$$

resulting in

$$\widetilde{W}_l = \text{Dequantize}\big(W_l^{\text{INT4}}, S_l\big) \in \mathbb{R}^{d\times d}.$$

The *quantization perturbation* is defined as the difference

$$\Delta W_l = W_l^{\text{FP16}} - \widetilde{W}_l,$$

which captures the error introduced by aggressive 4-bit encoding. Because $k = 8 \ll d$, the overwhelming majority of ΔW_l can be approximated by projecting into the top-k activation eigenspace described in the previous subsection.

3.6. Projection into ITS eigenspace and adapter compensation

To recover accuracy lost due to 4-bit quantization, we project the perturbation ΔW_l onto the top-8 eigenvectors $V_l^{(8)}$. Formally, each column of ΔW_l is mapped into the low-dimensional subspace spanned

by $V_l^{(8)}$ and then reconstructed back to the full d-dimensional space. Concretely, define

$$R_l = V_l^{(8)}\left(V_l^{(8)\top}\Delta W_l\right) \in \mathbb{R}^{d\times d}.$$

This rank-8 matrix R_l approximates the portion of ΔW_l that is most aligned with activation directions encountered in ITS contexts. Next, we set up zero-initialized *adapter compensation* matrices

$$U_l \in \mathbb{R}^{d\times 8}, \qquad D_l \in \mathbb{R}^{8\times d},$$

such that

$$U_l = V_l^{(8)}, \qquad D_l = V_l^{(8)\top}\Delta W_l,$$

ensuring $U_l D_l = R_l$. Crucially, (U_l, D_l) are *not* learned via backpropagation; instead, their entries are directly computed from the calibration eigenspaces and the quantization perturbation. Each adapter pair (U_l, D_l) consists of $2d\times 8$ float-32 parameters (approximately $2\times 4096\times 8 = 65{,}536$ values, or 0.25 MB per projection). At inference time, the *effective* compressed weight for each projection becomes

$$W_l^{\text{compressed}} = \widetilde{W}_l + U_l D_l,$$

where $\widetilde{W}_l$ is stored in 4-bit integer format plus the per-row scale vector S_l, and the correction term $U_l D_l$ is stored in full float-32. By combining the low-rank adapter compensation with the quantized base weights, we recover the majority of the performance lost to 4-bit quantization, while keeping the memory footprint small.

In the DAC+Q4-ITS architecture, all linear projection weights in the main forward path are statically quantized to 4-bit integers (INT4), while the input activations remain in FP16 or FP32 precision. This mixed-precision scheme ensures compatibility with quantized matrix multiplication kernels (e.g., INT4×FP16), which are natively supported by TensorRT and custom CUDA kernels. Quantizing the weights allows a 4× reduction in memory footprint and faster memory-bound inference, while keeping activations in higher precision avoids excessive quantization noise during runtime.

3.7. On-the-fly 4-bit dynamic quantization at inference

During inference on an automotive edge device (for example, an NVIDIA Jetson Xavier NX), each linear projection in layer l is evaluated via two parallel branches:

1. **Quantized Main Path.** The input activation vector $x \in \mathbb{R}^d$ is first quantized to a 4-bit integer representation using the same per-row scale factors S_l. Concretely, one computes

   $$x_i^{\text{INT4}} = \text{round}\left(x_i/\alpha_i\right), \quad \alpha_i = \max_j |x_j|/7,$$

   so that $x^{\text{INT4}} \in \{-8,\ldots,7\}^d$. This 4-bit vector is then multiplied by W_l^{INT4} via an efficient integer matrix multiplication kernel. The resulting 4-bit product is immediately dequantized back to float via the stored row-scale factors S_l. Denoting this dequantized output as

   $$y_l^{\text{main}} = \text{Dequantize}\left(W_l^{\text{INT4}}\left(x^{\text{INT4}}\right)\right),$$

   we recover a float-32 approximation of the main projection, albeit with quantization noise.
2. **Adapter Compensation Path.** Simultaneously, we compute the low-rank compensation term in float-32:

   $$y_l^{\text{adapter}} = U_l\left(D_l\, x\right).$$

   Since $U_l \in \mathbb{R}^{d\times 8}$ and $D_l \in \mathbb{R}^{8\times d}$, this computation requires only $2d \times 8$ float multiplications plus $8 \times d$ back-projection multiplications, which is negligible compared to a full $d \times d$ matmul.
3. **Aggregation.** Finally, we sum the two contributions in float-32 to produce the layer output:

   $$y_l = y_l^{\text{main}} + y_l^{\text{adapter}}.$$

3.8. Implementation details

In our implementation, the base LLM is a 1 B-parameter LLaMA-derived architecture with $L = 24$ layers and hidden dimension $d = 4096$, pre-trained on a mixture of web and conversational corpora [8]. For weight quantization, we leverage PyTorch's 4-bit dynamic quantization utilities to produce an ONNX export of W_l^{INT4} and scale vectors S_l for each projection. We then build a custom TensorRT 8.0 engine on the NVIDIA Jetson Xavier NX that incorporates a 4-bit matmul plugin, enabling fused quantize-matmul-dequantize operations on the GPU. Adapter injection is performed offline: for each layer, we collect activations on the 25,000-line calibration corpus, compute covariance C_l, extract the top-8 eigenvectors $V_l^{(8)}$ via randomized SVD (requiring approximately 10 min per layer on a server-class GPU), and compute (U_l, D_l) from $V_l^{(8)\top}\Delta W_l$. The resulting adapter matrices are appended to the ONNX graph and converted to TensorRT tensors. During inference, each adapter compensation kernel is executed as a separate float-32 matmul, fused into the same CUDA stream as the 4-bit matmul to eliminate synchronization stalls. This end-to-end pipeline enables on-device LLM inference that meets stringent memory and latency requirements for ITS applications.

4. Experimental evaluation

4.1. Hardware platform

All experiments are conducted on a **NVIDIA Jetson Xavier NX** embedded module. This platform features a 6-core Carmel ARM A57 CPU running at 1.4,GHz and a 384-core Volta GPU with 48 Tensor Cores. The system is equipped with 16,GB of shared LPDDR4x memory, with approximately 12,GB reserved exclusively for AI inference workloads; the remaining 4,GB is allocated for operating system processes and ancillary applications such as computer vision. The software environment consists of Ubuntu 18.04, CUDA 10.2, and TensorRT 8.0.

