Flash-Fusion: Enabling Expressive, Low-Latency Queries on IoT Sensor Streams with LLMs

Kausar Patherya
Georgia Institute of Technology
Atlanta, Georgia, USA
kpatherya3@gatech.edu

Ashutosh Dhekne
Georgia Institute of Technology
Atlanta, Georgia, USA
dhekne@gatech.edu

Francisco Romero
Georgia Institute of Technology
Atlanta, Georgia, USA
faromero@gatech.edu

Abstract

Smart cities and pervasive IoT sensor deployments have generated significant interest in IoT data analysis across fields like transportation and urban planning. At the same time, the rise of Large Language Models offer a new tool for exploring IoT data - particularly using its natural language interface for improved expressibility. Users today face two main challenges with gathering and interpreting IoT data today with LLMs: (1) data collection infrastructure is expensive, generating terabytes daily of low-level sensor readings that are too granular for immediate use, and (2) data analysis is time-consuming, requiring time-consuming iteration and technical expertise. Directly feeding all IoT telemetry to LLMs is impractical due to finite context windows, prohibitive token consumption (hundreds of millions of dollars at enterprise scales), and non-interactive latencies. What is missing is a system that first parses the user's query to determine the required analytical task, then selects the relevant data slices, and finally chooses the right representation before invoking an LLM.

To address these challenges, we present Flash-Fusion, an end-to-end edge-cloud system that addresses the IoT data collection and data analysis burden on users. Two core principles guide its design: (1) edge-based statistical summarization (achieving 73.5% data reduction) to tackle the data volume problem; and (2) cloud-based query planning that clusters behavioral data and assembles context-rich prompts to tackle the data interpretation problem. We deploy Flash-Fusion on a university bus fleet and evaluate against a baseline that feeds raw data directly to a state-of-the-art LLM. It achieves a 95% latency reduction and 98% decrease in token usage and costs while maintaining high-quality responses. Flash-Fusion enables personas across various disciplines - safety officers, urban planners, fleet managers, and data scientists - to quickly and efficiently iterate over IoT data without the burden of query authoring and preprocessing.

Keywords: Large Language Models, IoT, Edge Computing, Prompt Engineering, Context Injection, Transportation Analytics, Sensor Fusion, Driver Behavior, LLM Grounding, Real-Time Systems

1 Introduction

Smart cities run on the Internet of Things (IoT), where networks of sensors stream data from roads, vehicles, utilities and public spaces, enabling city operators to make informed and timely decisions [33]. This flow of information enriches public transportation systems, resulting in shorter trips, lower emissions and better daily experiences for residents [48]. At the same time, Large Language Models (LLMs) have emerged as powerful tools for processing sensor data [4, 73], particularly as they advance in their reasoning capabilities [65] and support larger context windows [47]. For example, they can interpret smartphone accelerometer data to recognize human activity [66], identify temperature anomalies in industrial machinery [4], or perform sensor fusion of optical and LIDAR data [72]. Users are able to query LLMs using natural language, which enables highly expressive queries over data such as *"Which bus routes are running behind schedule?"* [3].

Consider a fleet manager overseeing a city's bus transit system. To improve passenger safety, the manager would like to identify aggressive driving behaviors (e.g., harsh braking and acceleration) that endanger riders. Identifying such behaviors has two main challenges:

Data collection is expensive: To perform the analysis on the bus transit system, the fleet manager needs to first collect sufficient data. Building a data collection network requires a significant upfront investment in hardware, connectivity, and a scalable cloud backend, with projects often spanning months and costing hundreds of thousands of dollars [50]. The fleet manager likely needs to equip several vehicles with IoT sensors that generate a continuous stream of telemetry data, including GPS and accelerometer readings. The resulting data streams are massive–adding up to terabytes daily for city-wide fleets–and continuous, demanding real-time processing [70]. Cities invest in Intelligent Transportation System (ITS) infrastructure hoping the data can serve multiple departments, from urban planning to public safety. Urban planners study route trends to optimize bus networks, safety officers analyze braking patterns to spot risky intersections, and fleet managers track fuel use and maintenance needs [23]. However, the raw sensor readings are often too low-level and specific to be immediately useful across these disciplines.

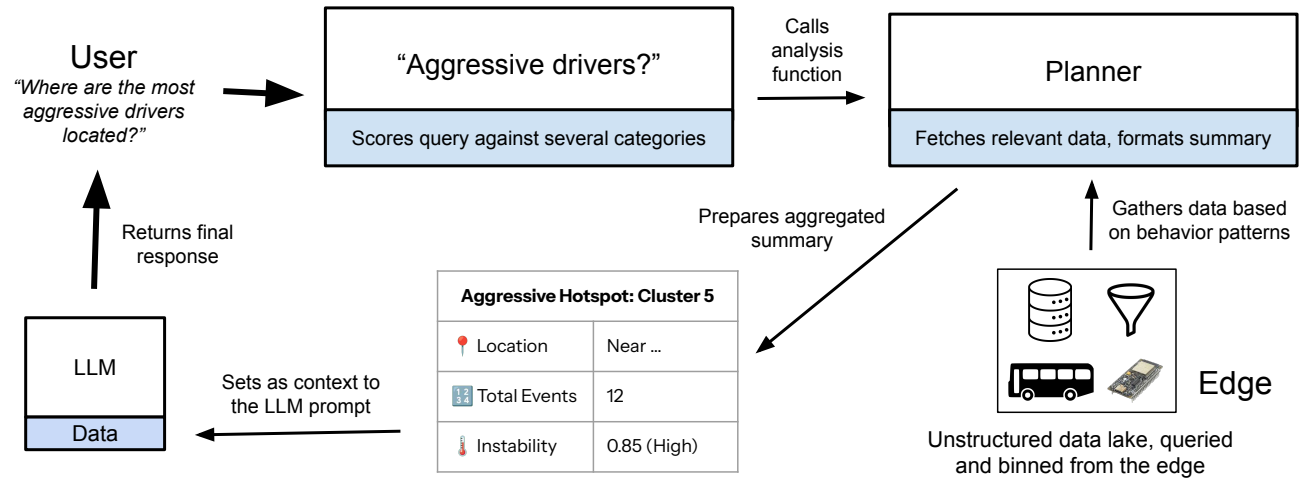

Figure 1. Flash-Fusion: Fast, Grounded Intelligence. Sensor data are aggregated on-device, minimizing transmission overhead. Compact summaries are clustered in the cloud by behavioral type. When a user asks a question, the system analyzes their intent and fuses relevant data into the LLM prompt, yielding responses that are *fast, verifiable, and grounded in real data.*

Data analysis is time-consuming: Even after the data is collected, making sense of it to answer the fleet manager's query is challenging [20]. This process involves building ETL (Extract, Transform, Load) pipelines to feed sensor readings into a database, where analysts must write intricate queries to hunt for patterns. Not only is this approach slow, but it also demands deep technical and domain expertise to distinguish sensor noise from an aggressive driving event [25]. This typically requires domain experts (e.g., the fleet manager) to sit with data analysis teams to carefully craft and iterate on queries to make sense of the data. Depending on the dataset size, this can take anywhere from hours to days [54].

LLMs are promising because they can ingest raw, high-volume sensor data and allow experts and non-experts alike to query it using natural language. This has the potential to reduce the need for intricate query authoring [41]. In principle, the fleet manager could feed the data from the IoT sensor into an LLM, and ask about unsafe driving events instead of relying on a data science team.

Unfortunately, directly funneling raw telemetry into an LLM is impractical because (1) context windows truncate long horizons, (2) per-token costs scale linearly with data volume, and (3) latency grows with input size, hindering interactive use. To illustrate, an enterprise-scale IoT system may deploy millions of devices and process trillions of data packets annually [36]. Assuming each JSON packet contains approximately 600 characters and that one token corresponds to four characters, this yields a monthly token volume of roughly 1.25×10^{13} tokens. At standard API pricing, the monthly input cost alone would reach approximately $375 M [30]. Beyond cost, LLMs are constrained by finite context windows; even with a 10^6-token capacity in models such as GPT-4.1, the window would be saturated in about 0.21 seconds under these data rates [55]. This means the LLM cannot identify long-term trends in the raw data stream because most of the data gets truncated. As a result, when fed noisy, unstructured, and contextless numerical data, LLMs are prone to generating plausible but incorrect information (hallucinations), which is unacceptable for safety-critical applications [10].

