Leveraging Small Language Models for Real-Time Decision Support in Climate-Induced Urban Mobility Disruptions

Venkatesan Murugan*, Disha Daniel†, A Harsha Kumar*, Sahaya Beni Prathiba‡*, R. Dhanalakshmi‡
*School of Computer Science and Engineering, Vellore Institute of Technology, Chennai 600127, India
†School of Electronics Engineering, Vellore Institute of Technology, Chennai, India
‡Centre for Cyber Physical Systems, School of Computer Science and Engineering,
Vellore Institute of Technology, Chennai 600127, India
Email: venkatesan.m2022@vitstudent.ac.in, dishadaniel24@gmail.com, ahkharsha@gmail.com,
prathiba.sbb@vit.ac.in, dhanalakshmi.r@vit.ac.in
*Corresponding author

***Abstract*—The increase in climate change brings major challenges to urban transportation systems, with extreme weather events showing the fragility of existing traffic management models. While Large Language Models (LLMs) offer sophisticated reasoning, their high computational and latency overhead makes them unsuitable for real-time, on-device decision support in offline or resource-constrained environments. This paper addresses the critical gap and proposes a domain-specific Small Language Model (SLM) with optimization on edge deployment. A key contribution is a synthetic data generation pipeline, which uses a complex multi-variate simulation model, used to create the Climate-Mobility Narrative Corpus (CMNC), a domain-specific dataset of over 1 million context-rich entries describing realistic climate-induced disruptions. The ClimateMobility-SLM (CM-SLM), a 270M-parameter Gemma model, is trained from scratch on this corpus. The framework is also tested using a set of downstream algorithms for alert generation and routing, and its performance is rigorously benchmarked. Extensive quantitative analysis and ablation studies show that the CM-SLM, when quantized, achieves a mean validation perplexity of 2.84 $\pm$ 0.05 and near-real-time inference speeds, significantly outperforming larger, general-purpose baseline models in both domain-specific accuracy and computational efficiency. This research demonstrates that purpose-built SLMs are a powerful tool to use for urban resilience, turning raw anomaly data into useful, human-readable Decision Support information for emergency response.**

***Index Terms*—Small Language Models (SLMs), Edge AI, Traffic Management, Urban Mobility, Synthetic Data Generation, Real-Time Decision Support, Natural Disaster, Deep Learning.**

I. Introduction

As climate change intensifies, urban centers face mounting pressure on their public infrastructure's resilience. Extreme weather events like flash and heat wave, snowstorms, etc. are becoming common and increasingly severe, placing a heavy burden on the transport infrastructure in a country of origin [1]–[4]. These climate induced disruptions cause a chain of failures, moving beyond simple inconvenience to cause widespread economic paralysis, compromise public safety, and severely limit emergency response and evacuation planning [2], [5]. The roads and public transport systems that keep a modern city running are very vulnerable to these black swan weather events.

The management of traffic has reacted by implementing various data-driven forecasting models that have adopted deep learning methodologies, including Long Short-Term Memory(LSTM) networks [6]–[9]. While effective at predicting traffic flow under normal, historical patterns, these models do not perform well when faced with the non-linear dynamics and unprecedented nature of climate-driven disruptions. They are good at numerical prediction but cannot perform semantic reasoning. A significant gap remains: the last-mile information gap. A traffic operator may see that a road's flow rate has dropped to zero, but traditional models cannot explain why (e.g., This is a flash flood) or suggest what to do (e.g., Advise dispatching high-clearance vehicles).

The use of Large Language Models (LLMs) has also become one of the helpful tools, as it is capable of both zero-shot reasoning and natural language generation. In theory, an LLM could ingest raw data and create the detailed, human-readable alerts that operators need. However, their practical application in this domain is very limited and not explored much. The immense size of models like GPT-4 or Llama 3 makes them impractical for real-time deployment in edge or resource-constrained environments, such as in-vehicle systems or local traffic control hubs [10]. High latency, coupled with a dependency on massive cloud infrastructure and significant operational costs, renders them unsuitable for systems that demand immediate, offline responses. Furthermore, as generalist models, they often lack the specialized, domain-specific knowledge needed to produce accurate and useful insights for an urban transportation crisis [11], [12].

This paper suggests that a Small Language Model (SLM), when purpose-built through domain-specific pretraining and optimized for on-device operation, can bridge this capability-efficiency gap. SLMs offer a balance between the reasoning ability of LLMs and the computational efficiency required

for real-time, offline applications, bringing the best of both worlds [13]–[15]. Developing such a powerful, specialized tool requires a high quality, domain-centric dataset that captures the complex relationships between quantitative traffic data and the qualitative, contextual nuances of climate events. Since no such public corpus exists, creating one was necessary.

This paper details a framework for the development, training, and deployment of a domain-specific SLM aimed at climate-mobility decision support. The primary contributions are:

1) **A New Synthetic Data Generation Pipeline:** This work introduces a method for synthetically generating a large-scale, domain-specific corpus. This pipeline uses a complex, multi-variate simulation model to generate realistic anomaly data, that is then turned into over 1 million unique, context-rich narratives. The resulting **Climate-Mobility Narrative Corpus (CMNC)** is a key contribution.
2) **Development of the ClimateMobility-SLM (CM-SLM):** The paper details the training of a 270M-parameter Gemma-based model from scratch on the CMNC. This includes a full architectural breakdown and an optimization workflow (e.g., 8-bit quantization) to get the model ready for high-performance inference on edge devices.
3) **A Set of Downstream Decision-Support Algorithms:** A set of four algorithms is suggested to show how this can be implemented in real life. These algorithms show how the SLM's narrative output can be programmatically converted into structured JSON alerts, assigned severity levels, and used to generate adaptive rerouting recommendations.
4) **Rigorous Empirical Evaluation:** The CM-SLM was rigorously benchmarked against several baseline SLMs (e.g., Qwen, SmolLM2) across a spectrum of metrics, including language perplexity, BLEU scores, inference speed, and memory footprint. This evaluation includes extensive ablation studies on the impact of corpus size and model quantization, confirming the framework's effectiveness.

The subsequent sections are as follows: Section II offers a review of related works of current traffic modelling, other techniques that are used in this domain, and edge computing. Section III details the high-level, end-to-end framework of the proposed system. Section IV details the technical core, covering the corpus simulation model, SLM architecture, and downstream algorithms. Section V is about the experimental design, including datasets, baselines, and evaluation metrics. Section VI presents and discusses the quantitative, qualitative, and ablation study results. Lastly, the paper ends with the conclusion provided in Section VII where the research findings in this area are summarised, the limitation is identified, and the future research ideas are proposed.

II. Related Works

The concepts incorporated in this paper relate to several of the emerging areas rapidly: urban transportation, generative artificial intelligence, and edge computing. The detailed review provides a background of the suggested system, identifies the limitations that are present, and gives the research gap that the given work is intended to fill.

A. Traditional Traffic and Mobility Forecasting

Intelligent transportation systems (ITS) have always been based on numerical traffic forecasting. Initial ways of predicting the traffic on a particular route of an artery were based on statistical time-series models, including AutoRegressive Integrated Moving Average (ARIMA). However, their limitations in capturing complex, non-linear spatial and temporal dependencies led to the use of machine learning and, followed by deep learning Techniques [6], [7].

Based on the Recurrent Neural Networks (RNNs) particularly Long Short-Term Memory(LSTM) and Gated Recurrent Unit (GRU) networks, nowadays it is quite a common approach to predict traffic in short-term [6], [7]. These architectures are excellent at learning historical patterns from vast datasets, effectively modeling the day-to-day traffic flow of urban traffic. Along with the new developments in graph neural networks and spatio-temporal models, the prediction capabilities have also been enhanced recently [16]–[20]. While accurate under normal conditions, these models share two key weaknesses. First, they are black boxes that provide numerical output without semantic context. Second, their forecasting capability is reduced considerably in the cases of high-impact events, including the disasters caused by climate. They can predict a traffic jam, but cannot reason about its cause or implications.

B. Generative AI for Transportation Systems

The arrival of generative AI, particularly LLMs, marks a major shift from numerical prediction to semantic reasoning. Early explorations have begun to use LLMs as high-level decision-making brains for autonomous systems or as sophisticated world models for traffic simulation [21], [22]. These models are able to comprehend unstructured data (such as weather reports) and relate with structured data (such as traffic flow) and produce Decision Support.

However, as highlighted in Section I, the direct application of foundation-scale LLMs in real-time mobility systems is often not practical [10], [11]. There is a trade-off between going with large models for advanced reasoning, which comes with the cost of the computational resources that is required. As summarized in Table I, generative models introduce the crucial element of semantic understanding, but at a computational cost that traditional models do not have. This work suggests that an SLM can optimize this trade-off.

C. The Rise of Small Language Models (SLMs)

In response to the computational requirements of LLMs, the research community has focused on developing right-sized

TABLE I
A COMPARISON OF VARIOUS TECHNIQUES THAT ARE USED IN MOBILITY MODELLING.

Feature	Statistical Models (e.g., ARIMA)	Deep Learning Models (e.g., LSTM [6])	Generative Models (e.g., LLM/SLM [11], [12])
Primary Task	Prediction	Prediction	Reasoning
Semantic Understanding	None	None	High
Handling Novel Events	Poor	Poor to Fair	Good (via reasoning)
Interpretability	High	Low (Black Box)	High (Natural Language)
Computational Cost	Low	Medium	Very High

TABLE II
PERFORMANCE AND DEPLOYMENT CHARACTERISTICS OF LLMS VS. SLMS FOR EDGE APPLICATIONS.