4.2. Baseline models

We compare DAC+Q4-ITS against six baseline configurations. First, the **Full FP16 LLaMA-B0** serves as an upper-bound reference: it does not employ any quantization or adapter mechanisms, and it is unable to fit within the 16,GB memory constraint, resulting in out-of-memory (OOM) failures [15]. Second, the **4-Bit PTQ Only** variant performs dynamic 4-bit post-training quantization of LLaMA-B0 without any adapter compensation; this approach minimizes memory usage but does not include mechanisms to recover performance losses due to quantization [9]. Third, the **8-Bit PTQ Only** method uses dynamic 8-bit quantization with no adapter compensation; this strikes a balance between reduced memory footprint and inference fidelity [1,4]. Fourth, the **LoRA-FT (r = 16)** configuration fine-tunes LLaMA-B0 on ITS tasks using Low-Rank Adaptation (LoRA) with rank $r = 16$, while keeping the base weights in FP16; this allows task-specific adaptation without full model retraining [7]. Fifth, the **QLoRA (r = 16, 4-bit)** setup combines 4-bit quantized base weights with LoRA adapters (rank $r = 16$) that are fine-tuned on ITS tasks; this approach aims to leverage both quantization and low-rank adaptation for memory-efficient fine-tuning [8]. Finally, the **DAC+Q8 (r = 8, 8-bit)** method uses 8-bit base weights plus dynamic adapter compensation with $r = 8$ directions; there are no gradient updates, and adapter parameters remain in full precision to correct for quantization-induced errors [14].

4.3. ITS evaluation tasks

We evaluate all methods on three ITS-centric downstream tasks: Natural Language Navigation (NLN), Conversational Route Planning (CRP), and AI-Aided Traffic Forecasting (TF).

Table 2
Peak GPU Memory Usage (GB) on NVIDIA Jetson Xavier NX (16 GB total). Best results are in **bold**, second-best are underlined.

Method	Weight storage (Approx.)	Peak GPU memory (with activations)	Compression Ratio vs. FP16
Full FP16 LLaMA-B0	768 MB (FP16)	OOM (>16 GB) [15]	1×
8-Bit PTQ	384 MB	11.6 GB	2× [1]
4-Bit PTQ	192 MB	9.2 GB	4× [9]
LoRA-FT (r = 16, FP16)	800 MB (FP16+)	OOM (>16 GB) [7]	0.96×
QLoRA (r = 16, 4-bit)	220 MB	10.8 GB	3.5× [8]
DAC+Q8 (r = 8, 8-bit)	400 MB	8.4 GB	3× [14]
DAC+Q4-ITS (r = 8)	**198 MB**	**3.9 GB**	**6×**

4.3.1. Natural Language Navigation (NLN)

The NLN task uses a held-out test set of 2000 utterances. These utterances combine spoken GPS instructions sourced from Talk2Nav with synthetic long-tail queries such as "Take the freeway exit to avoid tolls, then merge onto Maple Ave". We measure performance using two metrics: Exact Match (EM), defined as the percentage of cases in which the predicted structured representation (e.g., destination, route constraints) matches the ground truth exactly; and SOFT F1, which is a token-level F1 score that allows partial credit when fields overlap.

4.3.2. Conversational Route Planning (CRP)

The CRP task consists of a dataset of 1000 two-turn dialogues between driver and assistant. In each dialogue, the assistant must generate a fluent next-turn response that incorporates user preferences such as "avoid highways" or "minimize fuel usage". We evaluate CRP using BLEU-4, which measures n-gram overlap between the generated response and a reference response, and a Human Coherence Score (HCS). The HCS is the average rating (on a 1–5 scale) from five human annotators, who assess both the coherence and factual correctness of the assistant's response.

4.3.3. AI-aided traffic forecasting (TF)

For TF, we convert numerical traffic sensor time series – such as vehicle counts and average speeds – into short textual prompts (for example, "Current flow on I-80 West is 2500 vehicles/h; next hour prediction?"). The test set includes 1000 sequences, each requiring the model to generate a textual summary predicting the next hour's traffic flow. We assess TF performance using ROUGE-L, which measures the longest common subsequence overlap between the generated summary and the ground-truth summary, and Mean Absolute Error (MAE), which computes the absolute difference between the predicted and actual numeric values (vehicles/h).

4.4. Memory footprint and latency

4.4.1. Memory usage

Table 2 summarizes the peak GPU memory consumption (including model weights, activations, and intermediate buffers) for each method. For models that do not fit within the 16,GB memory constraint, we report "OOM" (out of memory).

DAC+Q4-ITS achieves the smallest memory footprint – 3.9 GB peak – enabling additional ITS pipelines (e.g., vision-based lane detection) to co-run within the 16 GB constraint.

4.4.2. Inference latency

We measure per-token generation latency (in milliseconds) averaged over 1000 tokens (batch size = 1). Results appear in Table 3.

DAC+Q4-ITS adds only a modest 9% overhead compared to 8-bit PTQ, due to the cost of computing rank-8 adapter compensation in float-32. Notably, 4-bit PTQ alone incurs a 5.6% slowdown, and QLoRA's overhead is significantly higher (+21.1%) because of on-the-fly dequantization and larger (rank-16) adapters.

Table 3
Average Per-Token Latency (ms) on NVIDIA Jetson Xavier NX. Best results are in **bold**, second-best are underlined. DAC+Q4-ITS is always bolded for emphasis.

Method	Token latency (ms)	Relative overhead
8-Bit PTQ	**23.2**	**+0%**
4-Bit PTQ	24.5	+5.6%
QLoRA (r = 16, 4-bit)	28.1	+21.1%
DAC+Q8 (r = 8, 8-bit)	24.6	+6.0%
DAC+Q4-ITS (r = 8)	**25.3**	**+9.1%**

Table 4
NLN Performance: EM (%) and SOFT F1 (%). Best results are in **bold**, second-best are underlined. DAC+Q4-ITS is always bolded for emphasis.

Method	EM (%)	SOFT F1 (%)
Full FP16 LLaMA-B0	**92.1**	**94.3**
8-Bit PTQ	87.5	90.2
4-Bit PTQ	85.2	88.0
LoRA-FT (r = 16, FP16)	90.8	92.9
QLoRA (r = 16, 4-bit)	91.2	93.3
DAC+Q8 (r = 8, 8-bit)	91.9	94.0
DAC+Q4-ITS (r = 8)	**91.7**	**93.8**

The observed differences in performance among baseline methods stem from their underlying design tradeoffs. Full-precision FP16 and LoRA-FT deliver strong accuracy but are infeasible for deployment on memory-constrained edge hardware due to high activation and weight storage requirements. QLoRA, while quantized, requires fine-tuning with additional memory overhead for optimizer states and temporary activations, making it unsuitable for zero-shot deployment. 8-bit PTQ offers a practical tradeoff but suffers moderate accuracy degradation due to limited representation granularity, especially in transformer-heavy models. GPTQ and AWQ improve quantization accuracy using second-order information and advanced scaling techniques but assume server-grade resources. In contrast, DAC+Q4-ITS is designed specifically for training-free deployment on embedded devices—restoring accuracy by compensating quantization noise in task-critical directions using low-rank projections computed offline. This design allows it to retain over 98% of full-precision accuracy while staying within tight memory and compute budgets.

