FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Simulation and Prediction of Public Transport Networks Using Large Events Models

João Pedro Seabra Pedrosa Botelho de Sousa

Mestrado em Engenharia Informática e Computação

Supervisor: Prof. João Pedro Mendes Moreira

Second Supervisor: Prof. Tiago Neves

July 14, 2026

Simulation and Prediction of Public Transport Networks Using Large Events Models

João Pedro Seabra Pedrosa Botelho de Sousa

Mestrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

President: Prof. Mariana Curado Malta
Referee: Prof. Bruno Martins
Referee: Prof. João Pedro Mendes Moreira

July 14, 2026

Resumo

As redes de transporte público apresentam dinâmicas complexas que resultam da interação entre os movimentos dos veículos, a procura de passageiros e as condições de tráfego. Os modelos preditivos existentes são predominantemente discriminativos e específicos para cada tarefa, centrados na estimação de tempos de chegada, e não suportam a simulação de sequências heterogéneas de eventos operacionais no âmbito de um único enquadramento generativo. Esta dissertação propõe a adaptação dos Modelos de Grandes Eventos (Large Events Model (LEM)) ao domínio do transporte público urbano, com o objetivo de unificar a simulação e a previsão de operações de redes de autocarros num único enquadramento generativo.

A abordagem reformula a evolução operacional de uma rede de autocarros como uma sequência de tokens discretos, codificando cada evento de paragem num bloco de oito tokens: tipo de evento, posição na sequência de paragens, identidade da paragem, bin de tempo de chegada, bin de atraso, bin de validações de passageiros, bin de intervalo entre veículos e um delimitador de fim de evento. Um transformador autorregressivo decoder-only (EventGPT, 1,82 M de parâmetros) é treinado com o objetivo padrão de modelação causal de linguagem sobre estas sequências. Os dados experimentais provêm da rede de autocarros Vianorbus (Porto, Portugal), abrangendo 5.371.754 eventos de paragem observados ao longo de 82 dias de operação, integrando feeds de horários General Transit Feed Specification (GTFS), atualizações em tempo real Global Positioning System (GPS)/Automatic Vehicle Location (AVL) e validações de passageiros.

O modelo obtém uma perplexidade de teste de 1,454 e a precisão agregada de tokens de 91,7% (avaliada nas 24.626/31.862 sequências de teste cujo comprimento permite avaliação em ciclo fechado) é impulsionada principalmente por campos estruturalmente determinísticos, com o token de atraso a atingir 56,0% de precisão top-1 em ciclo fechado (96,4% top-5 com resolução de 30 segundos por bin). No modo de rollout guiado por GTFS com prefixo de 20 paragens, a avaliação operacionalmente mais relevante, o EventGPT atinge um Mean Absolute Error (MAE) de atraso de 0,49 minutos, superando todas as dez baselines avaliadas, com uma melhoria de 5,8% face à melhor baseline (Média Histórica: 0,52 min). As sequências geradas apresentam qualidade estrutural quase perfeita: a taxa de eventos completos de 100% é garantida pela descodificação com restrições de esquema, enquanto a monotonicidade temporal de 99,9% é uma propriedade aprendida não explicitamente imposta na função de perda. Experiências multi-operador com três operadores portugueses e um estudo de ablação da tokenização caracterizam ainda as capacidades e limitações do modelo. Os resultados indicam que o paradigma LEM pode ser adaptado ao domínio sequencial das operações de transporte público, sugerindo uma direção de investigação promissora para outros sistemas complexos orientados a eventos.

Palavras-chave: Modelos de Grandes Eventos • Simulação de transporte público • Transformador autorregressivo • Previsão de atrasos • Modelação de sequências de eventos • Mobilidade urbana

Abstract

Public transport networks exhibit complex dynamics that arise from the interplay between vehicle movements, passenger demand, and traffic conditions. Existing predictive models are primarily discriminative and task-specific, focused on arrival time estimation, and do not support the simulation of heterogeneous operational event sequences within a unified generative framework. This dissertation proposes the adaptation of Large Events Models to the domain of urban public transport, aiming to unify simulation and prediction of bus network operations within a single generative framework.

The approach reformulates the operational evolution of a bus network as a sequence of discrete tokens, encoding each stop event as an eight-token block: event type, stop sequence position, stop identity, arrival time bin, delay bin, passenger validation bin, inter-vehicle headway bin, and an event-boundary delimiter. A decoder-only autoregressive transformer (EventGPT, 1.82M parameters) is trained with the standard causal language modeling objective over these sequences. The experimental data come from the Vianorbus bus network (Porto, Portugal), covering 5,371,754 observed stop events over 82 days of operations, integrating General Transit Feed Specification (GTFS) schedule feeds, Global Positioning System (GPS)/Automatic Vehicle Location (AVL) real-time updates, and smart-card passenger validations.

The model achieves a test perplexity of 1.454, and aggregate token accuracy of 91.7% (evaluated on the 24,626/31,862 test sequences whose length permits closed-loop evaluation) is driven primarily by structurally deterministic fields, with the delay token reaching 56.0% closed-loop top-1 accuracy (96.4% top-5 at 30-second bin resolution). In GTFS-guided delay-only mode with a 20-stop prefix, the operationally meaningful evaluation, EventGPT attains a delay Mean Absolute Error (MAE) of 0.49 minutes, outperforming all ten evaluated baselines, with a 5.8% improvement over the best-performing Historical Average (0.52 min). Generated sequences exhibit near-perfect structural quality: the 100% complete event rate is guaranteed by schema-constrained decoding, while 99.9% temporal monotonicity is a learned property not explicitly enforced in the training loss. Multi-operator experiments across three Portuguese operators and a tokenization ablation study further characterize the model's capabilities and limitations. The results indicate that the Large Events Model (LEM) paradigm can be adapted to the sequential domain of public transport operations, suggesting a promising research direction for other complex event-driven systems.