We make two observations to enable data-driven IoT telemetry analysis using LLMs. On the data collection side, we must be efficient with the data sent back to the cloud. Instead of transmitting raw sensor streams, the IoT device should only send summaries of on-device sensor readings (e.g., the mean and variance of acceleration over a 3-second window). This must be done carefully, as over-filtering can affect query accuracy and would require expensive data re-collection. On the data analysis side, the lack of a systematic way to explore IoT data with LLMs makes querying them expensive, slow, and oftentimes infeasible. We need a system that extracts the analytical task from the user's query, isolates the relevant data, and chooses the right representation (e.g., summaries, counts, geographic anchor points) before invoking an LLM. This enables the LLM to receive a more structured representation of the IoT data, reducing cost and improving query accuracy.

We present Flash-Fusion (Figure 1), an end-to-end system that lets users ask high-level questions about IoT data and receive accurate responses with both low latency and low cost. During data collection, the edge compresses raw streams into fixed-window statistical summaries (mean/variance of acceleration magnitude plus high-percentile axes with GPS/time

context) before transmission. This summarize-at-source design—a common approach in edge computing [6]—reduces transmission by 73.5% in our study while preserving the features needed for behavior analysis.

Once the condensed data reaches the cloud, the challenge shifts to interpretation: users need insights but lack the expertise to query complex IoT datasets. To alleviate users from spending significant time refining queries, the cloud planner extracts the analytical task from the user's query (e.g., identifying the keyword "dangerous" to trigger an *Aggressive Driving* analysis) and retrieves the appropriate data representation for the LLM. Flash-Fusion's pipeline addresses a key challenge in the ITS domain: bridging the semantic gap between a qualitative user query (e.g., *smoothest ride*) and quantitative sensor data (e.g., raw accelerometer readings). After the firmware summarizes and transmits sensor data, a serverless cloud backend clusters this data into distinct driving profiles, from *Calm* to *Very Aggressive*. At query time, the planner retrieves only the relevant cluster summaries, maps them to campus landmarks, and assembles a compact, context-rich prompt that translates low-level numbers into the high-level behavioral context an LLM needs to provide an accurate, nuanced answer. Flash-Fusion enables personas across various disciplines - safety officers, urban planners, fleet managers, and data scientists - to quickly and efficiently iterate over IoT data without the burden of manual data preparation and scripting.

We demonstrate Flash-Fusion's efficacy on vehicular fleets by deploying its edge module on a university bus fleet. Across a range of real-world queries from different personas, our system achieves a 95% latency reduction and a 98% decrease in token usage and API costs while maintaining high-quality, reliable responses compared to a state-of-the-art LLM baseline `Llama 3.1 8B` [29].

In summary, our primary contributions are the following:

1. A characterization of current limitations in applying LLMs to large-scale IoT telemetry using real vehicular data.
2. The design and implementation of Flash-Fusion, an end-to-end system that integrates on-device data reduction, serverless cloud-based clustering, and an LLM-powered planner to automate the data-to-insight pipeline for IoT analytics.
3. A quantitative and qualitative demonstration of Flash-Fusion's ability to reduce latency and cost while maintaining factual consistency compared to naïve, raw-data prompting of state-of-the-art LLMs.
4. The collection and development of a vehicular transportation dataset and application that we will open-source upon acceptance, broadening the scope of analysis possible for smart city transit systems.

2 Challenges and Limitations

In this section, we highlight the challenges of large-scale IoT data collection and using LLMs to analyze the data. We use the fleet manager example from Section 1 to highlight the practical constraints that motivate Flash-Fusion's design. We examine two key challenges: (1) the data collection problem, deciding what to capture and how to transfer it efficiently, and (2) the data analysis problem, translating user intent into LLM queries over massive sensor streams.

2.1 Data collection challenges

Let us return to our fleet manager example. The manager has equipped each bus with sensors – typically GPS receivers and accelerometers – that capture location, speed, and motion data. The goal is to collect sufficient data to identify safety risks (e.g., aggressive braking) while keeping transmission costs and power consumption manageable.

Two critical design considerations emerge: **(1) How often should data be collected?** Sampling at 10 Hz provides higher temporal resolution for detecting aggressive maneuvers but generates ten times the volume of 1 Hz sampling. For a single bus operating an 8-hour shift at 1 Hz, this amounts to 28,800 individual data points. The manager must balance the need for fine-grained data against the practical constraints of transmission and storage. Sample too infrequently, and critical events may be missed; sample too often, and the system becomes overwhelmed. **(2) Should data be transmitted raw or aggregated?** Simply reducing the frequency of data transmission is not enough, as this risks the loss of important details. The manager must make upfront decisions about what to discard, often without knowing which patterns will be useful for further analysis. Without an intelligent summarization strategy, the manager either over collects – wasting resources on redundant data – or under collects – sacrificing the insights they sought to gain in the first place.

2.2 Data analysis challenges

Once sensor data reaches the cloud, the challenge shifts to interpretation. The manager wants to ask questions like *"Which routes experience the most aggressive driving behavior?"* and receive answers in real-time (within seconds), exploring the data interactively. The manager considers feeding the LLM all the telemetry data with this prompt. State-of-the-art LLMs support context windows ranging from 100K to 1M tokens [45]. In the 8-hour shift example, assuming each data point requires approximately 20 tokens (including GPS coordinates, accelerometer readings, and timestamps), a single vehicle generates roughly 576,000 tokens per shift – eventually exceeding even the largest context windows.

The manager is forced to either subsample the data — analyzing disconnected fragments — or break queries into multiple rounds, manually stitching results together. To illustrate the impact of input size, consider three scenarios:

- **Small sample (100 data points, ≈ 2,000 tokens):** The LLM responds quickly but considers only a narrow window of activity (e.g., 1.5 minutes). It may miss broader behavioral patterns such as morning rush-hour, yielding fast but unreliable results.
- **Medium sample (5,000 data points, ≈ 100,000 tokens):** The LLM processes more data but still cannot see the full picture. Cost increases proportionally—at $0.03 per 1,000 input tokens, this single query costs $3.00. Latency also rises to several seconds.
- **Large sample (28,800 data points, ≈ 576,000 tokens):** The query approaches or exceeds context limits. API costs reach $17.28 per query, and response times become impractical for interactive use.

Even when context limits and costs are managed, LLMs struggle with raw sensor streams. A prompt filled with GPS coordinates like `(33.7756, -88.3963)` and accelerometer readings like `(ax: 0.23, ay: -0.45, az: 9.81)` lacks semantic context. The LLM cannot infer that these coordinates correspond to *"near Union Square"* or that the acceleration pattern indicates *"smooth highway driving"*. When fed raw sensor data, LLMs resort to surface-level matching or, worse, hallucinate.

These observations reveal a critical insight: LLMs require carefully curated, high-level summaries rather than raw telemetry data. Even if condensed data arrives from the edge, current workflows provide no mechanism for selecting and transforming this data based on user intent. Instead, fleet managers resort to manually curating datasets, writing custom scripts, or blindly dumping entire database tables into the prompt until the context limit is reached – an approach that yields incomplete or hallucinated responses.

3 Flash-Fusion Design

From the insights in Section 2, we design Flash-Fusion, using a three-tier architecture (Figure 2) to optimize the data-to-insight pipeline at every stage. The **Edge Tier** (Section 3.1) tackles the data volume problem by performing on-device aggregation, reducing transmission costs while preserving behavioral information. The **Cloud Tier** (Section 3.2) provides scalable storage and formats the data – converting low-level sensor readings into high-level behavioral clusters [12]. Finally, the **LLM Query Engine** (Section 4) automates the query-to-insight process, allowing users to extract answers through natural language without manual data preparation or custom scripting. When the query arrives, the system automatically identifies the relevant time window, retrieves cluster statistics for high-severity braking events, maps them to campus landmarks, and constructs a context-rich prompt for the LLM.

Flash-Fusion simplifies the process for users by eliminating the need to define how to handle their telemetry data.

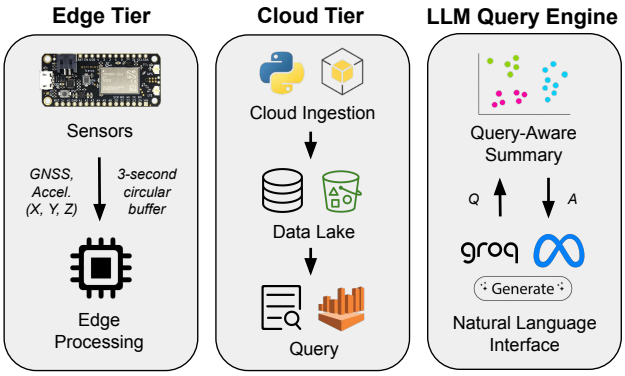

Figure 2. Flash-Fusion's Three-Tier Architecture. Data flows from the Edge (left), where on-device processing reduces data volume, through the Cloud (center) for scalable storage and machine learning, to the LLM (right) where data is transformed into a format suitable for interactive QA.