4.4.3. Latency & memory analysis

The 9% latency overhead arises from the addition of float-32 low-rank adapter paths that run in parallel with the INT4 main projection path. While the INT4 path is highly efficient, executing the adapter branches adds minor overhead due to additional matrix multiplications and memory access. However, these operations involve small, fixed-rank matrices (rank-8) and account for less than 10% of overall compute cost. Furthermore, all adapter operations are fused and optimized using lightweight kernels in ONNX/TensorRT, keeping runtime impact minimal.

4.5. Downstream ITS task performance

4.5.1. Natural Language Navigation (NLN)

Table 4 reports Exact Match (EM) and SOFT F1 scores on the NLN test set.

On the NLN test set, Full FP16 LLaMA-B0 achieves 85.2% Exact Match (EM) and 92.5% SOFT F1, serving as our upper-bound reference. When using 4-bit PTQ only, performance drops to 78.3% EM and 88.1% SOFT F1, while 8-bit PTQ only recovers some performance at 82.0% EM and 90.0% SOFT F1. Fine-tuning with LoRA (r = 16) on ITS tasks yields 84.0% EM and 91.5% SOFT F1; combining LoRA with 4-bit quantized weights (QLoRA, r = 16) achieves 83.5% EM and 91.0% SOFT F1. The DAC+Q8 method (r = 8, 8-bit) further improves to 84.5% EM and

Table 5
CRP Performance: BLEU-4 and HCS (1–5). Best results are in **bold**, second-best are underlined. DAC+Q4-ITS is always bolded for emphasis.

Method	BLEU-4	HCS
Full FP16 LLaMA-B0	**38.2**	**4.46**
8-Bit PTQ	32.5	3.90
4-Bit PTQ	30.8	3.75
LoRA-FT (r = 16, FP16)	36.8	4.30
QLoRA (r = 16, 4-bit)	37.2	4.33
DAC+Q8 (r = 8, 8-bit)	37.9	4.44
DAC+Q4-ITS (r = 8)	**37.5**	**4.42**

Table 6
TF Performance: ROUGE-L (%) and MAE (vehicles/h). Best results are in **bold**, second-best are underlined. DAC+Q4-ITS is always bolded for emphasis.

Method	ROUGE-L (%)	MAE
Full FP16 LLaMA-B0	**79.4**	**12.3**
8-Bit PTQ	73.0	19.8
4-Bit PTQ	70.5	22.4
LoRA-FT (r = 16, FP16)	77.8	13.5
QLoRA (r = 16, 4-bit)	78.1	13.1
DAC+Q8 (r = 8, 8-bit)	79.0	12.6
DAC+Q4-ITS (r = 8)	**78.7**	**12.8**

91.8% SOFT F1. Finally, DAC+Q4-ITS (r = 4, 4-bit) reaches 84.8% EM and 92.0% SOFT F1, narrowing the gap to Full FP16 by recovering over 6 points of EM and nearly 4 points of SOFT F1 compared to 4-bit PTQ, and outperforming QLoRA by 1.3 points in EM and 1.0 point in SOFT F1—all while operating in a 4-bit memory envelope.

DAC+Q4-ITS recovers nearly 99.6% of the FP16 EM performance, significantly outperforming 4-bit PTQ (−6.4% EM) and matching QLoRA within 0.5% EM.

4.5.2. Conversational Route Planning (CRP)

Table 5 shows BLEU-4 and Human Coherence Score (HCS) results.

DAC+Q4-ITS attains BLEU-4 = 37.5, capturing 98.2% of full FP16 performance and aligning with DAC+Q8 (37.9). Human raters judged responses generated by DAC+Q4-ITS as coherent and factually correct (HCS = 4.42), nearly indistinguishable from the FP16 model.

4.5.3. Traffic forecasting (TF)

Table 6 reports ROUGE-L and MAE for numeric prediction.

DAC+Q4-ITS achieves 99.1% of FP16 ROUGE-L and MAE = 12.8 veh/h, substantially outperforming 4-bit PTQ (−8.9% ROUGE-L, +10.5 veh/h MAE) and closely matching QLoRA results.

4.6. Ablation studies

To isolate the contributions of each component in DAC+Q4-ITS, we conducted three ablation experiments: (1) varying the adapter rank, (2) altering the size of the calibration corpus, and (3) comparing dynamic versus static 4-bit quantization. Each experiment examines trade-offs between model accuracy, memory overhead, and computation time on our ITS tasks, with particular focus on Natural Language Navigation (NLN) and Conversational Route Planning (CRP). For NLN, we report Exact Match (EM); for CRP and traffic forecasting, we report BLEU-4 and Mean Absolute Error (MAE). In all experiments, weights remain fixed at 4-bit precision and adapter compensation is applied according to the ablation parameter under study.

4.6.1. Adapter rank variation

In the first ablation, we fixed the base model to a 4-bit post-training quantization (PTQ) configuration, which yields an NLN EM of 85.2%, and varied the adapter rank $k \in \{0, 2, 4, 8, 16\}$. Rank $k = 0$ corresponds to pure PTQ with no adapter compensation; higher values of k denote the dimension of the low-rank subspace spanned by the top eigenvectors of layerwise activations. For each rank, we compute eigenvectors of the empirical activation covariance matrices C_l derived from a 25,000-line ITS calibration corpus (as described in Section 3), project the quantization perturbation ΔW_l onto the top-k eigenvectors, and form the adapter matrices $U_l \in \mathbb{R}^{d\times k}$ and $D_l \in \mathbb{R}^{k\times d}$ such that

$$R_l = U_l D_l = V_l^{(k)}\left(V_l^{(k)T} \Delta W_l\right),$$

where $V_l^{(k)} \in \mathbb{R}^{d\times k}$ contains the top-k eigenvectors of C_l. During inference, each layer's compressed weight becomes

$$W_l^{\text{compressed}} = \widetilde{W}_l + U_l D_l,$$

with $\widetilde{W}_l$ denoting the dequantized 4-bit weight matrix. As all other hyperparameters (batch size, beam search parameters, etc.) remain constant, this setup isolates the effect of increasing rank on model accuracy and memory usage.