Metric	Large Language Models (LLMs) [11]	Small Language Models (SLMs) [13]
Parameter Size	50B - 1T+	1B - 10B
Primary Deployment	Cloud / Data Center	Edge / On-Device
Inference Latency	High (seconds per response)	Low (ms per token)
Cost per Query	High (API-based)	Negligible (local)
Offline Capability	No	Yes
Domain Specialization	Generalist	Requires specific training
Key Weakness	Cost, Latency, Size	Lacks broad knowledge

Small Language Models (SLMs). These models, typically with fewer than 10 billion parameters (e.g., Gemma, Phi-3, Llama 3 8B), are specifically designed to offer a more efficient balance of performance and size. They are smaller hence are an ideal size of model capable of running on-device applications and edge applications, as well as achieve low-latency, low-cost, and offline-capable inference.

However, SLMs are not a complete solution. On their own, they often lack the deep, domain-specific knowledge of their larger counterparts. A general-purpose SLM, when prompted with highly technical traffic or climate jargon, will probably fail to produce a useful, accurate response. This requires domain-specific fine-tuning or, as this paper explores, complete domain-specific pretraining to give the model the needed specialized knowledge. The distinction between the LLM and SLM design philosophies is shown in Table II.

D. Edge AI: Inference in Resource-Constrained Environments

The hypothesis of running an SLM-on-the-Edge (Section I) is only possible because of recent advances in model optimization techniques. The techniques are significant in the adaption of models to execute on constrained resources of a handle unit of compute and memory, like an in-vehicle infotainment system or a roadside traffic control unit.

- **Quantization:** This involves reducing the numerical precision of a model's weights and activations. Instead of 32-bit floating-point (FP32) numbers, models are converted to use 8-bit integers (INT8) or even 4-bit formats (e.g., NF4) [23], [24]. This reduces the memory footprint (by up to 4x-8x) and speeds up computation.
- **Pruning:** This procedure includes the detection and elimination of redundant or non-important weights of the neural network, forming a smaller, sparse model that can often be trained nearly as quickly as the original, with little or no loss of accuracy [23].
- **Knowledge Distillation:** This approach uses a large, high-performing teacher model (like GPT-4) to train a smaller student model (like the CM-SLM). The student model learns from the teacher's output distribution, effectively transferring knowledge into a more compact form [25], [26].

The presented framework is based on the extensive use of quantization to render the CM-SLM to be practically applicable to inference in real-time. The deployment of these optimised SLMs on edge devices has also been proved viable as observed by other recent studies in this area of research [14], [15].

E. Synthetic Data Generation for Domain-Specific AI

The main barrier to developing a specialized CM-SLM is the lack of a suitable training corpus. Publicly available datasets for mobility are extensive but fall into the wrong categories. As shown in the analysis in Table III, a critical gap exists. Datasets like UCTB [27] are purely quantitative time-series data, lacking any semantic or narrative component. Conversely, autonomous vehicle datasets (e.g., nuScenes) are vision-centric.

This cold start issue frequently arises in specialized artificial intelligence. A common solution is the use of synthetic data generation. Often, this involves using a large teacher LLM such as GPT-4 to generate thousands of training examples, a technique that has been used and proven to be effective in various domains [28], [29]. However, this approach can be costly and risks contaminating the student model with the teacher's natural biases. The alternative looked at in this paper is the use of a programmatic, simulation-based pipeline to

TABLE III
ANALYSIS OF EXISTING DATASETS FOR TRAINING MOBILITY-FOCUSED LANGUAGE MODELS.

Dataset	Data Type	Key Limitation for this Work
UCTB [27]	Quantitative Time-Series	Lacks all semantic/narrative context.
nuScenes / Waymo	Lidar, Camera, Radar	Vision-centric; not text-based.
General Text (e.g., C4)	Web Scrape	Lacks domain jargon; uncurated.
CMNC (Proposed)	Synthetic Narratives	Purpose-built for semantic reasoning.

generate a corpus. This method offers good control over data diversity, structure, and factual grounding, making sure the final dataset is well suited for the final task. Real-time decision support systems must integrate human-machine interfaces to ensure operator trust [30]. Comprehensive reviews of synthetic data methodologies [28], [29] highlight best practices for data quality and domain specificity.

F. Research Gap and Problem Formulation

The literature review shows a clear research gap with several parts.

1) Traditional mobility models (LSTMs) can predict numbers but cannot *reason*.
2) Large Language Models (LLMs) have good reasoning but are too *large and slow* for real-time, on-device deployment.
3) Small Language Models (SLMs) are the right *size* but lack the specialized, domain-specific *data* to be effective in this niche.
4) While techniques for *synthetic data* and *model optimization* exist, they have not been combined into a single, working system to solve this specific climate-mobility challenge.

Therefore, a clear need exists for a complete system that includes: (1) a novel, large-scale, domain-specific narrative corpus for climate-mobility events, and (2) a specialized SLM trained on this corpus, optimized via techniques like quantization, to provide real-time, on-device semantic reasoning. This paper directly addresses this complex gap.

III. SYSTEM ARCHITECTURE

The system is designed to provide clear, useful information to an operator during a crisis, as illustrated in the use-case scenario in Figure 1. To address the research gap identified in Section II-F, a four-layer system is proposed. This architecture manages the full data lifecycle, from the first simulation to final on-device inference. The system is divided into logical components, each handling a different task: data intake, offline corpus generation, model training, or real-time deployment. The complete data flow and interaction between these layers are shown in Figure 2.

A. High-Level Framework Overview

The proposed system works in two distinct phases: an offline development phase (Layers 2 and 3) and a real-time deployment phase (Layers 1 and 4).

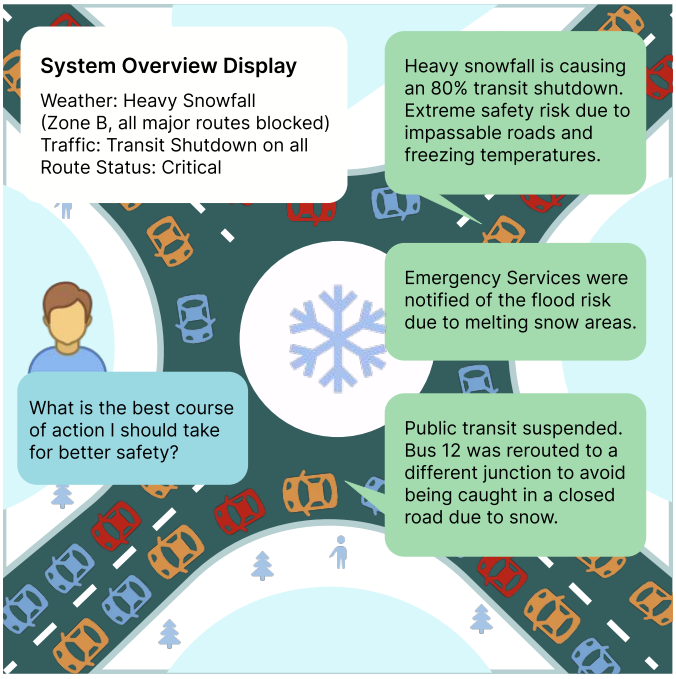

Fig. 1. A use-case scenario illustrating the proposed system's goal: providing real-time, natural language interaction and actionable insights on an edge device during a climate-induced mobility disruption.

In the offline phase, raw data is collected and used as the basis for the simulation. This simulation (Layer 2) generates the 1 million+ entry CMNC, which is then used to train the CM-SLM from scratch and optimize it via quantization (Layer 3).

In the real-time phase, the lightweight, quantized SLM is deployed onto an edge device (Layer 4), which receives live data (Layer 1) from its environment, processes it using the SLM, and generates actionable, human readable alerts to support immediate decision making.

B. Layer 1: Real-Time Data Ingestion

This layer represents the real-time data sources available to the deployed edge device. It is not used for training but provides the live inputs for inference. These inputs are assumed to be simple, structured data points, such as:

- **Weather Data:** Sourced from a local sensor or a weather API (e.g., '"temp": -2.5, "event": "snow", "rate_mmhr": 10').