Users only need to specify the question they want to answer with the data.

A key reason for building clusters is computational efficiency: building behavior profiles per-query would introduce latency in the LLM Query Engine, hindering interactive use. To address this, Flash-Fusion initializes clusters offline on historical summaries and then updates them on a schedule or when drift is detected (e.g., route changes, seasonal shifts). Centroids are versioned and applied consistently at query time to ensure reproducible analyses. This policy preserves generality, bounds compute cost, and avoids query-time re-clustering while still adapting to evolving data. In the following sections, we show how the edge turns raw signals into summaries and the cloud turns summaries into behavioral structure.

3.1 From raw signals to statistical summaries

To address the challenges in Section 2.1, Flash-Fusion's edge tier requires hardware capable of: (1) high-frequency sensor data acquisition (≥ 20 Hz for accelerometer, ≥ 1 Hz for GPS); (2) on-device compute for real-time statistical aggregation; (3) low-power connectivity suitable for mobile deployments; and (4) extended battery operation for minimal-maintenance deployments.

Edge processing must balance capturing high-fidelity behavior with minimizing power use and transmission costs. At 20 Hz, raw accelerometer data produces ≈ 39.5 MB per day (plus 11.9 MB of GPS data), quickly draining batteries and raising cellular costs. This data volume problem is common across vehicular sensing systems, where continuous high-frequency monitoring conflicts with resource constraints [35].

To address this, Flash-Fusion aggregates acceleration data into fixed-duration time windows, within which it computes

the normalized acceleration magnitude $\sqrt{x^2 + y^2 + z^2}$, and extracts statistical features from this time series. Instead of transmitting raw sensor readings, which would generate a payload of

$$P_{\text{raw}} = (f \times T \times N_{\text{axes}} \times B_{\text{reading}}) + P_{\text{gps}}, \tag{1}$$

where f is the sampling frequency, T is the window duration, N_{axes} is the number of axes, B_{reading} is the bytes per reading, and P_{gps} is the GPS payload, the system transmits a compact summary including mean and variance, and percentile analysis (p1, p10, p90, p99) for each axis. This captures routine patterns (mean/variance) and outliers (percentiles) [24].

Although aggregation removes sample-level detail, it retains the statistical features needed to classify driving behavior [32]. More broadly, this summarization approach characterizes time-series patterns across a wide range of IoT applications, from industrial monitoring to environmental sensing. Prior work shows that harsh braking and acceleration unfold over multi-second intervals [37], a duration well captured by windowed statistics. The resulting features—mean, variance, and percentiles—provide reliable signals for distinguishing smooth driving from erratic behavior and sudden shocks across diverse IoT deployments.

3.2 Transforming summaries into insights

Once edge summaries arrive in the cloud, the pipeline shifts from compression to semantics: storing summaries for scale, then structuring them for analysis and querying. The cloud tier must support: (1) scalable ingestion of continuous data streams from multiple vehicles; (2) efficient querying by time, location, and behavioral attributes; and (3) cost-effective long-term retention for historical analysis.

We implement a serverless cloud pipeline that decouples ingestion from analysis. Aggregated summaries flow from edge devices to an object storage layer, organized hierarchically by vehicle and timestamp. This schema-on-read approach eliminates complex ETL pipelines, enabling flexible querying through serverless SQL query engines [13]. To minimize storage costs, we leverage the pre-aggregated edge data, apply JSON compression, and implement lifecycle policies for archival. We filter for GPS fix quality, ensuring any location-based analyses use only high-confidence positioning data, a standard practice in geospatial applications [34].

The goal of the Cloud Tier is to bridge the semantic gap between high-level qualitative queries and low-level sensor readings. The challenge is determining how to structure the latter so the LLM Query Engine can efficiently identify relevant data and construct meaningful LLM prompts. For example, when the fleet manager asks about *aggressive driving*, the system needs to determine which subset of the data matches that concept.

We address this through offline behavioral clustering rather than alternatives like training small supervised learning models or rule-based thresholds. There are two benefits to clustering for query planning. First, it allows us to map feature vectors to a set of discrete labels (e.g., mapping the centroid of a low-instability cluster to the label *Calm*). This creates a shared vocabulary between the system and the user. Second, clusters can be built once and reused across many queries, similar to indexes built for applications like video analytics [38].

We choose K-Means for its computational efficiency, deterministic convergence, and simplicity. Its implicit assumption of spherical clusters is well-suited to our feature space, where behavioral attributes increase monotonically along each feature dimension. This creates naturally separable groups that K-Means can effectively partition. While more sophisticated methods like DBSCAN or hierarchical clustering could capture arbitrary cluster shapes, their added complexity is unnecessary here [40].

Within each sensor window (Section 3.1), we compute domain-specific features that capture the patterns relevant to anticipated queries. The choice of features depends on the application: for fleet management, we compute (1) *extreme event magnitude* - the Euclidean norm of the 99th percentile acceleration values across axes: $\sqrt{x_{p99}^2 + y_{p99}^2 + z_{p99}^2}$, which captures the severity of sudden maneuvers like sharp turns or harsh braking; and (2) *instability score* – the variance of acceleration magnitude, which measures motion smoothness. More generally, features should summarize central tendency (mean, medians) and variability (variance, percentiles) to enable diverse queries. For example, environmental monitoring uses thermal gradients and temporal gradients [62], while industrial IoT might track vibration intensity and periodicity [26].

Both features - extreme event magnitude and instability score - are normalized using Z-scores to ensure equal contribution during clustering. The K-Means objective minimizes intra-cluster variance:

$$J = \sum_{j=1}^{K} \sum_{x_i \in C_j} \|x_i - \mu_j\|^2 \tag{2}$$

where μ_j is the centroid of cluster C_j. For fleet management, we chose $k = 5$ (*Calm*, *Moderate*, *Slightly Unstable*, *Aggressive*, and *Very Aggressive*) clusters based on domain knowledge — transportation safety officers use similar classifications [11] — and empirical validation via elbow analysis.

Flash-Fusion's clustering framework can also be extended to other domains. Applications can define clusters that align with their domain vocabulary - manufacturing might use *Normal Operation*, *Degraded*, *Critical*, while smart buildings might cluster into *Peak Load*, *Reduced Load*, *Standby* [18].

4 LLM-Enabled Insight Generation

The final component of Flash-Fusion enables users to extract insights from sensor data through natural language interaction. This addresses the core challenge of applying LLMs to IoT data: their inability to reason about raw, high-volume sensor streams due to context limits, cost, and latency. Users pose questions such as *"Where is driving most dangerous?"*, leaving Flash-Fusion to handle three critical functions: (1) extracting the user's intent from their query, (2) retrieving and aggregating the relevant subset of data, and (3) constructing a context-rich prompt that grounds the LLM's response in empirical evidence. This layer transforms user queries into data-backed answers without requiring knowledge of SQL, clustering algorithms, or data schemas.

LLM configuration: For LLM calls, Flash-Fusion uses the `Llama 3.1 8B model`. This provides it with low-latency, high-quality responses suitable for real-time conversational interfaces [29]. Task-specific parameter tuning is applied to optimize response quality. Macro-level queries use a temperature of 0.7 and 500-token limit, allowing for greater narrative flexibility in high-level queries [52]. Micro-level insights use a more deterministic temperature of 0.5 and a 150-token limit to ensure factually focused and concise diagnostic responses.

To enhance user experience and reduce costs, Flash-Fusion implements a caching mechanism. Before prompting the LLM, the backend generates a hash of the prompt's content. This hash serves as a key to check an in-memory cache for previously generated responses. If a match is found, the cached response is returned instantly, bypassing a redundant and costly API call [9].

4.1 Querying Flash-Fusion

Users interact with Flash-Fusion through a natural language interface that supports a constrained set of analytical intents [44]. For instance, users can ask about *idle spots* or *wait duration*, and the system will recognize both as semantically equivalent responses for dwell time analysis.

While simple, this approach is fast and deterministic. A future implementation could use an LLM to propose analytical categories. By providing the model with a high-level description of the domain (e.g., *fleet management for passenger safety*), it can generate a candidate set of intents and associated keywords [17].