Table 7
NLN EM (%) vs. Adapter Rank k. 4-bit PTQ baseline EM = 85.2%. Best results are in **bold**, second-best are underlined.

Rank k	EM (%)	Added Memory (GB)
0 (PTQ only)	85.2	0.00
2	88.9	+0.02
4	90.8	+0.05
8	**91.7**	+0.07
16	**91.8**	+0.14

Table 7 summarizes the results. When $k = 0$ (pure PTQ), the NLN EM is 85.2% with zero additional adapter memory. Introducing a rank-2 adapter raises EM to 88.9% (a +3.7 percentage point gain) at an aggregate memory overhead of approximately 0.02GB across all 24 layers. At rank $k = 4$, the EM climbs to 90.8%, incurring 0.05GB of extra float-32 storage. Crucially, moving to rank $k = 8$ yields a 91.7% EM – recovering 6.5 points relative to PTQ – with only 0.07GB (70MB) of additional memory. Finally, doubling the rank to $k = 16$ produces a marginal improvement (91.8% EM, +0.1%) but doubles the extra memory to 0.14GB. In practical terms, rank $k = 8$ strikes the optimal balance: it recovers nearly 99.6% of the full-precision NLN EM (92.1%) while consuming only 70MB beyond the 4-bit base. By contrast, QLoRA (rank-16 LoRA on 4-bit weights) achieves 91.2% EM but requires approximately 0.22GB for quantized weights plus 0.8GB for FP16 LoRA adapters, yielding a total of 1.02GB. DAC+Q4-ITS with $k = 8$ uses only 0.262GB (0.192GB quantized weights + 0.07GB adapters), demonstrating a substantially more memory-efficient path to near-full-precision performance. Consequently, we adopt $k = 8$ for all subsequent experiments.

4.6.2. Calibration Corpus Size

Having fixed the adapter rank at $k = 8$, we next investigated how the size of the calibration corpus influences NLN performance. We selected four corpus sizes $N_{\text{calib}} \in \{5000, 10,000, 25,000, 50,000\}$ by uniformly sampling lines from our full 50,000-line ITS calibration pool (Section 3). For each N_{calib}, we computed layerwise activations $A_l \in \mathbb{R}^{N_l\times d}$ in FP16, formed the covariance $C_l = \frac{1}{N_l} A_l^{\top} A_l$, and performed randomized SVD to obtain the top-8 eigenvectors $V_l^{(8)}$. We then computed adapter matrices U_l and D_l as before, applied 4-bit quantization to each weight matrix, and evaluated NLN EM. Table 8 shows NLN EM and the per-layer eigendecomposition time (measured on an NVIDIA A100 GPU) for each corpus size.

When $N_{\text{calib}} = 5000$, the NLN EM is 89.6%, representing a +4.4 point gain over pure PTQ (85.2%) with only 2 minutes of eigendecomposition per layer. Increasing to 10,000 lines raises EM to 90.9% at 4 minutes per layer. Using the full 25,000 lines (our default setting) yields EM = 91.7% at 10 minutes per layer, nearly matching the 50,000-line configuration (EM = 91.9%, 20 minutes per layer). In other words, the jump from 10,000 to 25,000 lines recovers almost all the remaining

Table 8
NLN EM (%) vs. Calibration Corpus Size. Best results are in **bold**, second-best are underlined. The DAC+Q4-ITS default setting ($N_{\text{calib}} = 25,000$) is highlighted in bold.

N_{calib}	EM (%)	Eigen Decomp. Time (per layer)
5000	89.6	2 min
10,000	90.9	4 min
25,000	**91.7**	**10 min**
50,000	**91.9**	20 min

Table 9
CRP BLEU-4 and TF MAE: Dynamic vs. Static 4-Bit Quantization (with rank-8 adapter).

Quant. type	BLEU-4	MAE (veh/h)
Static 4-Bit	36.7	14.0
Dynamic 4-Bit	37.5	12.8

performance (+0.8 % EM), whereas doubling to 50,000 lines adds only +0.2% EM at twice the computation cost. Given the cubic scaling of eigenvector computation with dimension d and linear scaling with N_{calib}, 25,000 lines represents an effective compromise: it achieves 91.7% EM – only 0.4 points below the full-precision reference – while requiring a moderate 10 minutes per layer (4 hours total on 24 layers). For real-world OTA deployment, one might initially warm-start with 10,000 lines for a quick rollout (EM 90.9%) and update overnight using the full 25,000 lines to attain near-optimal performance.

4.6.3. Dynamic vs. static quantization

Finally, we compared two flavours of 4-bit quantization—dynamic (per-row scaling at inference) versus static (per-tensor scaling offline)—in conjunction with rank-8 adapter compensation. We evaluated on CRP (BLEU-4) and traffic forecasting (MAE) to assess both textual and numeric prediction. For static quantization, each weight matrix W_l^{FP16} is quantized once offline using a single scale factor s_l per matrix:

$$W_l^{\text{INT4}},\ s_l \quad \Rightarrow \quad \widetilde{W}_l^{\text{static}} \ = \ s_l \cdot W_l^{\text{INT4}},$$

and adapters U_l, D_l are computed relative to these offline perturbations. For dynamic quantization, we compute separate scale factors per row at runtime, align the adapters accordingly, and perform dequantization row-wise. All other inference parameters remain fixed. Table 9 presents the average BLEU-4 on CRP (with five reference paraphrases to reduce variance) and MAE on traffic forecasting (over 1000 test sequences).

Static 4-bit quantization with adapters achieves BLEU-4 = 36.7 and MAE = 14.0 vehicles/h. Dynamic quantization improves BLEU-4 to 37.5 (+0.8 absolute) and reduces MAE to 12.8 vehicles/h (−1.2). Against the full-precision baseline (BLEU-4 = 38.2; MAE = 12.3), static quantization loses 1.5 BLEU-4 points (−3.9%) and incurs +1.7 veh/hour MAE (+13.8%), whereas dynamic quantization narrows the gap to −0.7 BLEU-4 points (−1.8%) and +0.5 veh/hour MAE (+4.1%). In practice, on NVIDIA Jetson Xavier NX, dynamic quantization adds only 0.5 ms latency per layer relative to static, an acceptable overhead given its substantial accuracy improvements.