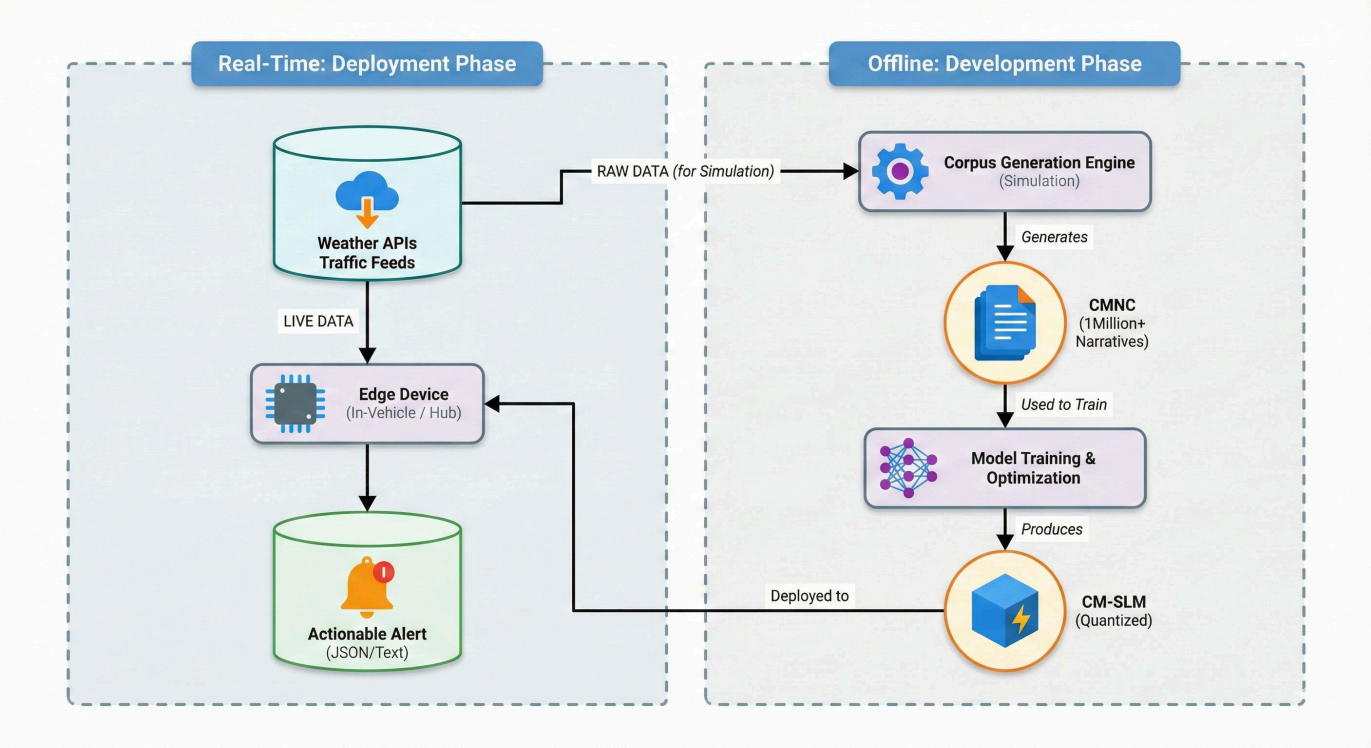

Fig. 2. High-level system architecture, demonstrating the distinct offline (development) and real-time (deployment) phases. The Corpus Generation Engine and Model Training are computationally heavy, one-time tasks. The resulting quantized SLM is lightweight and designed for efficient inference on an edge device, which ingests live data to produce alerts.

- **Traffic Data:** Sourced from local road sensors, GPS, or a municipal API (e.g., '"road_id": "I-95-NB", "avg_speed_kph": 15, "status": "anomaly"').

This raw data provides the prompt that is fed into the deployed SLM for analysis.

C. Layer 2: Offline Corpus Generation Engine

This is the core of the data generation pipeline and is explained further in Section IV. This layer is responsible for creating the new CMNC. It takes foundational real-world data (e.g., historical traffic flow from the UCTB [22], climate records) and uses them as the statistical basis for a complex simulation. This engine programmatically generates realistic, time-series anomaly data and then connects this quantitative data to qualitative, human-readable narratives. The output is a text-based, instruction-style dataset with over 1 million entries formatted specifically for training the CM-SLM.

D. Layer 3: Model Training and Optimization

This layer performs the computationally heavy task of model creation. The 1M+ entry CMNC from Layer 2 is used as the sole training dataset. The 270-million parameter Gemma-based model is trained from scratch on this corpus. This domain-specific pretraining makes sure the model learns the specific vocabulary, context, and cause-and-effect relationships of the climate-mobility domain.

After training, the optimization step is performed. The model's weights, which is stored in 32-bit floating-point (FP32) precision, undergo 8-bit quantization (INT8). This process, discussed in Section II-D, reduces the model's memory footprint by nearly 4x and greatly speeds up its inference speed on compatible edge hardware. The final output of this layer is the lightweight, optimized CM-SLM, ready for deployment.

E. Layer 4: On-Device Deployment and Inference

This final layer represents the end user of the framework. The optimized CM-SLM from Layer 3 is loaded onto a resource-constrained edge device (e.g., an in-vehicle computer or a traffic management hub). This device operates in real-time, receiving live data feeds from Layer 1.

When a new piece of data arrives (e.g., an anomaly trigger), it is formatted into a prompt and fed to the local CM-SLM. The SLM performs inference a process that takes milliseconds and generates a human-readable text output.This output text summarises the circumstance, evaluates the probable impact, and gives the decision support. This output is then fed into the downstream algorithms (detailed in Section IV) to create structured alerts for the human operator or an automated system.

IV. Methodology

The framework's success depends on two core components: a very specialized training corpus and an efficient model architecture. This section explains the new simulation and data generation pipeline, the architecture of the ClimateMobility-Small Language Model (CM-SLM), and the set of downstream

algorithms that convert the model's output into practical, real-world actions.

A. The Climate-Mobility Narrative Corpus (CMNC)

Training a robust, domain-specific language model requires a large-scale, high-quality corpus. As our analysis established (Table III), no such public corpus exists for the climate-mobility domain. So, the first major contribution of this work is creating the **Climate-Mobility Narrative Corpus (CMNC)**, a synthetic dataset of over 1 million text-based entries. The generation of this corpus was a multi-stage process.

1) Data Sourcing and Preprocessing: The simulation was not built in isolation. To base the synthetic data on reality, foundational statistics were taken from real-world public datasets. This included historical traffic flow patterns from the UCTB repository [22] and historical weather data from NOAA repositories. This data was not used directly but was analyzed to extract key parameters (e.g., mean/variance of rush hour traffic, typical duration of a snow event) which were then used to set the parameters for the simulation engine.

2) Quantitative Anomaly Modeling: A random noise generator is not enough to model the complex dynamics of the real-world traffic disruption. To create a difficult and realistic simulation, the disruption score $D(t)$ for a given road segment at time t was modelled as a combination of multiple components. This approach, inspired by complex signal modelling in other domains, is described in Equation 1.

$$D(t) = \mathcal{B}(t) + \mathcal{E}(c, d_t) + \mathcal{N}_{structured}(t) \tag{1}$$

Where each component is defined as:

- $\boldsymbol{\mathcal{B}(t)}$ **(Baseline Traffic):** This component models the predictable 24-hour diurnal cycle of urban traffic. This component is represented as a sum of sinusoidal functions that mirror the natural flow of morning or evening rush hours.
- $\boldsymbol{\mathcal{E}(c, d_t)}$ **(Event Impact):** This is the core non-linear event function. It models the impact of a specific climate event c (e.g., flood, snow) over its duration d_t. it is modelled using a logistic function to simulate the gradual onset, peak disruption, and slower recovery phase of a major event.
- $\boldsymbol{\mathcal{N}_{structured}(t)}$ **(Structured Noise):** This component injects realistic, short-term volatility. It is a mix of several noise types, including periodic high-magnitude spikes (simulating random accidents), baseline sensor noise, and a drifting component (simulating sensor degradation).

By systematically varying the parameters of Equation 1, the engine generated the large time-series dataset. Each event consisted of the quantitative disruption score $D(t)$ and the set of causal parameters $\{c, d_t\}$ that produced it.

3) The Narrative Synthesis Engine: The next step was to turn this quantitative time-series data into a qualitative, text-based training corpus. This was achieved using the Narrative Synthesis Engine, a programmatic pipeline shown in Algorithm 1.

Algorithm 1 The Narrative Synthesis Engine

Require: Simulated event $\mathcal{S} = \{D(t), c, d_t, \text{location}\}$
Require: Template Database $\mathcal{T} = \{\mathcal{T}_{snow}, \mathcal{T}_{flood}, \dots\}$
Require: Detail Database $\mathcal{D} = \{\text{streets, vehicles, } \dots\}$
Ensure: A unique narrative entry $(prompt, response)$

// 1. Select Template based on Event
$event\_template \leftarrow$
  SelectTemplate$(\mathcal{T}, \mathcal{S}.c)$
// 2. Extract Key Quantitative Data
$severity \leftarrow$ CalculateSeverity$(D(t))$
$duration\_text \leftarrow$ FormatDuration$(\mathcal{S}.d_t)$
// 3. Create the Prompt (Input for SLM)
$prompt \leftarrow$ GeneratePrompt$(\mathcal{S}, severity)$
// e.g., "EVENT: snow, LOCATION: I-95, SEVERITY: critical"
// 4. Populate Narrative (Output for SLM)
$narrative \leftarrow event\_template$
$narrative \leftarrow$
  ReplaceToken$(narrative, \text{"[SEVERITY]"}, severity)$
$narrative \leftarrow$
  ReplaceToken$(narrative, \text{"[LOCATION]"}, \mathcal{S}.location)$
$narrative \leftarrow$
  ReplaceToken$(narrative, \text{"[DURATION]"}, duration_text)$
// 5. Add Unique Details to Prevent Overfitting
$street \leftarrow$ RandomSample$(\mathcal{D}$.streets$)$
$vehicle \leftarrow$ RandomSample$(\mathcal{D}$.vehicles$)$
$narrative \leftarrow$ AddDetail$(narrative, street, vehicle)$
// e.g., "I-95 is blocked near the Main St exit. A snowplow"
return $(prompt, narrative)$

This engine uses an advanced templating system with over 25 unique narrative structures (e.g., News Bulletin, Incident Report, Public Safety Alert). It maps the quantitative data from the simulation into these templates. For example, a high $D(t)$ value combined with the event parameter $c =$ "snow" would be matched to a Snowstorm Alert template. The engine then fills the template with precise, randomised information to generate the required human-readable narrative.