Data retrieval. Once the intent is classified, the system triggers the corresponding aggregation function, which operates on the pre-clustered indexed dataset from Section 3.2. For example, the query above executes a function that identifies the top-k locations with the highest frequency of *Aggressive* cluster labels, computes their average instability scores, and maps them to named campus landmarks. This targeted retrieval ensures that only relevant data is processed, reducing cost and latency compared to scanning the entire corpus.

4.2 Providing the relevant context

A major obstacle to using LLMs for IoT data analysis is their sensitivity to input format and volume. As described in Section 1, dumping thousands of JSON objects into a prompt is both expensive and ineffective. To address this, the LLM Query Engine retrieves a data summary relevant to the analytical task and constructs a structured, high-signal prompt. For instance, when analyzing aggressive driving hotspots, the system generates a concise text block:

Aggressive Driving Summary

Total events: 2,450
Observation window: 2024-10-21 14:00–15:30 UTC
Hotspots:

- Location A: 15 events, instability 0.87
- Location B: 11 events, instability 0.75
- Location C: 8 events, instability 0.81

This summary is then injected into a category-specific prompt template that instructs the LLM on how to interpret the provided statistics. By receiving only the necessary context – typically 50-100 tokens instead of tens of thousands – the LLM can provide faster, cheaper, and more accurate responses. This approach leverages the LLM's strength (natural language synthesis) while avoiding its weaknesses (raw numerical reasoning).

4.3 Response validation

LLMs are prone to hallucinate – generating plausible but factually inaccurate statements – particularly when working with structured data [8]. For our motivating example, when the fleet manager tries to identify instances of aggressive driving, the LLM may misstate event counts, invent street names, or misattribute behavior to non-existent features. To ensure every insight is empirically grounded, Flash-Fusion performs a set of automated validation checks on the LLM's generated response [22].

Flash-Fusion's validation checks fall into three categories: (1) *factual consistency:* the system verifies that any quantitative statements in the response (e.g., "15 events", "instability 0.87") directly match the source data provided in the prompt; (2) *geographic grounding:* any mentioned landmarks are cross-referenced against a provided directory (e.g., a city's official map data) to prevent the fabrication of place names; (3) *behavioral alignment:* the narrative tone of the response is checked to ensure it aligns with the event's behavioral classification (e.g., a *Calm* event is not described with alarming language). Additional checks screen for generic AI language, such as apologies or refusals.

To quantify the reliability of a response, we define a simple scoring heuristic based on the number of validation issues detected:

Table 1. Examples of Flash-Fusion Responses across Macro & Micro-Level Driving Contexts. Flash-Fusion can draw summaries from all the driving data (macro-level) and from a specified set of datapoints (micro-level).

Task	Keyword	Example Query	Flash-Fusion Response
Macro-Level (All Data)			
Aggressive Driving	Behavioral	"Tell me about aggressive driving behaviors around campus."	"Aggressive events cluster near Union Square (16 such events)."
Dwell Time	Temporal	"Which zones show the longest dwell times?"	"Vehicles dwell longer near West Campus Housing."
Moderate Behavior	Event Counting	"How many instances of moderate driving?"	"593 instances of moderate driving behavior were observed."
Route Efficiency	Operational	"Compare route efficiency between morning and evening."	"Evening runs show longer idle durations and reduced speed."
Spatial Patterns	Geospatial	"What are driving patterns like around Union Square?"	"High-density congestion patterns near the square."
Micro-Level (Select Data)			
Sudden Braking	Very Aggressive	"Why did the driver brake near Library Crosswalk?"	"Pedestrian-triggered stop near a high-traffic zone."
Sharp Turn	Slightly Unstable	"What caused lateral instability at Finch & Green?"	"Sharp turn consistent with a tight 90° intersection maneuver."
Vertical Shocks	Aggressive	"Are repeated shocks localized to poor road segments?"	"Repeated vertical shocks may indicate poor pavement quality."

$$S = \max(0,\ 100 - 20 \times N_{\text{issues}}), \tag{3}$$

where N_{issues} is the number of validation issues detected. This score allows the system to programmatically handle outputs based on their confidence level. For instance, low-confidence responses ($S < 60$) can be flagged for review or retried with a modified prompt, while high-confidence responses ($S \geq 80$) can be directly presented to the user. This end-to-end approach of preparing data *before* the LLM call, structuring the prompt *during* the call, and validating the response *after* the call ensures responses are reliable.

4.4 Micro-level event analysis

Beyond broad queries, users can explore individual data points on the map and generate LLM analyses for each, triggering the micro-level insight pipeline. This system generates highly specific, empirically grounded explanations for individual driving events. For example, when a user selects the *"Sudden Braking Event"* from Table 1, the pipeline first determines the the closest landmark to this event (*"Library Crosswalk"*). It then scans nearby data points (within a 100-meter radius) to find local patterns, such as high pedestrian traffic. The event's cluster classification (*Very Aggressive*), its location, and the neighborhood context are combined into a detailed prompt for the LLM.

5 Implementation

Flash-Fusion is implemented in ≈4,000 lines of code (LoC) spanning three components: the edge firmware (≈1,000 lines of C), the cloud data pipeline (≈300 lines of Python and SQL), and the LLM query interface (≈3,000 lines of Python).

Edge Hardware: Telemetry collection and on-device aggregation is performed by the Nordic nRF9160 System-in-Package (SiP), deployed on an Actinius Icarus carrier board [14, 51]. We use the nRF9160 because its SiP architecture provides a single-chip solution that meets all our edge requirements: an Arm-Cortex-M33 processor for real-time sensor data processing, an integrated LTE-M/NB-IoT modem with power-saving modes (PSM and eDRX) for efficient data transmission, an integrated GPS receiver, and a LIS2DH12 3-axis accelerometer [60, 68]. Other potential off-the-shelf options that provide comparable LTE-M/NB-IoT, GNSS, and low-power capabilities include Quectel BG95-M3, u-blox SARA-R5, and Sierra Wireless HL78 [5, 21, 49].

Firmware: The edge software runs on Zephyr RTOS using event-driven work queues and circular buffers to decouple

acquisition from processing and prevent data loss during transmission delays—following edge-computing best practices [19].

Data Ingestion: Data flows from the Actinius Icarus board to Amazon Web Services (AWS) through a Python-based bridge application running on a host computer. Amazon S3 is our primary data lake, where sensor streams are organized hierarchically by vehicle and timestamp. AWS Athena provides SQL access to the S3 data lake using a schema-on-read approach.

6 Evaluation

Our evaluation aims to answer the following research questions:

1. How does Flash-Fusion compare to the `LLM Only`, raw-data baseline in terms of latency, token length, and cost? (Section 6.2)
2. Can Flash-Fusion produce grounded, useful LLM responses and mitigate hallucinations? (Section 6.3)
3. How effective is edge aggregation at reducing data volume while preserving the statistical features needed for behavioral analysis? (Section 6.4)
4. Can K-Means clustering produce well-separated, interpretable behavioral profiles from the aggregated edge data? (Section 6.5)

6.1 Setup

Baselines: We compare against a version of Flash-Fusion that directly sends the telemetry data to the LLM (`LLM Only`). When the dataset exceeds the model's context window, `LLM Only` employs a random sampling strategy: the data is first compressed by shortening JSON keys, then partitioned into chunks of approximately 3,000 tokens each. To manage processing time and API rate limits (30 requests/minutes, 6,000 tokens/minute), the system caps analysis at a maximum of 5 randomly sampled chunks, distributed evenly across the entire dataset. Each chunk is sent to the LLM with instructions to extract relevant observations. These intermediate findings are then aggregated in a final synthesis step, where the LLM weaves a single, coherent answer. This "summarize and synthesize" strategy, where a model processes chunks of a document and then synthesizes the results, is a common approach for querying documents that exceed a model's context window, as seen in long-context evaluation benchmarks such as LongBench [7].

Dataset: We collect telemetry data (timestamps, GPS coordinates, and 3-axis accelerometer readings) on a university system's bus fleet operating on its highest-traffic route for a period of 2 hours and 14 minutes. During this time, 2,648 distinct data points (1.06MB of data) are stored. We will open-source this dataset upon paper acceptance.