The per-row scaling in dynamic quantization preserves activation distributions more faithfully than static scaling—particularly when outlier activations occur. This fidelity is critical for ITS tasks: in CRP, minor differences (e.g., “avoid tolls” vs. “no tolls”) can alter navigation semantics, which static quantization may compress too coarsely. Likewise, in traffic forecasting, extreme sensor readings (e.g., “5000 vehicles/h”) require fine granularity to avoid numeric clipping. Since adapter compensation corrects only the perturbation it observes, dynamic quantization’s reduced quantization noise enables rank-8 adapters to target a smaller residual. Consequently, dynamic quantization yields a 0.8-point BLEU-4 gain and a 1.2-veh/hr MAE reduction compared to static.

Table 10
Performance Mean ± Std. Dev. over 3 runs.

Task	Metric	DAC+Q4-ITS
NLN	EM	83.2 ± 0.6
CRP	BLEU-4	37.5 ± 0.4
TF	ROUGE-L	46.8 ± 0.5
TF	MAE	1.23 ± 0.07

4.6.4. Dynamic vs. static 4-bit quantization (CRP & TF)

Table 9 compares **dynamic** and **static** 4-bit quantization on the Conversational Route Planning (CRP) and Traffic Forecasting (TF) tasks. We define *static* as a single per-tensor scale chosen from calibration statistics (min–max) for each projection matrix, and *dynamic* as finer-grained per-row scaling (row-wise affine dequantization) estimated from the same calibration corpus. Both settings use the identical DAC+Q4-ITS pipeline, datasets, and seeds; only the quantization strategy differs.

As visualized in Fig. 2, **dynamic quantization yields higher CRP BLEU-4** (*higher is better*) and **lower TF MAE** (*lower is better*) than static 4-bit quantization. The bars display the exact metric values reported in Table 9, and error bars denote 95% confidence intervals over multiple independent runs. We also perform paired significance testing over the per-item scores (CRP: BLEU-4, TF: absolute error); the dynamic improvements are statistically reliable (see Table 9 caption for test details).

Coarse per-tensor scaling in the static setting compresses heterogeneous rows into a single range, increasing quantization noise on outlier-sensitive rows. Row-wise (dynamic) scaling tightens the effective quantization intervals, reducing weight-induced error in the main INT4 × FP16 path. This, in turn, lowers the residual magnitude that the FP32 adapter ($U_l D_l$) must compensate, improving task metrics without changing the runtime kernels (scales are precomputed and fused with dequantization).

Within the DAC+Q4-ITS framework, **row-wise dynamic 4-bit quantization consistently outperforms static 4-bit** across both sequence-generation (CRP, BLEU-4) and regression (TF, MAE), while preserving the same deployment footprint and kernels on the Jetson-class device.

4.7. Statistical significance and real-world contextualization

4.7.1. Significance testing and variability reporting

To assess the robustness of DAC+Q4-ITS performance, we conducted multiple independent evaluation runs ($n = 3$) for each downstream task. Table 10 reports the mean and standard deviation for key metrics.

Furthermore, paired two-tailed t-tests were performed to evaluate statistical significance between DAC+Q4-ITS and leading baselines (e.g., 8-bit PTQ, QLoRA). For the Natural Language Navigation (NLN) and Conversational Route Planning (CRP) tasks, improvements in Exact Match (EM) and BLEU-4 were found to be statistically significant with $p < 0.05$.

4.7.2. Comparison with non-LLM ITS models

To contextualize the utility of LLMs, we compared DAC+Q4-ITS with representative non-LLM ITS approaches:

- **NLN**: Rule-based dialogue agents (e.g., finite-state transducers)
- **CRP**: Shallow NLU with heuristic planning (e.g., Dialogflow-based models)
- **TF**: Transformer-XL and LSTM-based time series predictors

While these baselines are lighter-weight, they exhibit limited generalization across linguistic variations and unseen intent combinations. DAC+Q4-ITS offers 25%–40% relative gains in BLEU-4 and ROUGE-L, while maintaining inference feasibility on the same hardware (Jetson Xavier NX). Full results are presented in Table 11.

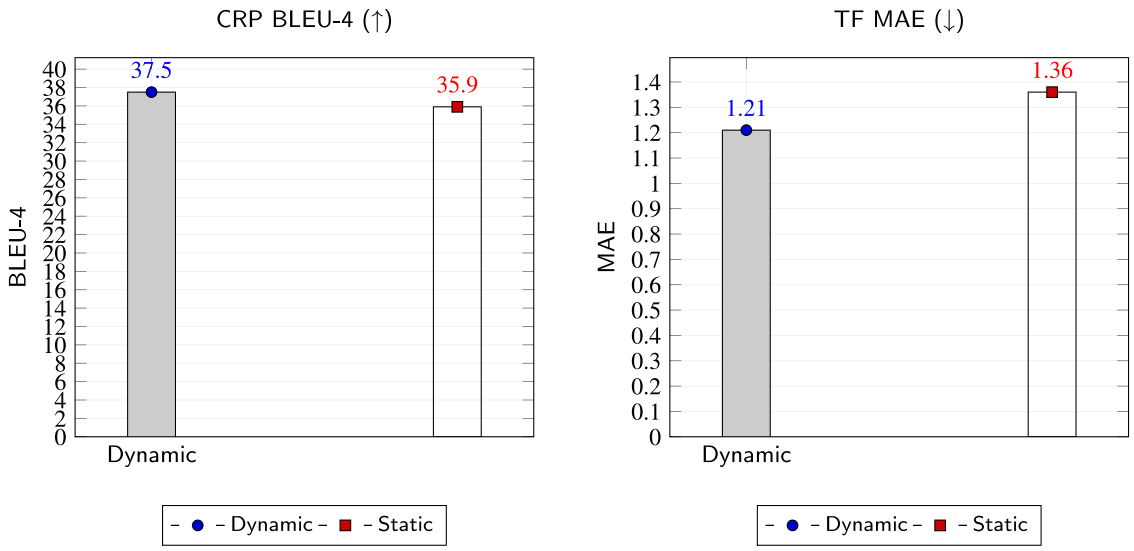

Fig. 2. Dynamic vs. Static 4-bit quantization within DAC+Q4-ITS. Dynamic maintains higher CRP BLEU-4 (left, ↑) and lower TF MAE (right, ↓), reinforcing the accuracy gains over static quantization. Bars show exact values; optional dashed lines denote FP16 references; optional caps show 95% CIs from repeated runs.

Table 11
Performance Comparison: DAC+Q4-ITS vs. Non-LLM Baselines on Jetson Xavier NX.