4) Corpus Validation and Quality Assurance: A synthetic corpus can be erroneous or not diverse. This was avoided by using a two-step process of validation. To check on the length of tokens, eliminate duplicates, and also check grammatical correctness, automated checks were first done on the corpus of 1 million+ entries. Second, the logical consistency and the naturalness of the generated narratives were manually checked through a random sample of 1,000 entries, which were checked by human reviewers. This feedback loop was used to improve the templates in Algorithm 1 before the final corpus was generated.

B. The ClimateMobility-SLM (CM-SLM)

The second core component is the Small Language Model (SLM) itself. from the base architecture to the attention mechanisms, were made to balance reasoning capabilities with the strict efficiency limits of edge deployment.

1) Base Architecture: The CM-SLM is a 270-million parameter decoder only transformer model, based on the Gemma 270M architecture [13], as illustrated in Figure 3. This size was

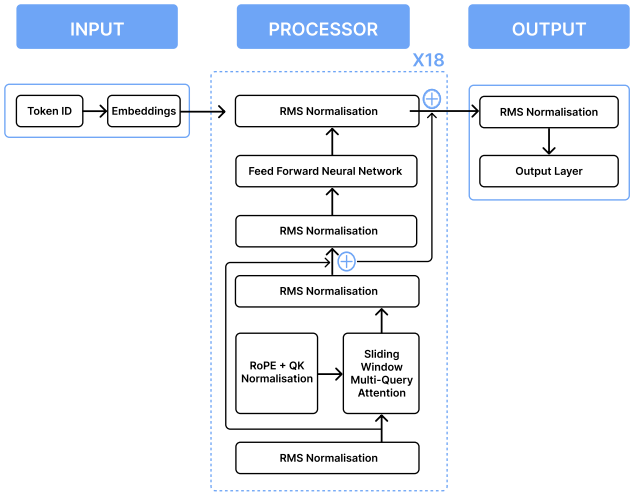

Fig. 3. Architecture diagram of the CM-SLM, illustrating the complete data flow. The input embeddings, 18 transformer blocks, and final output layer are all shown. Each block is equipped with RMS Normalisation, Sliding Window Attention, and a Feed Forward Network.

specifically chosen as it represents a good trade-off between model expressiveness and the memory footprint, making it a practical candidate for on-device inference. The model uses 18 transformer blocks (or layers) and an embedding dimension optimized for this scale. It was trained from scratch exclusively on the CMNC, allowing its limited parameter budget to be fully dedicated to learning the climate-mobility domain.

2) Token Embeddings: It starts by partitioning the input text prompt into an array of tokens through a subword-level tokenizer, namely, Byte Pair Encoding (BPE) and a vocabulary size of 50,257. These tokens are then encoded into high-dimensional vectors to encode semantic meaning into them and are inputted into the transformer blocks.

3) RMS Normalization: To achieve better numerical stability and faster computation, the model employs Root Mean Square (RMS) Normalization in place of traditional Layer Normalization. Given an input vector $x \in \mathbb{R}^d$ and learnable parameters, the normalization is computed. The variance is first computed as the mean of squared elements, as shown in Equation 2:

$$\text{var}(x) = \frac{1}{d}\sum_{i=1}^{d} x_i^2 \tag{2}$$

The Root Mean Square factor, defined in Equation 3 with an epsilon ϵ for numerical stability (typically 10^{-8}), is:

$$\text{rms}(x) = \sqrt{\text{var}(x) + \epsilon} \tag{3}$$

Each element is then normalized as shown in Equation 4:

$$\hat{x}_i = \frac{x_i}{\text{rms}(x)} \tag{4}$$

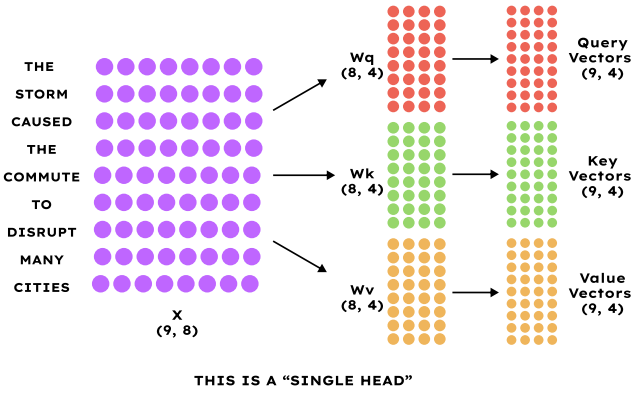

Fig. 4. Computation method for single-head attention, showing the transformation of input tokens (X) using query (Q), key (K), and value (V) projections.

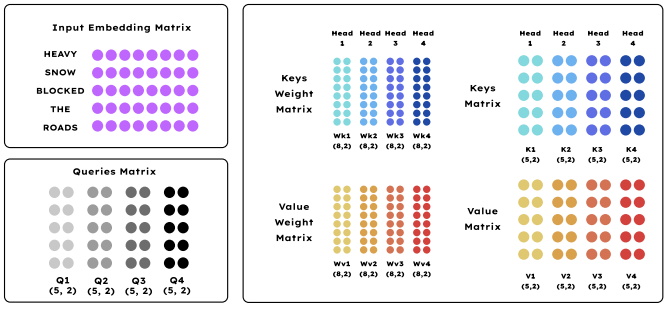

Fig. 5. The computation for Multi-Head Attention (MHA), which uses separate Key (Wk1, Wk2...) and Value (Wv1, Wv2...) weight matrices for each head.

Finally, the normalized vector is scaled using a learnable parameter scale_i, following the Gemma-style formulation in Equation 5:

$$y_i = \hat{x}_i \cdot (1 + \text{scale}_i) \tag{5}$$

This approach is computationally less expensive than standard LayerNorm as it avoids recentering the data with a mean, which leads to faster inference.

4) Multi-Query Attention (MQA): To significantly reduce the memory bandwidth needs during inference. The model is based on Multi-Query Attention (MQA) as opposed to Multi-Head Attention (MHA).

A standard attention technique maps the input embedding (X) into Query (Q), Key (K), and Value (V) vectors using separate weight matrices (Wq, Wk, Wv). This concept for a single head is shown in Figure 4.

In MHA, this is done in parallel for multiple heads, and each head has its own unique weight matrices (Wk1, Wk2... and Wv1, Wv2...), as illustrated in Figure 5. This is effective but computationally demanding.

In MQA, as shown conceptually in Figure 6, all attention heads share a single set of Key (K) and Value (V) projection matrices, while each head retains its own unique Query (Q) matrix. This greatly reduces the size of the Key-Value (KV)

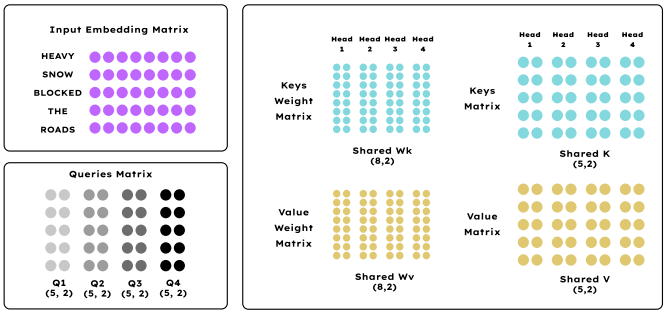

Fig. 6. Multi-Query Attention (MQA) computation. Unlike Multi-Head Attention (Fig. 5), all heads share a single Key (Shared K) and Value (Shared V) matrix, reducing memory and computational cost.

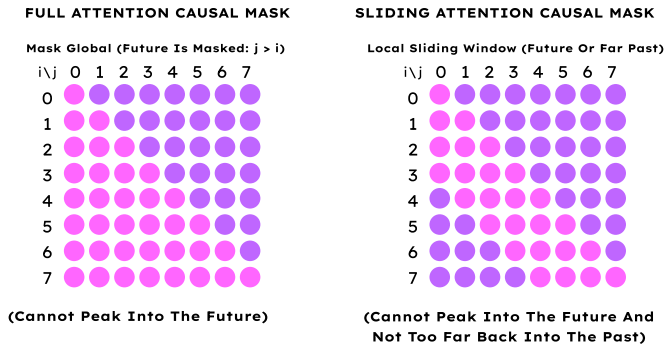

Fig. 7. A comparison between a Full Attention causal mask (left), permitting each token to attend to all predecessors, and a Sliding Window Attention mask (right), which restricts attention to the recent past.

cache, a major consumer of memory during text generation, and speeds up computation with almost no noticeable loss in model quality for this task.

5) Sliding Window Attention: The standard transformer architecture uses full attention, where every token attends to every previous token. This creates a cost in computation and memory that scales quadratically ($O(n^2)$) with the sequence length n. For a 270M model, a more efficient method is needed.