Metrics: We use the following metrics in our evaluation:

- *Latency:* to measure the time from the moment a user submits a natural language query until a complete response is returned, excluding artificial rate-limiting delays. For Flash-Fusion, this is the end-to-end processing time. For the `LLM Only` baseline, we report the end-to-end API latency call (actual model computation time) and exclude the chunk processing overhead. This is done assuming that other users' LLMs are not constrained by rate and token limits. Lower latency is better (i.e., improved user experience).
- *Token Usage:* to measure both input tokens (prompt size, including all injected context) and output tokens (generated response length). Token counts are extracted directly from the API response metadata rather than estimated. Lower token usage is better, as it impacts both response latency and cost.
- *Cost:* to be computed as the sum of input cost ($0.05 per 1M tokens) and output cost ($0.08 per 1M tokens) for the `Llama 3.1 8B` model via Groq [57]. For multi-step queries (`LLM Only` with chunking), we report cumulative cost across all API calls. Lower cost is better.

Procedure: We design five natural language queries, each representing a distinct analytical task relevant to fleet management (i.e., *Macro-Level* from Table 1). We run each query three times for both Flash-Fusion and `LLM Only`. For each run, we record end-to-end latency, token usage, and cost (Section 6.2). Each generated response is then qualitatively assessed (Section 6.3). The query order is randomized to prevent caching effects.

6.2 End-to-end performance

We first evaluate how Flash-Fusion compares to the `LLM Only` baseline in terms of latency, token length, and cost. Using the five-query benchmark described earlier, we measure the latency, token counts extracted from API metadata, and the corresponding costs.

Figure 3 shows that Flash-Fusion achieves 95% lower latency, returning answers in under one second, than the `LLM Only` baseline. The whiskers on the box plot represent the full range of observed latencies across three runs for each query. This latency reduction happens for two reasons: (1) Flash-Fusion requires only a single API call, whereas the `LLM Only`'s chunk-analyze-synthesize pipeline totals six API calls, each introducing another round of latency, and (2) aggregating the relevant data before querying the LLM shrinks the prompt size from tens of thousands of tokens to a few hundred, directly reducing LLM inference time. The low variance in Flash-Fusion's performance demonstrates the stable, reproducible nature of our approach. The higher variance in the `LLM Only` baseline reflects the variable overhead linked to multi-chunk processing.

Table 2 quantifies the efficiency gains in token usage and cost. Flash-Fusion achieves a 98% reduction in both token

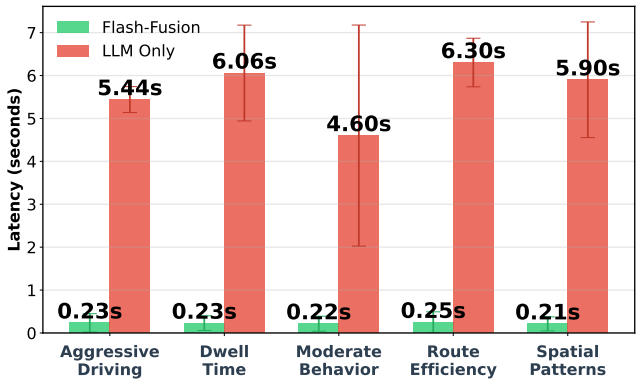

Figure 3. Flash-Fusion Latency versus `LLM Only`. Flash-Fusion cuts query latency by 95% compared to the `LLM Only` baseline across five queries. Its single condensed API call replaces six sequential calls for chunk processing and synthesis. Whiskers show the minimum and maximum latencies over three query runs.

Table 2. Cost and Token Comparison. Flash-Fusion reduces total token usage (input + output) and API costs by over 98% on average across all queries.

Query	Mode	Tokens	Cost ($\times 10^{-5}$ USD)	Red.
Aggressive Driving	Flash-Fusion	423	2.68	**98.3%**
	LLM Only	30,956	159	
Dwell Time	Flash-Fusion	505	2.97	**98.2%**
	LLM Only	31,820	164	
Moderate Behavior	Flash-Fusion	384	2.35	**98.5%**
	LLM Only	31,323	161	
Route Efficiency	Flash-Fusion	363	2.22	**98.6%**
	LLM Only	31,423	162	
Spatial Patterns	Flash-Fusion	406	2.54	**98.4%**
	LLM Only	31,135	160	

usage and cost. By sending a condensed textual summary, Flash-Fusion prompts are consistently under 500 tokens. The `LLM Only` baseline, however, consumes over 30,000 tokens per query across its six API calls. Lower token usage translates to proportional cost savings, making Flash-Fusion a more scalable and economically viable solution. The consistency of this reduction across diverse query types (aggressive driving, dwell time, route efficiency) demonstrates that the approach generalizes well across analytical tasks.

6.3 Assessing answer quality

This section evaluates Flash-Fusion's ability to produce more reliable and actionable insights than the naïve approach of feeding raw telemetry to an LLM (`LLM Only`).

Due to rate limits, `LLM Only` cannot process the full dataset in a single query (15–20 minutes per run). Instead, it uses a "summarize and synthesize" strategy, a standard approach for queries that exceed an LLM's context window [7]. The model receives five randomly sampled chunks of telemetry—GPS coordinates, timestamps, and 3-axis accelerometer readings—and must extract relevant events from each before stitching them into a single answer. This requires it to perform both numerical aggregation and natural-language reasoning. Flash-Fusion, by contrast, provides the LLM with one concise, pre-aggregated summary generated by our query planner (Section 4.2). This summary includes high-level, human-readable features such as behavioral cluster counts, average instability scores, and named locations, all tailored to the user's query.

Figure 4 shows a side-by-side comparison for three representative queries. Flash-Fusion consistently produces geographically grounded, factual responses. When asked about dwell time, it correctly reports *"170.6 minutes near West Campus Housing"* and provides plausible causal hypotheses like traffic or pedestrian congestion. For the event counting query, it accurately reports all 593 instances of moderate driving behavior found in the dataset. In contrast, `LLM Only` returns uninterpretable GPS coordinates and acknowledges incomplete coverage in every response. For the counting query, it reports 27 instances but warns *"these instances are not representative of the entire dataset."* It also uses more technical jargon – terms like chunks, sampled, and dataset – that expose the sampling mechanism to end-users.

These differences stem directly from the design choice to decouple statistical summarization from natural language synthesis. As detailed in Section 4.2, Flash-Fusion prepares a prompt with aggregated, high-level context (e.g., landmark names, behavioral classifications) prior to the API call. The `LLM Only` baseline simultaneously acts as a data analyst and a narrator, a task it performs poorly on raw telemetry, resulting in responses that are overly verbose, less confident, and less useful.

6.4 Aggregating data at the edge

The section quantifies the data reduction achieved by edge aggregation compared to raw transmission. We deploy Flash-Fusion's edge module on a university bus for a 2-hour 14-minute mission. The accelerometer samples at $f = 20$ Hz and GPS at 1 Hz. Three-second aggregation windows are sufficient to capture complete maneuvers without fragmentation. At 30 mph, a bus covers approximately 40 m in 3 s - a scale appropriate for mapping events to specific intersections or road segments.

Without edge processing, each 3-second window generates 1,584 bytes (1,440 bytes from the accelerometer + 144 bytes from GPS). Our aggregated packets averages 420 bytes, yielding a 73.5% per-transmission reduction. Over the full 2-hour and 14-minute mission, the impact is even more significant: a naïve system would have transmitted a projected 4.00 MB of raw data, whereas the Flash-Fusion platform

Question	Flash-Fusion	LLM Only
Which parts of campus did the bus have the highest dwell time?	170.6 minutes near West Campus Housing. This suggests traffic congestion or issues causing extended dwell times like… ✓ Location names clearly mentioned	The area around coordinates [33.773, -84.401] and [33.774, -84.402] have high frequency and duration. Based on analysis of 5 random chunks… ✗ Raw coordinates with data limitation
How many instances of moderate driving behavior?	593 instances of moderate driving behavior were observed. This was the most common behavior, suggesting traffic rule following… ✓ Complete dataset view	27 instances between 16:33:37 and 16:31:46. However, these instances are not representative of the entire dataset, as other chunks show no instances… ✗ Partial sample with accuracy warning
Tell me about aggressive driving behaviors.	Two distinct hotspots where aggressive driving is common. The cluster near Union Square had 16 events. Consider targeted interventions… ✓ Geographically grounded	Based on the 5 random chunks, there is evidence of aggressive driving. In chunk 3, specific instances were observed between 16:01 and 16:04… ✗ Technical jargon specific to chunking

Figure 4. Response Comparison between Flash-Fusion & `LLM Only`. Flash-Fusion produces geographically grounded and actionable responses, while `LLM Only` returns uninterpretable coordinates, data limitation disclaimers and technical jargon.

transmits only 1.06 MB. This translates directly to lower cellular costs and extended battery life. Moreover, the aggregated data retains all features necessary for downstream analysis - behavioral clustering, route efficiency, and anomaly detection all operate on transmitted statistics without requiring raw samples.