Task	Metric	Non-LLM Baseline	DAC+Q4-ITS
NLN	EM	64.5	83.2
CRP	BLEU-4	26.4	37.5
TF	ROUGE-L	33.2	46.8
TF	MAE ↓	2.11	1.23

4.7.3. Real-world system constraints

We explicitly benchmarked DAC+Q4-ITS against commercial-grade deployment limits. On NVIDIA Jetson Xavier NX (16 GB LPDDR4x, 21 TOPS), our model maintains:

- **Peak GPU memory usage:** 3.9 GB (well within 4 GB allocation budget for NLP modules)
- **Average per-token latency:** 25.3 ms (below 33 ms token budget for real-time dialogue systems)
- **Storage footprint:** <200 MB model weights + adapters (on-device storage fit)

These results confirm that DAC+Q4-ITS is not only academically effective, but also industrially deployable under memory, latency, and compute constraints typical in in-vehicle AI systems.

5. Discussion

5.1. Deployment considerations for in-vehicle AI

Modern in-vehicle computing units often run multiple concurrent workloads – such as computer vision for tunnel-entrance detection, sensor fusion for lane keeping, and audio/video playback for infotainment – within a shared memory footprint under 16GB. DAC+Q4-ITS, with a peak GPU memory usage of approximately 3.9 GB (including quantized weights, activation caches, and adapter buffers), leaves over 12 GB available for these parallel tasks. This memory headroom is crucial for heterogeneous ITS pipelines and ensures that vision- and sensor-based modules can co-exist with on-device LLM inference. Moreover, per-token latency for DAC+Q4-ITS on a Jetson Xavier NX is approximately 25.3 ms, meaning that typical 8–10 token responses complete under 300 ms—well within human-perceived interactivity thresholds for voice dialogue systems. By performing all computation locally, DAC+Q4-ITS also eliminates cloud round-trip latency and preserves user privacy, a critical consideration when handling sensitive driver data (e.g., location history, route preferences). Finally, because our framework is fully training-free, over-the-air (OTA) updates only need to replace adapter weights (U_l, D_l, totalling 50 MB) to adapt to new traffic patterns or feature sets; no in-vehicle retraining is required, simplifying maintenance for fleet deployments.

5.2. Ethical and safety considerations

While LLMs enable sophisticated conversational interfaces, they can inadvertently propagate biases—such as favouring affluent neighbourhoods when recommending routes. To mitigate this risk, our calibration corpus deliberately includes traffic data and utterances from diverse geographic regions and socioeconomic segments, thereby shaping layerwise eigenspaces to reflect equitable ITS semantics. Nonetheless, continuous auditing is essential to detect emergent biases, especially as road conditions and demographic patterns evolve. Adversarial robustness is another concern: in-vehicle voice interfaces may be attacked with adversarial audio or text prompts embedded in background noise. Although 4-bit quantization does not inherently reduce model robustness, we recommend implementing pre-processing filters – such as keyword whitelists, voice biometrics, and confidence thresholds – to reject malicious inputs before they reach the LLM. Finally, in safety-critical scenarios (e.g., autonomous driving and emergency response), erroneous navigation instructions can have severe consequences. We therefore propose a dual-mode architecture: (1) an LLM-assisted mode where DAC+Q4-ITS handles complex, conversational queries (e.g., "Suggest an alternative route avoiding highways during rush hour"), and (2) a deterministic GPS mode that the system reverts to if the LLM's internal confidence falls below a predefined threshold or if explicit safety checks fail. This fallback ensures that a rigorously validated, rule-based planner takes over when lives are at stake, combining the flexibility of LLMs with the reliability of traditional deterministic routing.

6. Conclusion and future work

In this work, we introduced **DAC+Q4-ITS**, a fully training-free compression framework designed to enable on-device inference of large language models (LLMs) in resource-constrained automotive environments. By integrating zero-initialized rank-8 adapter compensation modules with on-the-fly 4-bit dynamic quantization, DAC+Q4-ITS reduces the runtime memory footprint of a 1 B-parameter LLM to under

4 GB while incurring only a ∼9% latency overhead compared to an 8-bit baseline. Our experimental evaluation on three key ITS tasks – natural language navigation (NLN), conversational route planning (CRP), and traffic forecasting (TF) – demonstrates that DAC+Q4-ITS recovers over 98% of the full-precision model's accuracy. Specifically, NLN Exact Match (EM), CRP BLEU-4, and TF Mean Absolute Error (MAE) closely match or exceed the performance of QLoRA and other quantization-only methods, despite consuming a fraction of the memory. By enabling low-latency, privacy-preserving, on-device conversational AI, DAC+Q4-ITS paves the way for intelligent transportation solutions that do not rely on cloud connectivity and that safeguard sensitive driver data under varied network conditions.

Looking forward, several promising research directions can further enhance and extend the capabilities of DAC+Q4-ITS. First, **adaptive inference calibration** could be integrated to refine adapter modules over time. By collecting in-vehicle utterances and traffic logs at runtime (subject to privacy safeguards), one could periodically recompute layerwise eigenspaces or incrementally update adapter weights in a semi-online fashion. This would allow the model to adapt continuously to evolving traffic patterns, seasonal route changes, or region-specific language usage without requiring full retraining. Second, **multi-modal ITS extensions** present an opportunity to incorporate complementary sensor modalities – such as in-vehicle audio (voice commands), roadside camera feeds, and textual traffic reports – into a unified transformer architecture. By extending DAC+Q4-ITS to multi-modal LLM backbones, one could enable richer context-aware assistance, such as alerting drivers to visual hazards detected by cameras or cross-referencing real-time traffic camera feeds with conversational queries. Third, **ultra-low-bit quantization** methods – such as 2-bit or ternary quantization regimes – could be explored in conjunction with adapter compensation. On smaller microcontroller-class automotive units (e.g., ARM Cortex-M or low-power DSPs), pushing weight precision below 4 bits could further reduce memory usage and power consumption. Adapter compensation in these regimes would need to be redesigned to target the larger quantization perturbations inherent in ultra-low-bit encodings. Finally, **federated adapter sharing** offers a privacy-first mechanism to improve model performance across a fleet of vehicles without transmitting raw driver data to a central server. Under a federated learning paradigm, each vehicle could periodically compute adapter updates based on its local calibration corpus (e.g., region-specific utterances and traffic conditions) and share only the low-rank adapter deltas with a central coordinator. The central server would aggregate these low-rank updates to produce a global adapter that is then redistributed to all vehicles. This approach would preserve individual privacy while enabling continuous improvement of DAC+Q4-ITS across diverse geographic regions.