The CM-SLM uses Sliding Window Attention (SWA), as shown in Figure 7. In this mechanism, each token only attends to a fixed-size window of w preceding tokens, where $w = 512$. This reduces the computational complexity of the model from $O(n^2)$ to $O(n \cdot w)$. This optimization is key for processing longer narratives on edge devices without major slowdowns. Out of the 18 transformer blocks, 15 use SWA, while 3 are full-attention blocks to keep some global context ability.

6) Positional Encoding (RoPE): To tell the model the order of tokens, Rotary Position Embeddings (RoPE) are used. RoPE rotates the query and key vectors based on their position. The rotation angle θ for a vector at position p and index i is defined by Equation 6:

$$\theta = \omega_i p \quad \text{where} \quad \omega_i = \frac{1}{10000^{2i/d}} \tag{6}$$

Algorithm 2 Real-Time Alert Generation and Prioritization

Require: Raw text T_{slm} from CM-SLM
Require: Keyword maps $M_{sev}, M_{loc}, M_{imp}$
Ensure: Structured JSON alert object J_{alert}

$J_{alert} \leftarrow \{\}$
$T_{lower} \leftarrow \text{ToLower}(T_{slm})$
// 1. Assign Severity
if Contains$(T_{lower}, M_{sev}$."critical") **then**
  J_{alert}.severity $\leftarrow$ "CRITICAL"
elsif Contains$(T_{lower}, M_{sev}$."major") **then**
  J_{alert}.severity $\leftarrow$ "HIGH"
else
  J_{alert}.severity $\leftarrow$ "INFO"
end if
// 2. Extract Key Information (Example)
J_{alert}.location $\leftarrow$ ExtractRegex(T_{slm}, M_{loc})
J_{alert}.impact $\leftarrow$ ExtractRegex(T_{slm}, M_{imp})
J_{alert}.advice $\leftarrow$ ExtractAdvice(T_{slm})
J_{alert}.timestamp $\leftarrow$ CurrentTime()
return J_{alert}

Here, d denotes the dimensionality of the embedding vector. This method has shown better performance in maintaining relative positional information and allows for better performance on sequence lengths not seen during training.

7) Model Optimization for Edge Deployment: The final step is model optimization. Once FP32 model is fully trained, 8-bit post-training quantization (INT8) is applied to it. This converts the model's 32-bit floating-point weights to 8-bit integers. As discussed in Section II-D, this step is vital for the target application, as it reduces the model's storage and memory footprint by approximately 4x (e.g., from ∼1.1 GB to ∼270 MB) and accelerates inference speed, especially on CPUs or specialized edge hardware.

C. Downstream Decision-Support Algorithms

The text output from the CM-SLM is a human-readable narrative. To make this narrative useful for programs, a set of three downstream algorithms processes this text and translates it into structured data for an application or dashboard.

1) Real-Time Alert Generation: The first algorithm, detailed in Algorithm 2, is responsible for analyzing the SLM's free-text response. It uses preset keywords and regular expressions to extract key-value pairs i.e IMPACT, LOCATION, and ADVICE, from the output. It also assigns a severity level based on keywords found in the text (e.g., critical, major → "SEVERITY_CRITICAL"). The output is a structured JSON object that is used to map to the dashboard.

2) Adaptive Rerouting Heuristic: The structured alert from Algorithm 2 can then be given to a routing system. Algorithm 3 shows a simple method for this. It takes the alert and a map of the road network. If the alert is of 'CRITICAL' severity, it strongly penalizes or removes the affected road sections from the navigation graph. For 'HIGH' severity, it might only add a large time penalty. This allows a GPS or traffic management

Algorithm 3 Adaptive Rerouting Heuristic

Require: JSON alert J_{alert}
Require: Road network graph $G = (V, E)$
Ensure: Updated graph G'
$G' \leftarrow G$
$affected\_segment \leftarrow J_{alert}$.location
$edge \leftarrow$ FindEdge$(G', affected_segment)$
switch J_{alert}.severity **do**
  case "CRITICAL":
    // Mark edge as unusable
    G'.RemoveEdge$(edge)$
  case "HIGH":
    // Add severe time penalty
    $edge$.weight $\leftarrow edge$.weight $\times 10$
  case "INFO":
    // Add minor time penalty
    $edge$.weight $\leftarrow edge$.weight $\times 2$
end switch
return G'

Algorithm 4 Dynamic Anomaly Trigger

Require: New data point x_t
Require: Rolling stats $\mu_{roll}, \sigma_{roll}$
Require: Weather data W_{data}
Require: Base threshold Th_{base} (e.g., 3.0)
Ensure: Boolean decision 'TriggerSLM'
// 1. Calculate Anomaly Score
$score \leftarrow |(x_t - \mu_{roll})/\sigma_{roll}|$
// 2. Adjust Threshold Dynamically
$Th_{dynamic} \leftarrow Th_{base}$
if W_{data}.event $\neq$ "NONE" **then**
  // Increase sensitivity during weather events
  $Th_{dynamic} \leftarrow Th_{base} \times 0.5$
end if
// 3. Make Trigger Decision
if $score > Th_{dynamic}$ **then**
  return TRUE
else
  return FALSE
end if

system to dynamically change traffic routes based on the semantic meaning provided by the SLM.

3) Dynamic Anomaly Trigger: Finally, the system must decide when to ask the SLM. Checking every data point would be inefficient. Algorithm 4 suggests a dynamic threshold to manage this. It keeps a rolling statistical baseline of mean and standard deviation for the input traffic data. It also calculates the anomaly score for new data points. Importantly, the threshold to trigger an alert is not fixed; it is made more sensitive if external data reports an active climate event. This stops alert fatigue during normal conditions but makes it more sensitive when disruptions are likely.

V. Experimental Setup

This section provides the setup used for training, optimising, and evaluating the ClimateMobility-Small Language Model (CM-SLM). All choices, from dataset splits to evaluation metrics, were designed for a fair, repeatable, and thorough comparison against baseline models, with a specific focus on the end-goal of real-time edge deployment.

A. Dataset and Baselines

1) CMNC Dataset Splits and Statistics: The 1 million entry Climate-Mobility Narrative Corpus (CMNC), generated as described in Section IV-A, was the dataset used for training and evaluation. The dataset was divided into three standard splits: 80% (800,000 entries) were allocated to the training set, 10% (100,000 entries) to the validation set, and the remaining 10% (100,000 entries) to the test set.

To assure statistical strength and consider random variations, all training and evaluation were performed 10 times using different random seeds values. The validation set was used only for hyperparameter tuning and selecting the best model checkpoint, while the test set was kept for the final performance report in Section VI.

2) Baseline Model Implementation: To provide a fair comparison, the CM-SLM was benchmarked against other well-known, open-source SLMs from the original paper, including **SmolLM2 (360M)**, **Qwen 2.5 (0.5B)**, and **SmolLM2 (135M)**.

An important difference must be noted: the CM-SLM was trained from scratch on the CMNC.The baseline models, which had already been trained on generic web-scale text, were fine-tuned using the CMNC. All models were trained/fine-tuned for the same number of steps on the same training set, and the checkpoint with the lowest validation perplexity was chosen.

B. Evaluation Metrics

To assess the model's capabilities, a set of language quality and performance measurements was used, such as

1) Language Quality Metrics:

- **Perplexity (PPL):** This is the main metric for language model quality. It measures how surprised a model is by the test set. A low perplexity score means that the predicted words closely match the ground-truth text, indicating a better understanding of the domain's language and patterns.
- **BLEU (Bilingual Evaluation Understudy):** A precision-based metric that measures the n-gram overlap between the generated and reference texts. It is a good indicator of how well the sentences flow and their accuracy.
- **ROUGE-L (Recall-Oriented Understudy):** A metric based on recall that finds the longest common subsequence (LCS) between the generated and reference texts. It is effective at evaluating a model's ability to get the main points of the target response.

TABLE IV
HYPERPARAMETER SEARCH SPACE FOR CM-SLM TRAINING.

Hyperparameter	Distribution	Search Range
Learning Rate	Log-Uniform	$[1 \times 10^{-5}, 1 \times 10^{-3}]$
Weight Decay	Uniform	[0.0, 0.1]
Batch Size	Choice	[16, 32, 64]
Dropout Rate	Uniform	[0.05, 0.2]
Warmup Steps	Integer	[500, 1000, 2000]

2) Performance Metrics:

- **Inference Speed (ms/token):** This measures the time taken for the model to generate a single token on the target edge device. This is a key metric for real-time applications, as a high latency (slow response) makes a model not usable for emergency decision support.
- **Memory Footprint (MB):** This measures the amount of RAM required by the quantized 8-bit model on the edge device. Lower values indicate better memory efficiency.
- **Power Consumption (W):** Average power drawn by the edge device (Raspberry Pi 5) during sustained model inference. Low values are better as they show the model's efficiency.
- **CPU Utilization (%):** Average percentage of CPU utilization during inference on the edge device (Raspberry Pi 5). Low values indicate less strain on the processor.

The mean and standard deviation for all performance metrics are reported across the 10 independent runs.

C. Hardware Environments

To show the actual development process, two separate hardware environments were used.

1) Training Environment: All model training, fine-tuning, and hyperparameter searches were run on a Kaggle notebook with **NVIDIA A100 (80GB)** GPU environment.

2) Simulated Edge Environment: All edge performance metrics (Inference Speed, Memory Footprint, Power Consumption, and CPU Utilization) were measured on a resource-constrained edge device: a **Raspberry Pi 5, 8GB RAM model**. Measurements were taken under controlled conditions to ensure realistic performance.