6.5 Assessing cluster generation quality

This section evaluates whether the behavioral clustering produces meaningful, well-separated categories. Table 3 summarizes the cluster profiles. The event counts show a realistic distribution: most driving is moderate, with aggressive events forming a critical but rare tail.

Figure 5 shows the resulting cluster assignments in 2D feature space. The clusters exhibit clear separation: *Calm* events concentrate in the bottom-left (low instability, low magnitude), while *Very Aggressive* events occupy the top-right as distinct outliers. These clusters are well-separated, which is confirmed by a silhouette score of 0.616 - a score above 0.5 indicates that a reasonable structure has been found. The centroids of each cluster (marked with a black 'X') are well-spaced, confirming that the algorithm has identified genuinely different modes of operation.

This clear separation occurs because the two features capture complementary aspects of driving behavior. Instability score measures overall smoothness, while extreme event magnitude captures peak forces during sudden maneuvers. Together, they provide sufficient information to distinguish routine operation from safety-critical events.

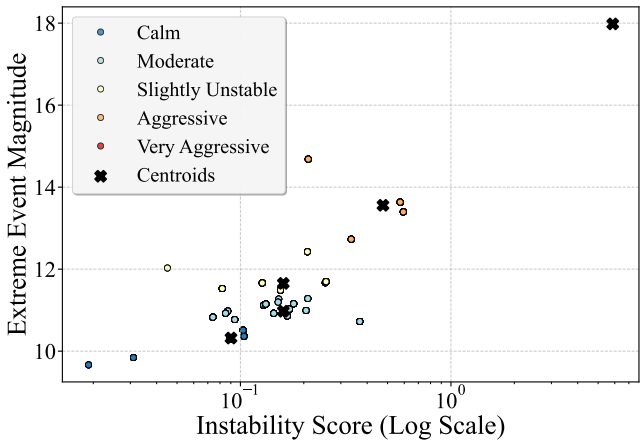

Figure 5. Separation of Behavioral Clusters. There is a clear distinction between clusters by examining the instability and extreme event magnitude, with a log scale highlighting a progression from calm to aggressive behaviors. The *Very Aggressive* cluster stands out as a high-impact outlier.

7 Discussion

In this section, we discuss the scope of our evaluation and future research directions that build upon the Flash-Fusion framework.

Table 3. K-Means Cluster Profiles for Driving Behavior. We show the five behavioral clusters discovered by K-Means. *Instability* refers to the variance of acceleration magnitude (a measure of smoothness), while *Extreme Event* is the 99th percentile acceleration norm (a measure of peak force). Most driving is moderate, with aggressive events forming a critical but rare tail.

Cluster	Event Count	Instability	Extreme Event	Characteristics
Calm	186	0.09	10.32	Smooth driving, minimal extreme events.
Moderate	593	0.16	10.97	Common in stop-and-go traffic.
Slightly Unstable	260	0.16	11.65	Sharp turns or uneven roads.
Aggressive	164	0.48	13.56	Moderate extreme events.
Very Aggressive	16	5.87	17.98	Harsh braking or acceleration.

Data Quality & Sensor Fusion: During data collection, data points occasionally appear on sidewalks or building facades rather than roadways, despite the certainty that the bus remained on the designated route. Multipath effects and non-line-of-sight signals can introduce positioning errors of several meters, especially in dense urban areas [39]. These accuracy constraints could impact the spatial analysis of behavioral hotspots, particularly when determining whether aggressive driving events occur precisely at intersections versus adjacent areas. Future work in sensor fusion could explore techniques where LIDAR odometry or 3D map-matching could be integrated to augment and correct raw GNSS data, leading to precise geospatial analysis [46].

Addressing Data Imbalance: The data imbalance observed in our behavioral clustering - where safety-critical *Very Aggressive* events constitute only 1.31% of the dataset - is typical of real-world anomaly detection tasks. While Flash-Fusion captured these rare cases, future work could improve robustness through (1) one-class classification, which models normal behavior and flags deviations as anomalies [67], or (2) synthetic data generation, using GANs to create realistic aggressive driving samples and balance the dataset [16].

8 Related Work

Flash-Fusion integrates three research domains: (1) **large-scale IoT data analytics**, for managing high-volume sensor streams; (2) **edge-cloud systems**, providing the architectural foundation for efficient data processing; and (3) **context-aware AI**, where the gap between event detection and human-intelligible explanation is explored.

8.1 Large-scale IoT data analytics

The deployment of IoT sensors in smart cities generates massive data volumes that challenge traditional analytics pipelines. A key problem is bridging the semantic gap between raw numerical data (e.g., GPS coordinates, accelerometer readings) and operational insights (e.g., driver safety, route efficiency) [53, 56]. Current systems often rely on machine learning models like support vector machines (SVMs), random forests, or deep learning architectures (LSTMs, CNNs) to classify events from sensor data [2, 27, 58, 59]. These models excel at identifying *what* happened (e.g., hard-braking event) but typically fail to explain *why* it happened, as they often lack real-world context like traffic conditions, road quality, or weather [64]. This limitation creates a need for systems that can not only detect patterns but also provide nuanced, context-aware explanations.

8.2 Edge-cloud systems

To manage the voluminous data streams from IoT devices, the literature strongly supports distributed architectures that span the edge-cloud continuum [28]. Information is processed locally on or near the device to reduce data transmission, minimize power consumption, and lower operational costs [69, 71]. In vehicular settings, this involves on-device aggregation of sensor readings before transmission to a central cloud platform.

Edge systems for mobile assets combine three components: efficient hardware, low-power connectivity, and an optimized operating system [1]. Recent hardware solutions, such as System-in-Package (SiP) designs that combine a microprocessor with an LTE modem and GPS, provide compact, energy-efficient platforms ideal for asset tracking. These are often paired with low-power wide-area network (LPWAN) technologies like LTE-M or NB-IoT, which are designed for long battery life in mobile deployments [31, 43]. The cloud backend typically employs serverless components, where a data lake provides scalable object storage and a query engine enables SQL-based analysis [42]. To summarize, the edge provides immediate, real-time processing, and the cloud offers vast scale and analytical power [15]. Flash-Fusion leverages this edge-cloud architecture as its data pipeline.

8.3 Context-aware AI with LLMs

A key limitation of traditional IoT analytics is the context gap - the inability to explain events in human-intelligible terms [63]. LLMs have emerged as a promising approach to bridge this gap. While early applications focused on text generation, recent work demonstrates their capability in structured data tasks, such as tabular anomaly detection [61]. LLMs have thus transitioned from simple text generators to versatile, knowledge-driven agents that can reason about

complex data. However, directly applying LLMs to raw IoT streams is impractical due to their high volume, cost, and the models' finite context windows. The challenge lies in designing a system that can summarize and structure sensor data to leverage the reasoning capabilities of LLMs without incurring prohibitive costs.

9 Conclusion

This work explores the challenge of transforming vast streams of IoT sensor data into actionable, human-centric intelligence. We present Flash-Fusion, an end-to-end framework that serves as a bridge between raw telematics and operational insight. Our design integrates three tiers: (1) an edge module that performs on-device statistical aggregation to reduce data transmission by 73.5%; (2) a cloud backend that uses unsupervised clustering to automatically classify driving behaviors; and (3) a query engine that grounds LLM outputs in empirical data, enabling context-aware, reliable insights. Our approach, validated through real-world deployment on a transit bus, demonstrates a 95% latency reduction and a 98% decrease in API token usage compared to naïve raw-data prompting. By offloading the burden of numerical aggregation from the LLM and providing it with concise, query-relevant summaries, Flash-Fusion makes LLM-powered analysis of IoT sensor streams both technically feasible and economically viable.

References

[1] Mahmoud Hussein Abdelkarem Abdelsamea, Mohamed Zorkany, and Neamat Abdelkader. 2016. Real time operating systems for the internet of things, vision, architecture and research directions. In *2016 World Symposium on Computer Applications & Research (WSCAR)*. IEEE, 72–77.

[2] Zouhair Elamrani Abou Elassad, Hajar Mousannif, Hassan Al Moatassime, and Aimad Karkouch. 2020. The application of machine learning techniques for driving behavior analysis: A conceptual framework and a systematic literature review. *Engineering Applications of Artificial Intelligence* 87 (2020), 103312.