Keywords: Large Events Models • Public transport simulation • Autoregressive transformer • Delay prediction • Event sequence modeling • Urban mobility

The United Nations Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) provide a global framework for achieving a better and more sustainable future for all. It includes 17 goals[1] to address the world's most pressing challenges, including poverty, inequality, climate change, environmental degradation, peace, and justice.

This dissertation addresses the simulation and prediction of public transport networks using Large Events Models, with the goal of improving the reliability and efficiency of urban transit systems. By enabling accurate real-time delay prediction and schedule adherence monitoring from Global Positioning System (GPS), smart-card, and General Transit Feed Specification (GTFS) data, the work contributes to more sustainable, inclusive, and data-driven mobility. The following SDGs are directly addressed by this project:

The specific Sustainable Development Goals mentioned have the following names:

SDG 9 Build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation

SDG 11 Make cities and human settlements inclusive, safe, resilient and sustainable

SDG 13 Take urgent action to combat climate change and its impacts

[1] https://sdgs.un.org/goals

SDG	Target	Contribution	Performance Indicators and Metrics
9	9.1	The GTFS-guided inference mode integrates directly with existing public transport infrastructure standards, improving the resilience and reliability of transit networks by providing accurate real-time delay predictions from live vehicle data.	Token prediction accuracy (91.7%), model perplexity (1.454), feasibility of integration with live GTFS feeds across multiple operators.
	9.5	The EventGPT architecture contributes a novel research framework for sequential event prediction in transport, demonstrating that a compact transformer model (1.82M parameters) trained on real operational data can capture complex delay dynamics across different networks.	Negative Log-Likelihood (NLL) and perplexity across ablation configurations, 5.8% improvement in delay Mean Absolute Error (MAE) over the Historical Average baseline (primary evaluation at prefix=20, horizon=3) and reproducibility of results on Vianorbus, Próximo, and Beja datasets.
11	11.2	By predicting bus stop delays with a MAE of 0.49 minutes (GTFS-guided delay-only mode), the model enables operators to offer more punctual and reliable services, reducing waiting times and encouraging modal shift from private vehicles to public transport.	Delay MAE (0.49 min), percentage reduction in delay variance per route and ridership growth on lines with model-informed scheduling.
	11.3	Processing smart-card validations, GPS traces, and GTFS data at scale (5,371,754 stop events over 82 days for Vianorbus alone) supports data-driven urban planning, enabling transit authorities to optimize network coverage, timetable design, and resource allocation.	Volume of stop events processed, number of municipalities and operators covered and planning decisions informed by predicted delay patterns.
13	13.2	Improved public transport reliability reduces dependence on private vehicles, contributing to lower urban CO_2 emissions. Accurate delay forecasts can inform demand-responsive scheduling and fleet electrification planning, embedding climate considerations into operational decisions.	Estimated CO_2 reduction per percentage-point increase in public transport modal share and reduction in unnecessary vehicle kilometers operated through model-informed schedule adjustments.

Declaration of honour

I declare that this dissertation is my original work and has not previously been submitted or assessed in any other course or curricular unit, whether at this or any other institution.

References to other authors (statements, ideas, thoughts), as well as the use of published works fully complies with the rules of attribution and are appropriately cited in the text and bibliography, in accordance with reference standards. I am aware that plagiarism and self-plagiarism constitute academic misconduct.

I further declare that I have used Artificial Intelligence tools to assist in improving the writing of this dissertation. I take responsibility for all content, ideas, and arguments presented in this dissertation, having carefully reviewed all AI-assisted text for errors, inaccuracies, and unfounded attributions.

João Pedro Seabra Pedrosa Botelho de Sousa

Assinado por: **João Pedro Seabra Pedrosa Botelho de Sousa**
Num. de Identificação: 15039419
Data: 2026.07.14 14:43:38 +0100

Acknowledgements

I would like to thank my supervisors at FEUP, Professor João Pedro Mendes Moreira and Professor Tiago Neves, for their support and feedback throughout this dissertation.

A special thank you to Paulo Correia and Ubirider for providing access to real-world operational data and for the opportunity to develop this work in an industry context.

Finally, I am grateful to my family and friends for their encouragement and patience throughout this journey.

João Pedro Seabra Pedrosa Botelho de Sousa

"All models are wrong, but some are useful."