D. Model Architecture and Training Protocol

1) CM-SLM Final Architecture: As detailed in Section IV-B, the final CM-SLM architecture is an 18-layer decoder-only transformer with an embedding dimension of 1024 (standard for Gemma 270M), Multi-Query Attention, Sliding Window Attention, and RoPE positional embeddings.

2) Hyperparameter Tuning: An automated hyperparameter search was performed to find the optimal value. The search methodically explored a preset range of values for the most critical parameters. The best performing combination, as measured by the lowest perplexity on the validation set, was then selected for the final, full duration training run. The search space for this optimization is defined in Table IV. The final, optimal parameters chosen by this process are presented in Section VI-A.

VI. RESULTS AND DISCUSSION

The analysis first reports the final model configuration, then provides a quantitative comparison against baseline models. It then presents a qualitative analysis of the model's outputs and a series of ablation studies to validate the core hypotheses of this work. All results are reported as the mean $\pm$ standard deviation across the 10 independent runs, as described in Section V-A1, to ensure statistical correctness.

A. Model Configuration and Tuning Results

The hyperparameter search detailed in Section V-D2 was performed, and the optimal value was selected based on the lowest perplexity achieved on the validation set. The final, best-performing hyperparameters used to train the CM-SLM for all 10 runs are presented in Table V.

The sensitivity check that led to choosing the optimal 3×10^{-4} learning rate is shown in Figure 8. This analysis confirms the choice by showing a clear spot where a lower or higher rate would worsen model performance.

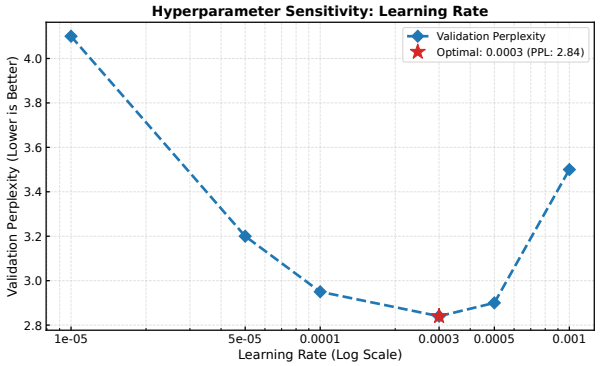

Fig. 8. Hyperparameter sensitivity analysis for the learning rate, plotting validation perplexity against a log-scale of learning rates. The 3×10^{-4} rate chosen (Table V) is validated as the optimal choice.

B. Quantitative Performance Analysis

The main evaluation compared the CM-SLM (trained from scratch) against the fine-tuned baseline models. The full results for both language quality and edge performance are shown in Table VI.

TABLE V
OPTIMAL HYPERPARAMETER VALUES FOR CM-SLM TRAINING, SELECTED VIA THE TUNING PROCESS.

Hyperparameter	Value	Rationale
Learning Rate	3×10^{-4}	A common, effective rate for AdamW with good stability.
Weight Decay	0.05	Provided mild regularization, preventing overfitting.
Batch Size	32	Balanced training speed with gradient stability.
Dropout Rate	0.1	Standard dropout for a model of this size.
Warmup Steps	1000	Allowed the model to stabilize before converging.

TABLE VI
COMPLETE QUANTITATIVE EVALUATION (MEAN ± STD. DEV. OVER 10 RUNS). LOWER IS BETTER FOR PPL, MEMORY, SPEED, POWER, AND CPU UTIL. HIGHER IS BETTER FOR BLEU AND ROUGE-L.

Metric Category	Model	Language Quality		Edge Performance (on Raspberry Pi 5)			
		PPL ↓	BLEU ↑	Memory (MB) ↓	Speed (ms/token) ↓	Power (W) ↓	CPU Util (%) ↓
Models	**CM-SLM (270M)**	**2.84 ± 0.05**	**0.68 ± 0.02**	**273 ± 2**	**45.2 ± 3.1**	**4.5 ± 0.3**	**40 ± 4**
	Qwen 2.5 (0.5B)	4.12 ± 0.10	0.55 ± 0.03	505 ± 4	78.5 ± 4.5	6.8 ± 0.5	65 ± 6
	SmolLM2 (360M)	4.95 ± 0.12	0.51 ± 0.03	364 ± 3	66.1 ± 3.9	5.5 ± 0.4	55 ± 5
	SmolLM2 (135M)	6.31 ± 0.15	0.42 ± 0.04	**138 ± 2**	**28.3 ± 2.5**	**3.2 ± 0.2**	**30 ± 3**

The results clearly show the advantage of the domain-specific pretraining. The CM-SLM achieves a significantly lower perplexity (2.84) than all baselines, demonstrating a much better understanding of the CMNC's language and structure. This is true even though the Qwen model has nearly double the parameters. The fine-tuned generalist models (Qwen, SmolLM2) could not adapt as well, resulting in higher perplexity.

Furthermore, the CM-SLM finds a great balance in performance. It is significantly faster and has a smaller memory footprint than the larger Qwen and SmolLM2 (360M) models. While the SmolLM2 (135M) is the smallest and fastest, its language quality (PPL 6.31) is much worse, making it not good for this task.

Figures 9 and 11 statistically confirm the training process, confirming the thoroughness of the 10-run analysis; they show the mean performance (represented by the line) bounded by a ± 1 standard deviation (the shaded region). The results are clear: the CM-SLM is not only faster (Fig. 9) but also achieves a higher BLEU score (Fig. 11), and is also significantly more stable and steady in its convergence, as shown by its smaller confidence band compared to the baselines. A complete view of the final model's trade-offs is presented in the radar plot (Figure 10), which visually shows the CM-SLM's better balanced profile.

C. Qualitative Analysis

While quantitative metrics are essential, qualitative analysis is needed to check the usefulness of the generated text. Table VII presents several example prompts from the held-out test set and the text that goes with it generated by the CM-SLM and the next-best baseline, Qwen 0.5B.

The examples show that both models can identify the event, but the CM-SLM's responses are more specific, useful, and use the right domain terms. The baseline model's text is often general or a bit unclear, which shows its general-purpose pretraining.

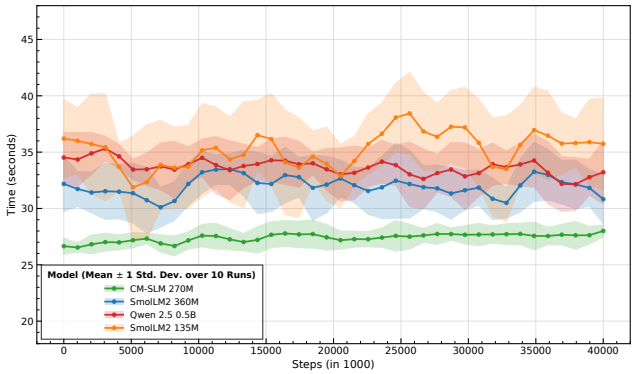

Fig. 9. Statistical analysis of training time per 1000 steps over 10 independent runs, the mean (solid line) and the ± 1 standard deviation interval (shaded area). This validates that the CM-SLM is not only the fastest but also the most stable and consistent during training.

D. Ablation Studies

To further check the framework, three ablation studies were conducted to separate the effect of (1) the corpus size, (2) key architectural components, and (3) model quantization.

1) Impact of Corpus Size: A main hypothesis of this work is that a large, domain-specific corpus is vital. To test this, again the CM-SLM was trained from scratch on smaller subsets of the CMNC: 100,000 and 500,000 entries. The validation perplexity was then compared to the model trained on the full 1,000,000-entry corpus.

Results, presented in Figure 12. The model also becomes less perplexed when the corpus size is larger, decreasing to

TABLE VII
QUALITATIVE COMPARISON OF GENERATED RESPONSES FROM THE TEST DATASET.

Test Set Prompt	CM-SLM (270M) Response	Qwen 0.5B (Baseline) Response
`EVENT: flood, LOCATION: I-80, SEVERITY: critical, TREND: rising`	**ALERT:** Critical flash flood on I-80. All lanes are impassable. Water is rising. **ADVICE:** Dispatch high-clearance vehicles only. Reroute all other traffic to Route 30. This is a life-threatening situation.	Flood event on I-80. Very bad. The road is closed. Cars should turn around. Water is high.
`EVENT: snow, LOCATION: Main St, SEVERITY: high, WIND: 40kph`	**ALERT:** High-impact snow event on Main St. Heavy accumulation and 40kph winds causing whiteout conditions. **ADVICE:** Advise No Travel order. Dispatch municipal snowplows and salt trucks immediately.	Snow on Main St. It is windy. Bad to drive. Tell people to stop. Call the snow trucks.
`EVENT: accident, LOCATION: US-101, SEVERITY: moderate, DETAIL: 3-car`	**ALERT:** Moderate 3-car collision reported on US-101. Right lane blocked. **ADVICE:** Expect major delays. Dispatch EMS and law enforcement. Suggest merging left.	Accident. US-101. 3 cars. Road is bad. Traffic will be slow. Send help.