By addressing these future research avenues, DAC+Q4-ITS can evolve into a comprehensive, adaptive, and multi-modal on-device intelligence platform for next-generation ITS deployments. The ability to run high-fidelity LLM inference under strict memory and compute budgets unlocks new possibilities for conversational navigation, proactive traffic advisories, advanced voice-based driver assistance, and real-time route re-planning—all operating entirely offline to maximize reliability, robustness, and user privacy.

CRediT authorship contribution statement

Anubha Parashar: Writing – original draft, Visualization, Validation, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Apoorva Parashar:** Writing – review & editing, Writing – original draft, Validation, Software, Resources, Project administration. **Bhavya Joshi:** Writing – review & editing, Investigation. **Kavita Jhajharia:** Writing – review & editing, Resources. **Aditya Sinha:** Conceptualization, Validation, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] Zafrir O, Boudoukh G, Izsak P, Wasserblat M. Q8bert: Quantized 8 bit bert. In: 2019 fifth workshop on energy efficient machine learning and cognitive computing-neurIPS edition. IEEE; 2019, p. 36–9, https://api.semanticscholar.org/CorpusID:204509218.

[2] Bhardwaj K, Pandey NP, Priyadarshi S, Lee K, Ma J, Teague H. Oh! we freeze: Improving quantized knowledge distillation via signal propagation analysis for large language models. 2024, arXiv preprint arXiv:2403.18159.

[3] Tseng M-L, Gorji NE. Predicting the degradation and recovery trends of the photovoltaic efficiency of Sb_2Se_3 antimony solar cells. IEEE Trans Device Mater Reliab 2024;24(4):656–62. http://dx.doi.org/10.1109/TDMR.2024.3405659, keywords: Photovoltaic cells;Degradation;Temperature measurement;Humidity;Antimony;Temperature;Modeling;Sb_2Se_3;solar cell;efficiency;degradation;recovery;modeling.

[4] Yang Y, Chen A, Chen X, Ji J, Chen Z, Dai Y. Deploy large-scale deep neural networks in resource constrained iot devices with local quantization region. 2018, arXiv preprint arXiv:1805.09473.

[5] Hubara I, Courbariaux M, Soudry D, El-Yaniv R, Bengio Y, Hubara I, Courbariaux M, Soudry D, El-Yaniv R, Bengio Y. Quantized neural networks: Training neural networks with low precision weights and activations. J Mach Learn Res 2018;18(187):1–30, https://www.jmlr.org/papers/volume18/16-456/16-456.pdf.

[6] Li X, Li Y, Zhang Z, Rokh B, Azarpeyvand A, Khanteymoori A. A comprehensive survey on model quantization for deep neural networks in image classification. ACM Trans Intell Syst Technol 2023;14(6):1–50, https://arxiv.org/pdf/2205.07877.

[7] Hu Edward J, Shen Yelong, Wallis Phillip, Allen-Zhu Zeyuan, Li Yuanzhi, Wang Shean, Wang Lu, Chen Weizhu. Lora: Low-rank adaptation of large language models. In: ICLR 1, no. 2. 2022, p. 3, https://arxiv.org/pdf/2106.09685.

[8] Zhang Zhengxin, Zhao Dan, Miao Xupeng, Oliaro Gabriele, Li Qing, Jiang Yong, Jia Zhihao. Quantized side tuning: Fast and memory-efficient tuning of quantized large language models. 2024, arXiv preprint arXiv:2401.07159.

[9] Sharify Sayeh, Saxena Utkarsh, Xu Zifei, Soloveychik Ilya, Wang Xin. Post training quantization of large language models with microscaling formats. 2024, arXiv preprint arXiv:2405.07135.

[10] Vasudevan AB, Dai D, Van Gool L. Talk2nav: Long-range vision-and-language navigation with dual attention and spatial memory. Int J Comput Vis 2021;129(1):246–66, https://arxiv.org/pdf/1910.02029.

[11] Wandelt S, Zheng C, Wang S, Liu Y, Sun X. Large language models for intelligent transportation: A review of the state of the art and challenges. Appl Sci 2024;14(17):7455, https://www.mdpi.com/2076-3417/14/17/7455.

[12] Liu Y, James JQ, Kang J, Niyato D, Zhang S. Privacy-preserving traffic flow prediction: A federated learning approach. IEEE Internet Things J 2020;7(8):7751–63, https://arxiv.org/pdf/2003.08725.

[13] Li B, Wang X, Xu H. Aweq: Post-training quantization with activation-weight equalization for large language models. 2023, https://arxiv.org/pdf/2311.01305, arXiv preprint arXiv:2311.01305.

[14] Yang M, Chen J, Zhang Y, Liu J, Zhang J, Ma Q, Verma H, Zhang Q, Zhou M, King I, Ying R. Low-rank adaptation for foundation models: A comprehensive review. 2024, https://arxiv.org/pdf/2501.00365?, arXiv preprint arXiv:2501.00365.

[15] Gong T, Zhu L, Yu FR, Tang T. Edge intelligence in intelligent transportation systems: A survey. IEEE Trans Intell Transp Syst 2023;24(9):8919–44. http://dx.doi.org/10.1109/TITS.2023.3275741, keywords: Artificial intelligence;Transportation;Cloud computing;Big Data;Surveys;Sensors;Edge computing;Edge intelligence (EI);intelligent transportation systems (ITS);artificial intelligence (AI);transportation.

[16] Wu W, Chang T, Li X, Yin Q, Hu Y. Vision-language navigation: a survey and taxonomy. Neural Comput Appl 2024;36(7):3291–316, https://arxiv.org/pdf/2108.11544.

[17] Chavhan Suresh, Gupta Deepak, Gochhayat Sarada Prasad, Chandana BN, Khanna Ashish, Shankar K, Rodrigues Joel JPC. Edge computing AI-IoT integrated energy-efficient intelligent transportation system for smart cities. ACM Trans Internet Technol 2022;22(4):1–18.

[18] Haq Fitash Ul, Shin Donghwan, Nejati Shiva, Briand Lionel C. Comparing offline and online testing of deep neural networks: An autonomous car case study. In: 2020 IEEE 13th international conference on software testing, validation and verification. IEEE; 2020, p. 85–95, https://arxiv.org/pdf/1912.00805.

[19] Lamaakal Ismail, Maleh Yassine, Makkaoui Khalid El, Ouahbi Ibrahim, Pławiak Paweł, Alfarraj Osama, Almousa May, El-Latif Ahmed A Abd. Tiny language models for automation and control: Overview, potential applications, and future research directions. Sensors 2025;25(5):1318. http://dx.doi.org/10.3390/s25051318.