George E. P. Box

Contents

List of Figures

List of Tables

List of Acronyms

ACM	Association for Computing Machinery
APC	Automatic Passenger Counter
AR	Autoregressive
ARIMA	Autoregressive Integrated Moving Average
AVL	Automatic Vehicle Location
DBIN	Delay Bin
ETA	Estimated Time of Arrival
FFN	Feed-Forward Network
GCN	Graph Convolutional Network
GELU	Gaussian Error Linear Unit
GPT	Generative Pre-trained Transformer
GPS	Global Positioning System
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
GTFS	General Transit Feed Specification
HDWAY	Headway Bin
IEEE	Institute of Electrical and Electronics Engineers
JSON	JavaScript Object Notation
KV	Key-Value
LEM	Large Events Model
LLM	Large Language Model
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MLP	Multi-Layer Perceptron
MPS	Metal Performance Shaders
NLL	Negative Log-Likelihood
RNN	Recurrent Neural Network
RMSE	Root Mean Squared Error
TBIN	Time Bin
VALBIN	Passenger Validation Bin

Chapter 1

Introduction

1.1 Context

Urban transportation systems are dynamic and complex, impacted by shifting passenger demand, vehicle delays, and traffic conditions. Seasonal fluctuations, unexpected major events, and operational setbacks further complicate these systems, making it difficult to provide precise real-time forecasts for public transportation operations. The interrelated and dynamic character of transportation systems is not adequately captured by traditional predictive models, which frequently rely on static assumptions or simplified rule-based systems. The goal of this dissertation is to model transport activities as tokenized event sequences to adapt Large Events Model (LEM)s to public transportation networks. These sequences serve as a more adaptable and data-driven alternative to traditional models by simulating the dynamic behavior of transport networks and representing a range of transport events, such as bus arrivals, boardings, and delays.

Recent advances in sequence modeling have demonstrated a strong predictive ability for sequential and interdependent event data. The foundation for expressing complex systems as tokenized event streams and forecasting future states from them is provided by LEMs, which were first used in fields such as sports analytics [18]. This dissertation examines the adaptation of these models to public transport networks, enabling the simulation of bus network operational dynamics using data-driven learning rather than prescriptive rules.

1.2 Motivation

Reliable and efficient public transport is essential for sustainable urban mobility and reduced traffic congestion. Accurate predictions not only improve operational efficiency, scheduling, and resource allocation, but also enable passengers to plan their journeys effectively, reducing waiting times and enhancing service satisfaction. However, existing models struggle to address the complexity and unpredictability inherent in real-time transport systems, especially under fluctuating conditions. This dissertation proposes a novel approach that leverages Large Events Models to simulate and predict transport network dynamics, offering a more flexible, data-driven alternative

to conventional methods. Learning from heterogeneous data sources, such as Global Positioning System (GPS) traces, General Transit Feed Specification (GTFS) feeds, and passenger counts, the proposed method adapts to real-world dynamics, capturing intricate temporal and causal patterns that static models cannot.

Because real-world transport operations are nonlinear and adaptable, current rule-based or static modeling approaches often fall short. LEMs are a promising paradigm change because they can capture intricate temporal and causal patterns across multimodal networks by directly learning from heterogeneous historical data. Their promise of offering a scalable and empirically supported substitute for conventional simulations is highlighted by their proven ability to represent robust event sequences in other areas [18].

1.3 Problem

The core problem lies in the absence of simulation and prediction frameworks that can represent the sequential and interdependent dynamics of transport networks at the operational scale. Due to their simplistic interactions and inability to adjust to real-time unpredictability, Estimated Time of Arrival (ETA) predictions are still incorrect under existing methodologies. To solve the problem, transport operations are reformulated as token sequences in this dissertation. These sequences, built from raw data such as GTFS feeds, GPS traces, and passenger validation counts, serve as inputs to auto-regressive models capable of capturing complex dynamics and producing predictive simulations.

The challenge lies in (i) converting diverse raw data streams into uniform tokenized sequences, (ii) training models that can use these sequences to discover significant temporal and causal patterns, and (iii) validating the interpretability and usefulness of the results for transportation stakeholders while verifying them against the actual ground truth.

1.4 Hypothesis

Representing public transport operations as tokenized event sequences and modeling them through autoregressive Large Events Models will enable realistic simulation of bus network operations and competitive ETA prediction performance compared to traditional and discriminative deep learning baselines, under in-distribution operating conditions.

This hypothesis relies on three assumptions:

1. The tokenization of various streams of events, such as arrivals, boardings, and delays, maintains crucial causal and temporal relationships.

2. Large Events Models have been shown to be applicable to the dynamics of sequential event domains.

3. Strong and scalable predictions can be made using a data-driven strategy, which improves the accuracy of network simulations and ETA forecasts.

If confirmed, this will show that public transport networks can be represented as intricate event systems in which sequence-based learning techniques directly improve predictive accuracy.

1.5 Research Questions

- **RQ1:** How can diverse real-world transport data be effectively tokenized to capture the sequential and causal dependencies within public transport systems for Large Events Models?
- **RQ2:** Can Large Events Models be effectively applied to simulate the dynamics of individual bus trips within an urban bus network, beyond the isolated prediction of ETAs?
- **RQ3:** How do the results of these models compare with traditional approaches in terms of accuracy, scalability, and practical usefulness?

Chapter 2

Literature Review

2.1 Review Approach

This chapter reviews the current state of public transport network modeling and prediction methods, focusing on advances in large-event models and autoregressive techniques. The review examines event tokenisation, spatio-temporal prediction, and real-time ETA forecasting, highlighting key contributions that have enhanced the resilience and reliability of public transport systems.

2.2 Methodology

2.2.1 Sources

To ensure a thorough and reliable coverage of pertinent research on the topic, a literature review was conducted in several reputable digital libraries and scholarly databases. The following primary sources were consulted for this review:

- **Institute of Electrical and Electronics Engineers (IEEE) Xplore:** Well known for emphasizing machine learning, intelligent systems, and transportation technology.
- **Association for Computing Machinery (ACM) Digital Library:** Focuses on sophisticated computational methods, such as predictive modeling and large-scale simulations.
- **Scopus:** Offers more extensive cross-disciplinary coverage, which helps to incorporate different approaches to public transportation forecasting.