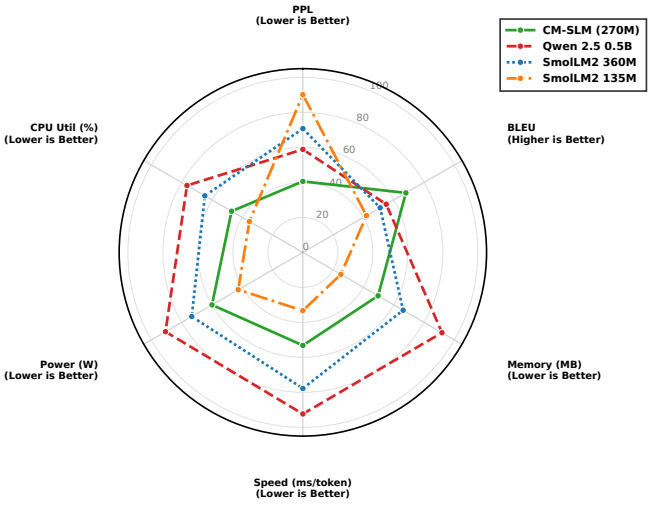

Fig. 10. Radar plot visually comparing model trade-offs using key metrics from Table VI (Raspberry Pi 5 edge performance). Values are scaled 0-100 based on defined maximums. For cost metrics (PPL, Memory, Speed, Power, CPU), lower scores are better, but for the BLEU score, higher is better.

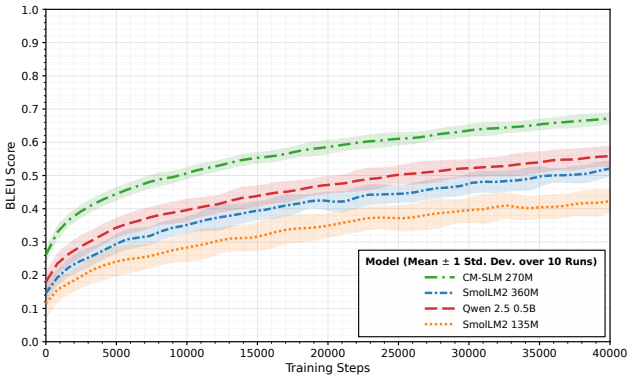

Fig. 11. Statistical analysis of BLEU score convergence over 10 independent runs, the mean convergence (line) buffered by its ± 1 standard deviation (shaded area). The CM-SLM achieves the highest mean BLEU score and exhibits the most consistent performance.

8.9 (100k) to 4.1 (500k) and eventually 2.84 (1M). This is reassuring that with the full size of 1 million entry corpus the rich and varied data required to entrust the model with the in-depth knowledge of the domain is obtained.

2) Impact of Architectural Components: A main hypothesis of this work is that the specific architectural choices outlined in Section IV-B, namely Sliding Window Attention (SWA), Multi-Query Attention (MQA), and Rotary Position Embeddings (RoPE), are key for balancing performance and accuracy. To validate this, a component ablation study was done, presented in Figure 13.

The results clearly support the proposed architecture. Removing SWA (i.e., using full attention) or MQA (i.e., using standard MHA) badly harms inference speed, making the model not usable for real-time edge deployment. On the other hand, removing RoPE in favor of standard positional embeddings greatly hurts model accuracy, making perplexity go up. This confirms that the CM-SLM architecture is a good model to achieve the target of low perplexity and high inference speed.

3) Impact of Model Quantization: The framework uses 8-bit quantization (INT8) to get edge-level performance. However, this optimization often lowers model accuracy. An ablation study checked this trade-off. The model was evaluated in three states: its original 32-bit (FP32) form, 8-bit (INT8) quantization, and a stronger 4-bit (NF4) quantization.

Figure 14 shows the trade-off between speed and accuracy. The FP32 model has the best perplexity (2.81) but is the slowest (120.5 ms/token). The 4-bit model is extremely fast (25.1 ms/token) but suffers a major loss in quality (PPL 9.75). The 8-bit model, however, is the ideal choice: it provides a $\sim$2.6x speedup over the FP32 model with only a very small 1% drop in perplexity (2.84 vs. 2.81). This confirms 8-bit quantization is the best for this deployment.

E. Discussion of Findings

Overall, the results in Table VI and the ablation studies (Figures 12, 14, and 13) strongly support the paper's main idea. This claim is also supported by the statistical 10-run analysis (Figures 9 and 11), which shows the steadiness and

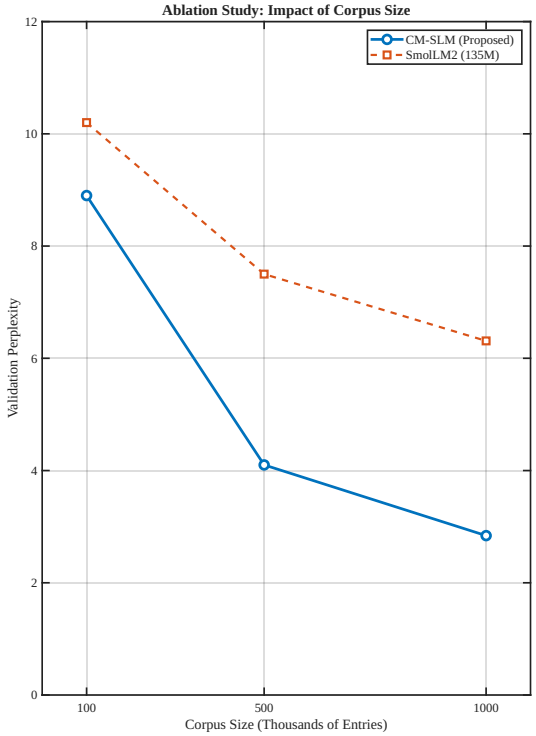

Fig. 12. Ablation study on corpus size. Validation perplexity improves significantly as the number of training narratives increases, confirming the value of the 1M entry CMNC.

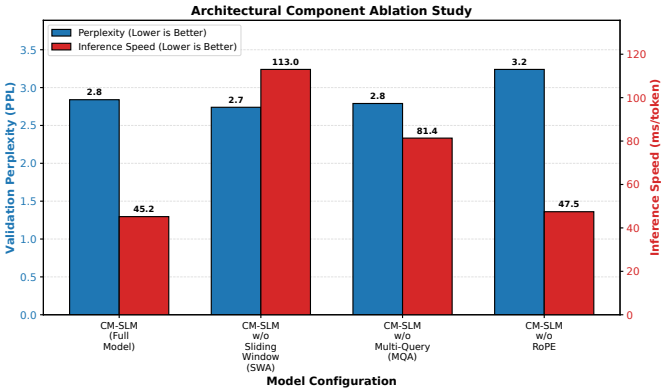

Fig. 13. Architectural component ablation study shows trade-off between model perplexity and inference speed. Removing SWA or MQA destroys performance, while removing RoPE destroys accuracy, validating the CM-SLM's design.

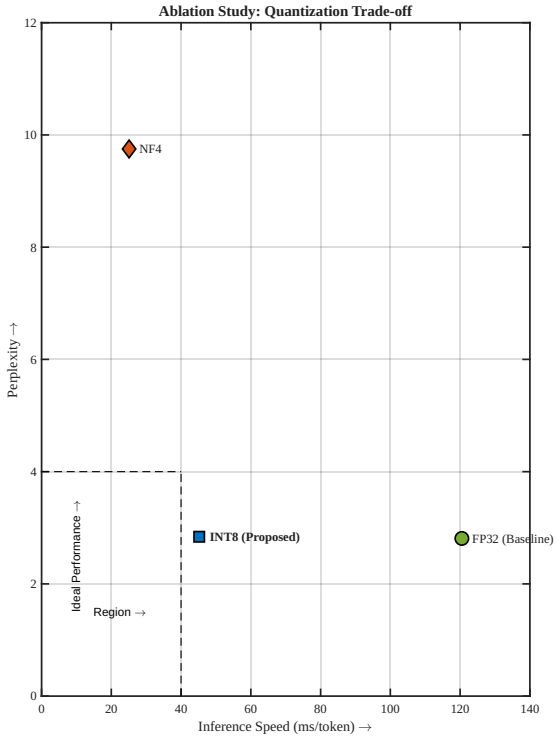

Fig. 14. Ablation study on model quantization, plotting the trade-off between inference speed and perplexity.

stability of the CM-SLM. The CM-SLM's better performance is not because of a larger parameter count but due to its highly specialized training. By training from scratch on a million entry, domain-specific corpus, the model's 270M parameters were fully set up to learn the specific language of climate-mobility, rather than using its ability on general internet knowledge.

The baseline models, even with fine-tuning, could not get past their general pretraining. Their poor performance shows fine-tuning is not enough for niche, high-stakes domains.

Furthermore, the performance analysis in Figure 14 and Table VI gives a practical plan for deployment. The 8-bit quantized 270M model gives a good result: its language quality is good enough to be trusted (PPL 2.84), and its performance (273 MB, 45 ms/token) is well inside the acceptable limits for real-world edge devices like the Raspberry Pi 5 used in our tests.

F. Limitations and Threats to Validity

While rigorous, this study has some limitations.

- **Synthetic Data:** The main limitation is the 100% man-made nature of the CMNC. While based on the complex model from Section IV-A2, it cannot fully get the chaos of real-world disasters.

- **Edge Device Performance:** While the Raspberry Pi 5 provides a realistic edge environment, measured performance could change in a final deployment due to concurrent software processes, specific OS configurations, or thermal management within an enclosure.
- **Single Data Type:** The framework is text-only. A real decision-support system would be better with different types of inputs, like live traffic camera video or audio from emergency radios.