[3] Harith Al-Safi, Harith Ibrahim, and Paul Steenson. 2025. Vega: LLM-Driven Intelligent Chatbot Platform for Internet of Things Control and Development. *Sensors* 25, 12 (2025), 3809.

[4] Tuo An, Yunjiao Zhou, Han Zou, and Jianfei Yang. 2024. Iot-llm: Enhancing real-world iot task reasoning with large language models. *arXiv preprint arXiv:2410.02429* (2024).

[5] Olli Apilo and Tapio Rautio. 2024. Evaluating the energy consumption savings at 450 MHz band for NB-IoT devices. In *Proceedings of the 14th International Conference on the Internet of Things*. 226–231.

[6] Joseph Azar, Abdallah Makhoul, Mahmoud Barhamgi, and Raphaël Couturier. 2019. An energy efficient IoT data compression approach for edge machine learning. *Future Generation Computer Systems* 96 (2019), 168–175.

[7] Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. 2024. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 3119–3137.

[8] Sourav Banerjee, Ayushi Agarwal, and Saloni Singla. 2025. Llms will always hallucinate, and we need to live with this. In *Intelligent Systems Conference*. Springer, 624–648.

[9] Fu Bang. 2023. Gptcache: An open-source semantic cache for llm applications enabling faster answers and cost savings. In *Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023)*. 212–218.

[10] Yejin Bang, Ziwei Ji, Alan Schelten, Anthony Hartshorn, Tara Fowler, Cheng Zhang, Nicola Cancedda, and Pascale Fung. 2025. Hallulens: Llm hallucination benchmark. *arXiv preprint arXiv:2504.17550* (2025).

[11] Soukaina Bouhsissin, Nawal Sael, and Faouzia Benabbou. 2023. Driver Behavior Classification: A Systematic Literature Review. *IEEE Access* 11 (2023), 14128–14153. doi:10.1109/ACCESS.2023.3243865

[12] Serge Thomas Mickala Bourobou and Younghwan Yoo. 2015. User activity recognition in smart homes using pattern clustering applied to temporal ANN algorithm. *Sensors* 15, 5 (2015), 11953–11971.

[13] Matthias Brantner, Daniela Florescu, David Graf, Donald Kossmann, and Tim Kraska. 2008. Building a database on S3. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. 251–264.

[14] Varinia Cabrera, Andrea Delbuggio, Hernán Cardoso, Diego Fraga, Alvaro Gómez, Martín Pedemonte, Rodolfo Ungerfeld, and Julián Oreggioni. 2023. Research platform to study sheep behavior. In *2023 IEEE Conference on AgriFood Electronics (CAFE)*. IEEE, 60–64.

[15] Gustavo André Setti Cassel, Vinicius Facco Rodrigues, Rodrigo da Rosa Righi, Marta Rosecler Bez, Andressa Cruz Nepomuceno, and Cristiano André da Costa. 2022. Serverless computing for Internet of Things: A systematic literature review. *Future Generation Computer Systems* 128 (2022), 299–316.

[16] Lan Chen, Hong Jiang, Lizhong Wang, Jun Li, Manhua Yu, Yong Shen, and Xusheng Du. 2025. Generative adversarial synthetic neighbors-based unsupervised anomaly detection. *Scientific Reports* 15, 1 (2025), 16.

[17] Yoonseo Choi, Eunhye Kim, Hyunwoo Kim, Donghyun Park, Honggu Lee, Jin Young Kim, and Juho Kim. 2025. BloomIntent: Automating Search Evaluation with LLM-Generated Fine-Grained User Intents. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*. 1–34.

[18] Djamel Djenouri, Roufaida Laidi, Youcef Djenouri, and Ilangko Balasingham. 2019. Machine learning for smart building applications: Review and taxonomy. *ACM Computing Surveys (CSUR)* 52, 2 (2019), 1–36.

[19] Andrew Eliasz. 2024. A Review of RTOS Fundamentals. *Zephyr RTOS Embedded C Programming: Using Embedded RTOS POSIX API* (2024), 19–67.

[20] Florian Endel and Harald Piringer. 2015. Data Wrangling: Making data useful again. *IFAC-PapersOnLine* 48, 1 (2015), 111–112.

[21] Mariano Falcitelli, Misal, Sandro Noto, and Paolo Pagano. 2024. Development of a Multi-Radio Device for Dry Container Monitoring and Tracking. *IoT* 5, 2 (2024), 187–211.

[22] Philip Feldman, James R Foulds, and Shimei Pan. 2023. Trapping LLM hallucinations using tagged context prompts. *arXiv preprint arXiv:2306.06085* (2023).

[23] Noelle Fields, Courtney Cronley, Kate Hyun, Stephen P Mattingly, Vivian J Miller, Ramezanpour Nargesi, Sheida Khademi, Shamsun Nahar, Jessica Williams, Erin Murphy, et al. 2019. *How can interdisciplinary teams leverage emerging technologies to respond to transportation infrastructure needs? A mixed-methods evaluation of civil engineers, urban planning, and social workers' perspectives.* Technical Report. National Institute for Transportation and Communities (NITC).

[24] Umberto Fugiglando, Emanuele Massaro, Paolo Santi, Sebastiano Milardo, Kacem Abida, Rainer Stahlmann, Florian Netter, and Carlo Ratti. 2018. Driving behavior analysis through CAN bus data in an uncontrolled environment. *IEEE Transactions on Intelligent Transportation Systems* 20, 2 (2018), 737–748.

[25] Tim Furche, George Gottlob, Leonid Libkin, Giorgio Orsi, and Norman Paton. 2016. Data wrangling for big data: Challenges and opportunities. In *Advances in Database Technology—EDBT 2016: Proceedings of the 19th International Conference on Extending Database Technology*. 473–478.

[26] D Ganga and V Ramachandran. 2018. IoT-based vibration analytics of electrical machines. *IEEE Internet of Things Journal* 5, 6 (2018), 4538–4549.

[27] Raymond Ghandour, Albert Jose Potams, Ilyes Boulkaibet, Bilel Neji, and Zaher Al Barakeh. 2021. Driver behavior classification system analysis using machine learning methods. *Applied Sciences* 11, 22 (2021), 10562.

[28] Panagiotis Gkonis, Anastasios Giannopoulos, Panagiotis Trakadas, Xavi Masip-Bruin, and Francesco D'Andria. 2023. A survey on IoT-edge-cloud continuum systems: Status, challenges, use cases, and open issues. *Future Internet* 15, 12 (2023), 383.

[29] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783* (2024).

[30] Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2024. Token-budget-aware llm reasoning. *arXiv preprint arXiv:2412.18547* (2024).

[31] Mona Bakri Hassan, Elmustafa Sayed Ali, Rania A Mokhtar, Rashid A Saeed, and Bharat S Chaudhari. 2020. NB-IoT: Concepts, applications, and deployment challenges. In *LPWAN Technologies for IoT and M2M Applications*. Elsevier, 119–144.

[32] Wenbo He, Xue Liu, Hoang Viet Nguyen, Klara Nahrstedt, and Tarek Abdelzaher. 2011. PDA: privacy-preserving data aggregation for information collection. *ACM Transactions on Sensor Networks (TOSN)* 8, 1 (2011), 1–22.

[33] Essam H Houssein, Mahmoud A Othman, Waleed M Mohamed, and Mina Younan. 2024. Internet of things in smart cities: Comprehensive review, open issues and challenges. *IEEE Internet of Things Journal* (2024).

[34] James W Cain III, Paul R Krausman, Brian D Jansen, and John R Morgart. 2005. Influence of topography and GPS fix interval on GPS collar performance. *Wildlife Society Bulletin* 33, 3 (2005), 926–934.

[35] Sergio Ilarri, Thierry Delot, and Raquel Trillo-Lado. 2015. A data management perspective on vehicular networks. *IEEE Communications Surveys & Tutorials* 17, 4 (2015), 2420–2460.

[36] Samsara Inc. 2025. Samsara Announces New Safety and AI-Powered Technology for Physical Operations.

[37] Reinier J Jansen and Simone Wesseling. 2018. Harsh braking by truck drivers: A comparison of thresholds and driving contexts using naturalistic driving data. In *Proceedings of the 6th Humanist Conference, The Hague, The Netherlands*. 13–14.

[38] Daniel Kang, John Guibas, Peter Bailis, Tatsunori Hashimoto, and Matei Zaharia. 2022. Semantic Indexes for Machine Learning-based Queries over Unstructured Data. In *SIGMOD*.