2.2.2 Query Formulation

The search queries were designed to capture key ideas related to the dissertation's focus on large-event models, ETA prediction, and public transportation. The following were the search queries:

- **QS1:** ("public transport" OR bus OR transit) AND ("arrival time" OR "ETA" OR "delay prediction") AND (transformer* OR autoregressive OR "Large Events* Model*") AND ("simulation" OR "prediction" OR "network model")

- **QS2:** (spatiotemporal OR "GPS trajectory" OR "mobility data" OR "geospatial data") AND ("public transport" OR "bus network" OR "transport network" OR "urban mobility") AND ("simulation" OR "prediction" OR "ETA" OR "arrival time")
- **QS3:** ("event sequence" OR event-based OR "sequence modeling" OR "time-series") AND (transformer* OR autoregressive OR "large language model" OR TimeGPT OR "Large Event* Model*") AND ("public transport" OR "transport network") AND ("ETA" OR "arrival time" OR simulation OR "state prediction")

2.2.3 Query Justification

- The main emphasis of QS1 is on the use of transformer or autoregressive models to forecast ETA, delays, or model network dynamics in public transportation. It provides a basis for understanding predictive architectures closely related to Large Events Models.
- By collecting studies incorporating spatiotemporal datasets (such as GPS, Automatic Vehicle Location (AVL), GTFS, or geospatial data) used for mobility modeling and forecasting, QS2 broadens the focus toward the data dimension.
- QS3 focuses on the sequence modeling perspective, identifying studies that apply time-series modeling and event-sequence prediction techniques to transport-related systems, particularly those that leverage Large Language Model (LLM)s, TimeGPT, or LEM designs.

These three queries cover the data, model, and sequence-learning elements essential to this dissertation.

2.2.4 Identification, Filtering, and Eligibility Processes

2.2.4.1 Identification Process

The combined execution of all three queries yielded approximately 2600 records across the selected databases:

Table 2.1: Number of retrieved papers per digital library

Source	Count
Scopus	993
IEEE Xplore	1402
ACM Digital Library	200

For management, every record was loaded into Zotero after being exported in BibTeX format. Duplicate detection, metadata organization, and source traceability were handled using Zotero.

2.2.4.2 Filtering Process

The filtering procedure followed a systematic multi-stage screening workflow consistent with established literature review practices, ensuring transparency and reproducibility in the selection process:

1. **Duplicate Removal:** Automated detection using Zotero.
2. **Publication Date:** Only works published from 2015 onward were retained.
3. **Language:** Publications written in languages other than English were not included.
4. **Full-Text Availability:** Records that could not be accessed in their entirety were eliminated.
5. **Publication Type:** Theses, workshops, and tutorials were not taken into account; only peer-reviewed journal articles and conference papers were.
6. **Relevance Screening:** A manual examination of titles and abstracts guaranteed a clear connection to event sequence modeling, simulation, spatio-temporal data, or public transportation prediction.

The final literature base was created by reducing the dataset to 42 eligible research works after these procedures. To improve the traceability of the selection process, the following table summarizes the screening stages and the number of records retained after each filtering step. This structured breakdown follows the multi-stage workflow described above and increases transparency and reproducibility in the creation of the final literature corpus.

Table 2.2: Screening and selection breakdown

Records identified	2595
After duplicate removal	2010
After publication year filter (> 2014)	1658
After language filter (English only)	1604
After full-text availability filter	972
After publication type filter (journal/conference only)	701
After title + abstract relevance screening	42

In the following sections, representative papers are cited across each major methodological family and across the main data and operational settings identified during screening, while the full corpus informed the synthesis presented in this chapter.

2.2.5 Inclusion and Exclusion Criteria

To ensure that only studies directly relevant to the objectives of this thesis were taken into account, inclusion and exclusion criteria were established. Research using spatio-temporal data sources such as GPS, AVL, or GTFS is included for tasks such as ETA estimation, delay prediction, and network simulation. These investigations focus on public transportation systems, transit networks,

or urban mobility. Included were works that used autoregressive, transformer-based, or large language models for event-based or sequential modeling.

Studies that had nothing to do with transportation contexts, were solely descriptive or policy-focused, did not use predictive modeling, and exclusively used static or rule-based approaches were excluded.

2.3 State of the Art

Historically, public transport prediction research has focused on applying statistical and machine learning approaches to spatiotemporal data to predict journey times, delays, and arrival times. The predominant corpus of work can be divided into four primary categories: network-aware or generative formulations, hybrid and ensemble techniques, deep learning-based spatiotemporal models, and classical statistical models.

2.3.1 Data sources, data quality, and operational constraints

Public transport prediction and simulation systems rely on heterogeneous data sources that capture different aspects of network dynamics. Scheduled service information (e.g., GTFS) provides planned stop sequences and timetables, while operational data streams such as GPS/AVL tracks represent real vehicle trajectories and delays. Demand-related signals, including smart card validations or Automatic Passenger Counter (APC) data, add information about passenger load and boarding patterns, which can strongly influence dwell times and operational variability [7, 29]. In practice, these sources are affected by missingness, irregular sampling, and noise, requiring robust preprocessing pipelines and modeling strategies that remain reliable under imperfect observations [25].