G. Future Work

Based on these limits, future work will look at three key areas.

1) **Real World Fine-Tuning:** The next step is to collect a smaller, high-quality dataset of real-world emergency logs and operator narratives. This dataset will be used to fine-tune the CM-SLM, connecting the synthetic pretraining and real-world use.
2) **Multi-Modal Integration:** Research will look into combining the CM-SLM with a vision model (e.g., CLIP) to create a multi-modal system that can understand both text prompts and live camera pictures.
3) **Federated Learning:** To let the model adapt to new events and local terminologies without risking privacy, a federated learning framework will be studied. Like a fleet of deployed edge devices to work together to fine-tune the main model.

VII. Conclusion

This paper showed a complete framework for developing and deploying a special Small Language Model (SLM) for real-time, on-device decision support in climate-induced urban mobility crises. This work deals with the limits of current models: traditional forecasting models cannot reason, while large language models are too slow and heavy for edge deployment.

The **ClimateMobility-SLM (CM-SLM)**, is a 270M-parameter model trained from scratch on a new, 1 million-entry man-made dataset, the **Climate-Mobility Narrative Corpus (CMNC)**. Generating this corpus was supported by a complex simulation model to make sure the training data was realistic and varied.

The test results are clear. The CM-SLM achieves a mean validation perplexity of 2.84 ± 0.05, doing much better than fine-tuned baseline models of comparable or larger size. This confirms that for specific, important areas, domain-specific pretraining is better than general fine-tuning. Furthermore, the 8-bit quantized model provides this high-quality reasoning with a small memory footprint (273 MB) and a fast inference speed (45.2 ms/token) on a realistic edge device. The ablation studies confirmed that both the 1 million entry corpus and the 8-bit quantization were key and best design choices.

This study provides a practical plan of a new form of intelligent, powerful, and effective urban governance system. This work demonstrates a means of enhancing the safety of the masses and the efficiency of transportation at a time of increasing climatic uncertainty by effectively bridging the gap between large scale reasoning and on-the-ground performance.

References

[1] S. A. Markolf, C. Hoehne, A. Fraser, M. V. Chester, and B. S. Underwood, "Transportation resilience to climate change and extreme weather events–beyond risk and robustness," *Transport policy*, vol. 74, pp. 174–186, 2019.

[2] D. Touloumidis, M. Madas, V. Zeimpekis, and G. Ayfantopoulou, "Weather-related disruptions in transportation and logistics: A systematic literature review and a policy implementation roadmap," *Logistics*, vol. 9, no. 1, p. 32, 2025.

[3] T. V. Ha and M. Arimura, "Mapping urban mobility during extreme weather events using mobile spatial statistics data," *Transportation Research Interdisciplinary Perspectives*, vol. 33, p. 101609, 2025.

[4] J. Huang, T. Zhang, D. Sun, and H. Wang, "Urban human mobility changes based on functional areas during extreme rainstorm event: A case of beijing "23· 7" rainstorm event," *Cities*, vol. 163, p. 106003, 2025.

[5] L. Kamalian *et al.*, "Analysis of the impact of climate-driven extreme weather events on uk train delays using bayesian network approach," *Reliability Engineering & System Safety*, vol. 254, p. 110390, 2025.

[6] B.-L. Ye, M. Zhang, L. Li, C. Liu, and W. Wu, "A survey of traffic flow prediction methods based on long short-term memory networks," *IEEE Intelligent Transportation Systems Magazine*, vol. 16, no. 5, pp. 87–112, 2024.

[7] P. Redhu, K. Kumar *et al.*, "Short-term traffic flow prediction based on optimized deep learning neural network: Pso-bi-lstm," *Physica A: Statistical Mechanics and its Applications*, vol. 625, p. 129001, 2023.

[8] Y.-T. Chen, A. Liu, C. Li, S. Li, and X. Yang, "Traffic flow prediction based on spatial-temporal multi factor fusion graph convolutional networks," *Scientific Reports*, vol. 15, no. 1, p. 12612, 2025.

[9] J. Tang, R. Zhu, F. Wu, X. He, J. Huang, X. Zhou, and Y. Sun, "Deep spatio-temporal dependent convolutional lstm network for traffic flow prediction," *Scientific Reports*, vol. 15, no. 1, p. 11743, 2025.

[10] S. Bhardwaj, P. Singh, and M. K. Pandit, "A survey on the integration and optimization of large language models in edge computing environments," in *2024 16th International Conference on Computer and Automation Engineering (ICCAE)*. IEEE, 2024, pp. 168–172.

[11] S. Ying, Z. Li, and M. Yu, "Beyond words: evaluating large language models in transportation planning," *Geo-spatial Information Science*, pp. 1–23, 2025.

[12] R. Sajja, S. Xiong, O. Mermer, M. Y. Sermet, and I. Demir, "A comprehensive bibliometric analysis of large language models in hydrology and environmental sciences," 2025.

[13] O. Lacombe, K. Kenealy, K. Black, R. Kumar, F. Visin, and J. Zhang, "Introducing gemma 3 270m: The compact model for hyper-efficient ai," https://developers.googleblog.com/en/introducing-gemma-3-270m/, 2025, google Developers Blog.

[14] S. Jang and R. Morabito, "Edge-first language model inference: Models, metrics, and tradeoffs," *arXiv preprint arXiv:2505.16508*, 2025.

[15] M. R. Islam, N. Dhar, B. Deng, T. N. Nguyen, S. He, and K. Suo, "Characterizing and understanding the performance of small language models on edge devices," in *2024 IEEE International Performance, Computing, and Communications Conference (IPCCC)*. IEEE, 2024, pp. 1–10.

[16] A. Zhang, "Dynamic graph convolutional networks with temporal representation learning for traffic flow prediction," *Scientific Reports*, vol. 15, no. 1, p. 17270, 2025.

[17] S. Arunachalam, "Long-term and short-term traffic flow prediction with different weather conditions using optimized wheel-graph attention based neural network," *Future Generation Computer Systems*, p. 108198, 2025.

[18] H. Xiao, B. Zou, and J. Xiao, "Graph convolution networks based on adaptive spatiotemporal attention for traffic flow forecasting," *Scientific Reports*, vol. 15, no. 1, p. 8935, 2025.

[19] J. Zhang, Y. Yang, X. Wu, and S. Li, "Spatio-temporal transformer and graph convolutional networks based traffic flow prediction," *Scientific Reports*, vol. 15, no. 1, p. 24299, 2025.

[20] S. Komarovsky and J. Haddad, "Spatio-temporal graph convolutional neural network for traffic signal prediction in large-scale urban networks," *Transportation Research Interdisciplinary Perspectives*, vol. 32, p. 101482, 2025.

[21] N. Maksoud, H. AlJassmi, L. Ali, and A. R. Masoud, "Applications of large language models and generative ai in transportation: A systematic review and bibliometric analysis," *Transportation Research Interdisciplinary Perspectives*, vol. 34, p. 101699, 2025.

[22] R. Zhang, K. Xiong, H. Du, D. Niyato, J. Kang, X. Shen, and H. V. Poor, "Generative ai-enabled vehicular networks: Fundamentals, framework, and case study," *IEEE Network*, vol. 38, no. 4, pp. 259–267, 2024.

[23] R. D. Rachmanto, Z. Sukma, A. N. Nabhaan, A. Setyanto, T. Jiang, and I. K. Kim, "Characterizing deep learning model compression with post-training quantization on accelerated edge devices," in *2024 IEEE International Conference on Edge Computing and Communications (EDGE)*. IEEE, 2024, pp. 110–120.

[24] C. J. Schaefer, S. Joshi, S. Li, and R. Blazquez, "Edge inference with fully differentiable quantized mixed precision neural networks," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, pp. 8460–8469.

[25] X. Lan, Y. Zeng, X. Wei, T. Zhang, Y. Wang, C. Huang, and W. He, "Counterclockwise block-by-block knowledge distillation for neural network compression," *Scientific Reports*, vol. 15, no. 1, p. 11369, 2025.

[26] A. A. Neeralgi, R. R. Avarsekar, M. Bhajantri, R. Khatod, U. Kulkarni, and S. M. Nadaf, "Knowledge distillation using deep learning techniques: A survey," in *2024 IEEE Conference on Engineering Informatics (ICEI)*. IEEE, 2024, pp. 1–10.

[27] L. Chen, D. Chai, and L. Wang, "Uctb: An urban computing tool box for spatiotemporal crowd flow prediction," *arXiv: 2306.04144*, 2023.

[28] Y. Lu, M. Shen, H. Wang, X. Wang, C. van Rechem, T. Fu, and W. Wei, "Machine learning for synthetic data generation: a review," *arXiv preprint arXiv:2302.04062*, 2023.

[29] V. C. Pezoulas, D. I. Zaridis, E. Mylona, C. Androutsos, K. Apostolidis, N. S. Tachos, and D. I. Fotiadis, "Synthetic data generation methods in healthcare: A review on open-source tools and methods," *Computational and structural biotechnology journal*, vol. 23, pp. 2892–2910, 2024.

[30] L. Sobrie and M. Verschelde, "Real-time decision support for human–machine interaction in digital railway control rooms," *Decision Support Systems*, vol. 181, p. 114216, 2024.