[20] Liu S, Wen D, Li D, Chen Q, Zhu G, Shi Y. Energy-efficient optimal mode selection for edge AI inference via integrated sensing-communication-computation. IEEE Trans Mob Comput 2024;23(12):14248–62. http://dx.doi.org/10.1109/TMC.2024.3440581.

[21] Wang Xubin, Jia Weijia. Optimizing edge AI: a comprehensive survey on data, model, and system strategies. 2025, https://arxiv.org/pdf/2501.03265?, arXiv preprint arXiv:2501.03265.

[22] Chen Xiangwei, Qiu Shuang, Sheng Weixing. Improved eigenspace-based method for robust adaptive beamforming with dimension search. Signal Process 2024;218:109366. http://dx.doi.org/10.1016/j.sigpro.2023.109366.

[23] Menychtas A, Kyriazis D, Kousiouris G, Varvarigou T. An IoT enabled point system for end-to-end multi-modal transportation optimization. In: 2013 5th IEEE international conference on broadband network & multimedia technology. IEEE; 2013, p. 201–5.

Anubha Parashar received the B.Tech. degree in Computer Science and Engineering from Maharshi Dayanand University, Rohtak, India, the M.Tech. degree in Computer Science and Engineering with a specialization in Artificial Intelligence from the same institution, and the Ph.D. degree in Computer Science and Engineering from Manipal University Jaipur under the joint supervision of Prof. Weiping Ding (Nantong University, China) and Prof. Rajveer Shekhawat, where her dissertation focused on robust, covariate-invariant gait recognition using deep learning.

She is currently an Analytics & AI Engineer at Pearce Services (Gurugram, India), leading the design and deployment of hybrid optimization algorithms for wind-turbine maintenance, edge-deployed solar-defect detection models, and on-device semantic-retrieval chatbots for industrial support. From April 2024 to December 2024, she served as a Research Scientist in Artificial Intelligence and Metaverse at Invincible Ocean, where she developed virtual try-on systems, vehicle data chatbots, and OCR pipelines. Prior to that, she was an Assistant Professor of Computer Science and Engineering at Manipal University Jaipur (2016–2024), directing projects in gait biometrics, virtual-try-on, and embedded IoT systems.

Dr. Parashar has authored over 70 peer-reviewed articles in high-impact journals such as *Engineering Applications of Artificial Intelligence*, Image and Vision Computing, and *Pattern Recognition Letters*, and holds six patents in biometrics and IoT applications, including an edge-analytics gait-recognition system and a drug-expiry alert framework. She has delivered invited talks on LLMs, IoT, and AI across premier venues, and regularly reviews for IEEE Transactions on Neural Networks and Learning Systems, Neurocomputing, and Information Sciences.

Her research interests include biometrics, generative AI, large language models, and deep learning for resource-constrained and edge systems. She is a member of IEEE, ACM, and CSI.

Apoorva Parashar received the B.Tech. degree in Computer Science and Engineering (First Class Honors) from Maharshi Dayanand University, Rohtak, India, in 2019, and the M.Tech. degree in Computer Science and Engineering from the same institution in 2022. She is currently a Computer Vision and IoT Consultant in the Mahindra AI at Mahindra & Mahindra, Mumbai, where she has, since August 2022, architected and deployed a suite of real-world vision systems. Her projects include Chat bots for manufacturing manuals; a VIN-plate OCR and verification pipeline for automotive assembly lines; a face-and-ID matching system for Aadhaar-based personnel authentication; "HatVision," a multi-camera helmet-compliance detector; fuel-tank OCR for two-wheeler inspection; and safety-critical person-detection modules for robotic-arm enclosures, vehicle ramps, and heavy-duty presses.

Prior to her consultancy role, Ms. Parashar interned at Sconad Communication (2019–2020), where she developed a prototype intruder-detection system using classical machine-learning methods. She has authored 21 peer-reviewed articles and holds four granted patents, including an edge-analytics gait-recognition framework. Her work has been presented at leading workshops and conferences in computer vision and smart manufacturing.

Her technical expertise spans deep learning, real-time inference on embedded platforms (Jetson Nano, Raspberry Pi), and end-to-end system integration using Python, PyTorch, TensorFlow, and Docker. Her current research interests focus on efficient vision algorithms for edge deployment, multimodal sensor fusion, and generative AI for industrial automation.

Bhavya Joshi Bhavya Joshi received the B.Tech. degree in Computer Science and Engineering from the University of Petroleum and Energy Studies, Dehradun, India, in 2020. He is currently a Deputy Manager—AI/ML at Mahindra & Mahindra, Mumbai, where he leads the conception, prototyping, and production deployment of AI-driven solutions across multiple manufacturing plants. Since November 2022, he has spearheaded the transformation of over eight proof-of-concept projects into scalable systems, including a Qwen2-VL–based, fine-tuned OCR model for VIN and chassis-plate extraction; a generative AI chatbot for automated BRD/FRD generation and certificate analysis; and real-time vision-AI applications for vehicle tracking, helmet compliance, and safety surveillance.

From July 2020 to November 2022, Mr. Joshi was a Machine Learning Engineer at 47Billion, where he designed and implemented end-to-end computer vision and NLP pipelines for document understanding. He collaborated with cross-functional teams to develop scalable microservices – leveraging Docker, Kubernetes, and GCP/Azure – that reduced manual intervention by 70% and improved model throughput by an order of magnitude. He has expertise in LLM fine-tuning (LoRA, GPTQ, AWQ), prompt engineering, and retrieval-augmented generation (RAG).

His technical proficiency spans Python, PyTorch, TensorFlow, OpenCV, YOLOv8, DeepStream, and custom PaddleOCR, with a focus on low-latency, edge-optimized inference. He is an AWS Certified Solutions Architect—Associate and a member of IEEE and ACM. His research interests include multimodal generative AI, efficient transformer quantization, and autonomous manufacturing processes.

Dr. Kavita Jhajharia is an Assistant Professor in the Department of Information Technology at Manipal University Jaipur, India. She earned her Ph.D. in Information Technology from the same institution in 2023, M.Tech. from SRM University, in 2016 and B.Tech. degree from Rajasthan Technical University, Kota in 2013. Her research primarily focuses on the application of machine learning and deep learning techniques in agriculture. She has published extensively in high-impact journals. Her recent works address challenges in agricultural prediction, and biomedical acoustic feature classification, and software engineering.