2.3.2 Classical Statistical and Time-Series Models

The early methods for predicting bus journey time and ETA relied on time-series and probabilistic formulations based on statistical presumptions. Temporal dependencies in bus journey durations have been extensively modeled using Autoregressive (AR), Autoregressive Integrated Moving Average (ARIMA), and seasonal models, often assuming logarithmic normal or Gaussian distributions [5].

Beyond point forecasting, classical approaches often model travel time as a probability distribution, which supports reliability-aware decision making and passenger-facing uncertainty communication. Distributional formulations and stochastic sequence models have been proposed to capture variability under heterogeneous operating conditions [4, 16, 19].

To address non-stationarity in transit operations, regime-switching formulations introduce latent operating modes (e.g., congestion vs. free-flow, typical vs. atypical conditions) and infer transitions between them over time. Such models provide a principled way to represent abrupt changes in travel time patterns that are not well captured by stationary assumptions [3].

Interpretability, minimal data requirements, and transparent estimation processes are advantages of these models. Bayesian extensions allowed for posterior inference over trip times or speeds across routes and segments by incorporating uncertainty and spatial correlation [8].

Statistical models have structural limitations despite their advantages. Usually, they call for explicit route segmentation, manual feature engineering, and stationarity assumptions. They are brittle in unusual circumstances, such as outages or significant public events, because of their limited ability to model non-linear interactions among traffic, demand, and network-wide dependencies.

2.3.3 Deep Learning for Spatio-Temporal ETA Prediction

Deep learning techniques gained popularity as high-frequency GPS and AVL data became available. The temporal dynamics of bus trajectories and arrival times have been widely modeled using Recurrent Neural Network (RNN)s, Long Short-Term Memory (LSTM)s, and Gated Recurrent Unit (GRU)s. Spatial correlations between adjacent stops or successive road segments are frequently encoded using convolutional layers.

By explicitly modeling transport networks as graphs, graph-based models significantly improved the state of the art. While temporal components capture sequential dynamics, Graph Convolutional Network (GCN)s and their variations depict spatial relationships between stops or road segments. In bus networks, multi-relational GCNs have shown increased accuracy in capturing upstream and downstream impacts [23]. These methods assume a fixed network topology and typically operate in synchronized time steps.

Deep spatiotemporal models are essentially discriminative, despite their superior predictive accuracy compared to classical approaches. Instead of simulating complete network dynamics, they are tailored for ETA point prediction. Their outputs are not directly compositional across diverse event types, such as passenger boarding, delays, or service disruptions, and their internal representations remain opaque.

2.3.3.1 Trajectory and segment sequence learning

With the availability of high-frequency GPS and AVL data, deep learning approaches have been widely applied to model the temporal evolution of vehicle trajectories and stop-level arrival times. A common formulation represents each trip as a sequence of stops or road segments and learns dependencies directly from the trajectory histories, often incorporating contextual signals such as traffic or time-of-day [12, 22]. Recent methods also learn joint representations of route structure and trajectory dynamics, improving generalization when only GPS trajectories are available [14].

2.3.3.2 Graph-based and network-aware spatio-temporal models

Graph-based models improve the prediction of ETA by representing transit systems as networks, where nodes correspond to stops (or road segments), and edges encode connectivity and spatial influence. Graph neural networks and attention-based extensions capture spatial correlations,

while temporal components model sequential dynamics, leading to improved performance under network-wide interactions [6, 17] — a pattern also observed in road-traffic graph prediction [13].

Despite their strong predictive accuracy, these approaches typically assume a fixed topology and operate over synchronized time steps, remaining primarily discriminative and output-specific (e.g., predicting ETA directly) rather than simulating heterogeneous operational events.

2.3.4 Hybrid and Ensemble Approaches

Several studies integrate statistical techniques with machine learning or ensemble numerous predictors to address the shortcomings of individual models. Hybrid methods use neural networks and clustering to capture regime-dependent transit time dynamics [32]. Ensemble learning methods aggregate predictions from heterogeneous models to improve robustness and reduce variance [33].

These methods increase system complexity without addressing a fundamental limitation: the lack of a uniform representation of transport dynamics, even as they boost accuracy and stability. Although events are not expressly treated as first-class modeling entities, they are handled implicitly as features.

Hybrid designs often combine a strong temporal predictor with explicit filtering or correction mechanisms to mitigate noise and cumulative error along routes, improving robustness in real-world deployments. For example, Kalman-filter-based augmentations and profile-similarity corrections can stabilize the prediction of travel times under irregular observations [26, 27]. Ensemble methods further aggregate heterogeneous predictors to improve generalization across regimes, capture complex seasonality, and reduce variance in operational settings [15, 31].

2.3.5 Route-Specific and Generative Modeling

Route-specific and generative views have been investigated in more recent studies. Some studies suggest training specific models for each route or service pattern rather than learning a single global predictor, demonstrating that route heterogeneity is a significant factor in ETA precision [21]. To improve uncertainty estimation, generative formulations aim to simulate the full distribution of journey times or arrivals rather than point estimates.

Although these approaches get closer to simulation, they remain limited to specific outputs, such as arrival times, and do not generalize to arbitrary event sequences across the network.