[39] Malek Karaim, Mohamed Elsheikh, and Aboelmagd Noureldin. 2018. GNSS error sources. In *Multifunctional operation and application of GPS*. IntechOpen.

[40] Kamran Khan, Saif Ur Rehman, Kamran Aziz, Simon Fong, and Sababady Sarasvady. 2014. DBSCAN: Past, present and future. In *The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014)*. IEEE, 232–238.

[41] İbrahim Kök, Orhan Demirci, and Suat Özdemir. 2024. When iot meet llms: Applications and challenges. In *2024 IEEE International Conference on Big Data (BigData)*. IEEE, 7075–7084.

[42] Amol Kulkarni. 2023. Amazon athena: Serverless architecture and troubleshooting. *International Journal of Computer Trends and Technology* 71, 5 (2023), 57–61.

[43] Mads Lauridsen, István Z Kovács, Preben Mogensen, Mads Sorensen, and Steffen Holst. 2016. Coverage and capacity analysis of LTE-M and NB-IoT in a rural area. In *2016 IEEE 84th Vehicular Technology Conference (VTC-Fall)*. IEEE, 1–5.

[44] Fei Li and Hosagrahar V Jagadish. 2014. NaLIR: an interactive natural language interface for querying relational databases. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 709–712.

[45] Kuan Li, Liwen Zhang, Yong Jiang, Pengjun Xie, Fei Huang, Shuai Wang, and Minhao Cheng. 2025. LaRA: Benchmarking Retrieval-Augmented Generation and Long-Context LLMs–No Silver Bullet for LC or RAG Routing. *arXiv preprint arXiv:2502.09977* (2025).

[46] Jingbin Liu, Yifan Liang, Dong Xu, Xiaodong Gong, and Juha Hyyppä. 2023. A ubiquitous positioning solution of integrating GNSS with LiDAR odometry and 3D map for autonomous driving in urban environments. *Journal of Geodesy* 97, 4 (2023), 39.

[47] Jiaheng Liu, Dawei Zhu, Zhiqi Bai, Yancheng He, Huanxuan Liao, Haoran Que, Zekun Wang, Chenchen Zhang, Ge Zhang, Jiebin Zhang, et al. 2025. A comprehensive survey on long context language modeling. *arXiv preprint arXiv:2503.17407* (2025).

[48] Xing-Gang Luo, Hong-Bo Zhang, Zhong-Liang Zhang, Yang Yu, and Ke Li. 2019. A new framework of intelligent public transportation system based on the internet of things. *IEEE Access* 7 (2019), 55290–55304.

[49] Praveen Mohanram, Alice Passarella, Elena Zattoni, Roberto Padovani, Niels König, and Robert H Schmitt. 2022. 5G-Based Multi-Sensor Platform for Monitoring of Workpieces and Machines: Prototype Hardware Design and Firmware. *Electronics* 11, 10 (2022), 1619.

[50] Emmanuel Ogundare. 2024. Understanding the mediating role of artificial intelligence in urban transportation planning for smart city development and its implications for the United States. *International Journal of Innovative Science and Research Technology* 9 (2024), 10–5281.

[51] Maria Leonor da Anunciação Moreira Paiva et al. 2023. *Nordic Semiconductor-Remaining a Market Leader by Overcoming Macroeconomic and Industry Specific Risks*. Master's thesis. Universidade NOVA de Lisboa (Portugal).

[52] Max Peeperkorn, Tom Kouwenhoven, Dan Brown, and Anna Jordanous. 2024. Is temperature the creativity parameter of large language models? *arXiv preprint arXiv:2405.00492* (2024).

[53] Charith Perera, Chi Harold Liu, Srimal Jayawardena, and Min Chen. 2014. A survey on internet of things from industrial market perspective. *IEEE Access* 2 (2014), 1660–1679.

[54] Gil Press. 2016. *Cleaning Big Data: Most Time-Consuming, Least Enjoyable Data Science Task, Survey Says*. Forbes, updated April 14, 2022.

[55] Stefano Rando, Luca Romani, Alessio Sampieri, Luca Franco, John Yang, Yuta Kyuragi, Fabio Galasso, and Tatsunori Hashimoto. 2025. LongCodeBench: Evaluating Coding LLMs at 1M Context Windows. *arXiv preprint arXiv:2505.07897* (2025).

[56] Partha Pratim Ray. 2018. A survey on Internet of Things architectures. *Journal of King Saud University-Computer and Information Sciences* 30, 3 (2018), 291–319.

[57] Jonathan Ross. 2025. *Groq On-demand Pricing for Tokens-as-a-Service*. https://groq.com/pricing

[58] Khaled Saleh, Mohammed Hossny, and Saeid Nahavandi. 2017. Driving behavior classification based on sensor data fusion using LSTM recurrent neural networks. In *2017 IEEE 20th international conference on intelligent transportation systems (ITSC)*. IEEE, 1–6.

[59] Mohammad Shahverdy, Mahmood Fathy, Reza Berangi, and Mohammad Sabokrou. 2020. Driver behavior detection and classification using deep convolutional neural networks. *Expert Systems with Applications* 149 (2020), 113240.

[60] Ashish Kumar Sultania, Pouria Zand, Chris Blondia, and Jeroen Famaey. 2018. Energy Modeling and Evaluation of NB-IoT with PSM and eDRX. In *2018 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 1–7.

[61] Che-Ping Tsai, Ganyu Teng, Phillip Wallis, and Wei Ding. 2024. AnoLLM: Large Language Models for Tabular Anomaly Detection. In *Proceedings of the Thirteenth International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=7VkHffT5X2 To appear.

[62] Ruben Usamentiaga, Miguel Angel Fernandez, Alberto Fernandez Villan, and Juan Luis Carus. 2018. Temperature monitoring for electrical substations using infrared thermography: architecture for Industrial Internet of Things. *IEEE transactions on industrial informatics* 14, 12 (2018), 5667–5677.

[63] Marc Vila, Maria-Ribera Sancho, Ernest Teniente, and Xavier Vilajosana. 2023. Critical infrastructure awareness based on IoT context data. *Internet of Things* 23 (2023), 100855.

[64] Yanan Xin, Natasa Tagasovska, Fernando Perez-Cruz, and Martin Raubal. 2022. Vision paper: causal inference for interpretable and robust machine learning in mobility analysis. In *Proceedings of the 30th International Conference on Advances in Geographic Information Systems*. 1–4.

[65] Fengli Xu, Qianyue Hao, Zefang Zong, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, Jiahui Gong, Tianjian Ouyang, Fanjin Meng, et al. 2025. Towards large reasoning models: A survey of reinforced reasoning with large language models. *arXiv preprint arXiv:2501.09686* (2025).

[66] Huatao Xu, Liying Han, Qirui Yang, Mo Li, and Mani Srivastava. 2024. Penetrative ai: Making llms comprehend the physical world. In *Proceedings of the 25th International Workshop on Mobile Computing Systems and Applications*. 1–7.

[67] Hongzuo Xu, Yijie Wang, Songlei Jian, Qing Liao, Yongjun Wang, and Guansong Pang. 2024. Calibrated one-class classification for unsupervised time series anomaly detection. *IEEE Transactions on Knowledge and Data Engineering* 36, 11 (2024), 5723–5736.

[68] Joseph Yiu. 2020. *Definitive Guide to Arm Cortex-M23 and Cortex-M33 Processors*. Newnes.

[69] Wei Yu, Fan Liang, Xiaofei He, William Grant Hatcher, Chao Lu, Jie Lin, and Xinyu Yang. 2017. A survey on the edge computing for the Internet of Things. *IEEE access* 6 (2017), 6900–6919.

[70] Li Zhu, Fei Richard Yu, Yige Wang, Bin Ning, and Tao Tang. 2018. Big data analytics in intelligent transportation systems: A survey. *IEEE transactions on intelligent transportation systems* 20, 1 (2018), 383–398.

[71] Yousaf Bin Zikria, Rashid Ali, Muhammad Khalil Afzal, and Sung Won Kim. 2021. Next-generation internet of things (iot): Opportunities, challenges, and solutions. *Sensors* 21, 4 (2021), 1174.

[72] Yuval Ziv, Barouch Matzliach, and Irad Ben-Gal. 2025. Sensor Fusion for Target Detection Using LLM-Based Transfer Learning Approach. *Entropy* 27, 9 (2025), 928.

[73] Mingyu Zong, Arvin Hekmati, Michael Guastalla, Yiyi Li, and Bhaskar Krishnamachari. 2025. Integrating large language models with internet of things: applications. *Discover Internet of Things* 5, 1 (2025), 2.