2.3.6 Event-Based and Sequence Modeling Approaches

By treating transport systems as discrete-event sequences rather than continuous time series, event-based modeling signifies a conceptual change. Arrivals, departures, delays, and exchanges are all represented as ordered events in this paradigm. Auto-regressive token-based models have demonstrated excellent performance in sequence prediction problems outside of transportation, particularly in natural language processing [1, 28].

Event-sequence modeling is still understudied in transportation research. The fine-grained structure of existing work is often lost when events are reduced to temporal bins or aggregated. Heterogeneous events can be successfully tokenized and modeled autoregressively, enabling simulation and counterfactual analysis, as demonstrated by the Large Events Model framework established in sports analytics [18].

Large public events can induce abrupt and spatially concentrated changes in mobility demand, creating distribution shifts that are difficult to capture with stationary forecasting models. Event-centric approaches have been proposed to model citywide crowd dynamics around major events, motivating representations that treat disruptions and demand surges as first-class events rather than weak contextual features [11].

Transformer-based architectures have also been adapted specifically for bus arrival time prediction, demonstrating that attention mechanisms can capture long-range dependencies across stops and contextual signals [10]. Nevertheless, most existing transformer approaches remain discriminative, predicting ETA directly, rather than learning a generative sequence of heterogeneous transport events. However, this paradigm has not yet been applied systematically to public transport networks.

2.3.7 Reliability and uncertainty-aware prediction

Public transport systems exhibit high variability due to traffic conditions, demand fluctuations, and operational disruptions. As a result, uncertainty-aware forecasting is increasingly relevant, since point predictions can be overconfident in atypical conditions. Approaches that explicitly model heteroscedasticity and prediction uncertainty provide more informative outputs for both passengers and operators, enabling reliability-aware planning and decision making [20].

2.3.8 Operational tasks beyond ETA

Beyond arrival time estimation, related research targets operational objectives such as dispatching decisions, passenger load forecasting, and bus bunching prediction. These tasks highlight that network dynamics depend not only on vehicle motion but also on passenger flow and operational interventions. This motivates modeling frameworks that can represent boardings, crowding, dispatch actions, and bunching states explicitly, which aligns with event-based and tokenized representations for simulation and prediction [7, 9, 29].

2.3.9 Summary of dominant modeling families

Table 2.3 summarizes the main methodological families identified in the reviewed literature, their typical inputs and outputs, and key limitations that motivate the event-centric approach proposed in this thesis.

Family	Typical inputs	Outputs	Key limitations
Classical time-series / probabilistic	GTFS + historical travel times	ETA / travel time (point or distribution)	Limited non-linearity; stationarity assumptions; weak handling of disruptions [3, 16, 19]
Trajectory deep learning	GPS/AVL sequences + context	ETA / stop arrival time	Discriminative objective; limited event heterogeneity [12, 14]
Graph-based spatio-temporal models	Stop/road graphs + GPS/AVL	ETA / travel time	Fixed topology; time-binning; not generative [6, 17]
Hybrid / ensemble	GPS/AVL + engineered features	ETA / travel time	Complexity; still feature-driven and output-specific [26, 31]
Uncertainty-aware prediction	GPS/AVL + historical variability	Intervals / calibrated uncertainty	Often added as a post-hoc layer; limited event simulation [20]
Event-centric / large-event modeling	Event streams + context	Event sequence simulation	Tokenization design and scalability remain open challenges [11, 18]

Table 2.3: Summary of modeling families for public transport prediction and simulation.

2.4 Critical Discussion and Research Gap

In the reviewed literature, three structural limitations are evident.

1. The majority of ETA prediction systems are task-specific and discriminative. They are not intended to replicate network evolution or produce believable future event sequences; instead, they are optimized for arrival time prediction. They are therefore incapable of supporting large-event simulation, stress testing, or scenario analysis.

2. Fixed representations like grids, graphs, or route segments are closely linked to spatial and temporal interdependence. Although these representations achieve high accuracy, they are rigid and make it challenging to add new event types or adapt to distributional drift without retraining significant portions of the model.

3. In transportation systems, events are handled implicitly. Instead of being explicitly described as events, operational choices, passenger demand, and disruptions are encoded as characteristics. This hinders compositional thinking and restricts the system-level interpretability of model outputs.

The lack of a cohesive, event-centric modeling paradigm for public transportation networks is the gap this dissertation addresses. This thesis attempts to describe heterogeneous transport events in a single sequence space by modifying Large Events Models using auto-regressive token-based architectures. This bridges the gap between dynamic network modeling and ETA forecasting by enabling simulation and prediction within a single model.

The suggested framework does not imply a tight temporal structure or a fixed output goal, unlike earlier methods. Instead, it provides a basis for scalable data-driven simulation of public transportation systems under typical circumstances and during major events by learning the conditional distribution of future network states given previous events.

Moreover, major events and operational disruptions induce distribution shifts that degrade static predictors, reinforcing the need for representations and training objectives that can adapt to non-stationary event streams [3, 11].

2.5 Summary

This chapter presented a systematic literature review and state-of-the-art analysis of methods for public transport prediction and simulation. A structured review methodology was used to identify, filter and analyze relevant scientific contributions, resulting in a curated corpus of 42 high-relevance publications covering spatio-temporal data modeling, ETA prediction, and event-sequence learning. The state-of-the-art discussion synthesises this corpus by focusing on representative works that best illustrate the dominant approaches and limitations identified in the literature.

The state-of-the-art analysis indicated a movement from classical statistical models to deep spatiotemporal and graph-based approaches, as well as hybrid and ensemble methods. These methods are primarily discriminative, task-specific, and dependent on fixed representations of transport networks, despite achieving high predictive accuracy in estimating arrival times. In the transportation sector, event-based and generative techniques remain rare and have not yet been systematically incorporated into network-level simulation frameworks.

This analysis revealed a glaring research gap: the lack of a single event-centric modeling framework that can support the simultaneous simulation and prediction of the public transportation network. This gap drives the investigation of auto-regressive token-based architectures and Large Events Models as the basis for the approach presented in the following chapters.

Chapter 3

Proposed Approach

This chapter describes the proposed approach to simulate and predict public transport network dynamics using LEMs. It covers: (i) the overall approach and rationale, and (ii) the experimental process and evaluation design. The chapter is written as a design plan; the implementation choices adopted are reported in Chapters 4 and 5.

3.1 Overview of the Proposed Approach

The goal of this dissertation is to adapt auto-regressive token-based sequence models to the domain of public transportation systems, enabling both simulation and prediction of network states. Rather than treating the transport system as a static graph or a set of independent time series, the proposed method represents the operational evolution of a transit network as a sequence of discrete events.

The key idea is to reformulate heterogeneous transport observations, such as scheduled stop sequences, vehicle GPS updates, observed arrivals and departures, passenger validation counts, and delay information, as a unified event stream. Each event is encoded as a token (or a structured token composed of multiple discrete elements), allowing a transformer-based auto-regressive model to learn the conditional distribution of future events given a history of past events.

This approach directly addresses the limitations identified in the state-of-the-art review, where most ETA predictors are discriminative and output-specific, and where disruptions and demand surges are often only weakly represented as contextual features. By contrast, an event-centric representation enables the model to simulate plausible network futures by repeatedly sampling the next event and feeding it back as input, thereby generating a complete roll-out of the network evolution.

3.1.1 Core Problem Restatement

Public transport networks exhibit complex dependencies between vehicle movement and traffic conditions, passenger demand and dwell times, network interactions such as delay propagation and bus bunching, and distribution shifts caused by disruptions and large events.

Traditional methods often model these components through fixed assumptions (e.g., constant dwell time, segment-based travel time, stationary regimes) and struggle when the system enters atypical operating conditions. This dissertation explores whether Large Events Models can provide a scalable alternative by learning these dependencies directly from historical event sequences.

3.1.2 Approach Summary

The proposed solution is composed of three major components:

1. **Data preprocessing and tokenization**: transform raw operational data (GTFS, GPS/AVL traces, passenger counts) into sequences of discrete event tokens.
2. **Large Events Model training**: train an auto-regressive model that predicts the next event token given the past context, enabling both prediction and simulation.
3. **Simulation visualization**: implement a demonstration interface capable of replaying and comparing simulated trajectories against observed ground truth.

Together, these components define a complete pipeline from raw data to simulation outputs, supporting evaluation both in terms of predictive accuracy (e.g., ETA error) and simulation realism (e.g., event plausibility and delay propagation patterns).

3.2 Event Representation and Tokenization Strategy

A central design decision is how to represent a public transport network as a token sequence. Unlike natural language, where tokens correspond to subwords, public transport requires encoding structured heterogeneous information: entities (vehicles, stops, routes), actions (arrival, departure, boarding), time, and potentially numerical values such as passenger counts.

3.2.1 Event Types

The proposed event vocabulary will include vehicle movement events (arrival at a stop, departure from a stop, and optionally passing intermediate points if the GPS density supports it). In addition, operational state events will represent system-level changes such as increases or decreases in delay, service interruptions when available, and derived aggregation indicators. Finally, passenger flow events will capture boarding counts, alighting counts when available, and inferred or measured occupancy changes.

In the minimal viable setting, the system will support stop-arrival events with timestamps, which already allow ETA prediction and network roll-outs; departure events would further enable dwell-time modeling but are treated as an extension rather than a requirement. Passenger-related events are expected to improve realism by enabling the model to learn the variability of dwell-time and effects of congestion.

3.2.2 Token Structure

Two alternative tokenization strategies will be explored:

3.2.2.1 Flat Token Vocabulary

In a flat vocabulary, each token represents a fully specified event such as:

`ARRIVAL_STOP_S123_VEH_V45`

This representation is simple, but scales poorly: the vocabulary grows combinatorially with the number of stops and vehicles, and generalization to unseen combinations becomes difficult.

3.2.2.2 Factorized (Compositional) Tokenization

In a factorized approach, each event is represented as a short sequence of tokens that describe the event attributes:

`[ARRIVAL, STOP=S123, VEH=V45, ROUTE=R2, TIMEBIN=T17]`

This approach is more scalable and allows sharing statistical strength between entities. It also resembles structured event modeling used in Large Events Models in other domains, where a single "event" is represented as a group of categorical variables.

In this dissertation, the factorized strategy will be prioritized due to its better scalability and compositionality.

3.2.3 Temporal Encoding

Time is a critical component for ETA prediction. However, raw timestamps are continuous and must be represented in a form suitable for token prediction.

The following time encodings will be evaluated:

- **Absolute time bins**: discretize time-of-day into fixed intervals (e.g., 1 minute or 5 minutes).
- **Delta time tokens**: encode the time elapsed since the previous event (quantized).
- **Hybrid encoding**: includes both absolute time bin and delta time, improving the ability to capture periodicity and short-term dynamics.

The hybrid approach is expected to yield the best results, as it supports both schedule-related periodic patterns and irregular delay propagation.

3.2.4 Handling Missingness and Noise

Real-world transport data are often incomplete and noisy. Typical issues include irregular GPS sampling, missing stop events, inconsistent vehicle identifiers, and partial availability of passenger counts. To mitigate these issues, the preprocessing pipeline will include stop-snapping or

map-matching heuristics to align GPS points with stops, filtering of impossible jumps using speed constraints, interpolation for small gaps, and explicit `MISSING` tokens when attributes are absent. Representing missingness explicitly is important to avoid biasing the event stream toward unrealistically clean trajectories.

3.3 Large Events Model Design

3.3.1 Model Objective

Let the transport system be represented as a sequence of tokens:

$$x_1, x_2, \ldots, x_T$$

The model is trained to maximize the likelihood of the next token given past context:

$$p(x_t \mid x_{<t})$$

This corresponds to standard auto-regressive training, where the loss is the negative log-likelihood:

$$\mathscr{L} = -\sum_{t=1}^{T} \log p(x_t \mid x_{<t})$$

For factorized events, the objective may be extended to predict multiple attributes conditioned on the same history, either sequentially (attribute tokens in order) or jointly (multi-head output).

3.3.2 Architecture Choice

The baseline architecture will be a transformer decoder, inspired by large language models. This choice is justified because transformers model long-range dependencies through attention, train efficiently compared to RNN-based alternatives, and are directly compatible with tokenized sequences. Two variants will be considered: a standard decoder-only transformer (Generative Pre-trained Transformer (GPT)-style) and a structured event transformer where event boundaries are explicitly modeled using delimiters or learned event embeddings. The standard decoder-only model will serve as the initial baseline due to its simplicity and reproducibility.

3.3.3 Context Window and Scalability

Public transport networks may generate large numbers of events per day. To keep training feasible, the event stream will be segmented into fixed length L tokens.

Key design considerations include:

- choosing L large enough to capture delay propagation across multiple stops,
- ensuring that segmentation does not break event boundaries,

- avoiding excessive memory use during training.

Sliding windows with overlap will be used to improve the coverage of long trajectories while preserving training efficiency.

3.3.4 Simulation via Auto-Regressive Roll-out

After training, simulation is performed by selecting an initial context sequence (for example, events from the beginning of a day), decoding the next token, appending it to the context, and repeating until the desired horizon is reached. This produces a synthetic event stream that can be interpreted back into network states such as predicted arrival and departure times, inferred delays, and simulated passenger loads if passenger-related events are included. To control realism and reduce degeneracy, decoding strategies such as top-k sampling, nucleus sampling, and temperature scaling will be tested.

3.4 Experimental Process and Evaluation Design

This section defines the experimental protocol used to validate the hypothesis of the dissertation and answer the research questions.

3.4.1 Research Design

The research design is primarily **experimental and quantitative**, complemented by qualitative visual inspection of rollout outputs through the visualization component.

The experiments are designed to evaluate:

- **tokenization quality** (RQ1),
- **sequence prediction and simulation performance** (RQ2),
- **comparison to baseline approaches** (RQ3).

3.4.2 Datasets and Data Splits

Training and evaluation data will be obtained from public transport operational records provided by the hosting institution. The dataset will be split temporally to avoid leakage, with an earlier period used for training, an intermediate period for validation, and a later period for testing. This setup better reflects deployment conditions, where models are trained on past data and evaluated on future unseen periods, including distribution drift.

3.4.3 Baselines

To contextualize performance, the proposed model will be compared against a schedule baseline derived from the GTFS timetable, a historical average baseline based on time-of-day travel time

statistics, a classical time-series baseline such as AR/ARIMA at stop or segment level, if feasible, and a simple deep learning baseline such as an LSTM or GRU trained on stop sequences. These baselines represent increasing complexity and allow quantifying the value added by event-centric transformer modeling.

3.4.4 Evaluation Metrics

The evaluation will focus on both **prediction accuracy** and **simulation realism**.

3.4.4.1 ETA Prediction Metrics

For stop-level arrival time prediction, the following metrics will be used:

- Mean Absolute Error (MAE),
- Root Mean Squared Error (RMSE),
- Mean Absolute Percentage Error (MAPE), when applicable.

The metrics will be computed per stop, per route, and aggregated throughout the network to identify where the performance gains are most significant.

3.4.4.2 Event Prediction Metrics

Since the model predicts tokens, sequence-level metrics will also be used. These include token-level accuracy, Negative Log-Likelihood (NLL) and perplexity, and top-k accuracy for categorical attributes. These metrics quantify how well the model learns the event distribution, independently of how it is translated into ETA.

3.4.4.3 Simulation Quality Metrics

Simulation evaluation is more challenging because multiple plausible futures may exist. The evaluation will therefore consider distributional similarity of headways and dwell times, correlation patterns in delay propagation between simulated and real data, and the frequency of invalid sequences such as departures occurring before arrivals. In addition, qualitative evaluation is performed by visual inspection of time–space diagrams, focusing on whether simulated bus trajectories appear realistic.

3.4.5 Ablation Studies

Ablation experiments will test the contribution of key design choices. In particular, experiments will compare vehicle-only event streams against tokenizations that include passenger flow, absolute time encoding against delta-time and hybrid encodings, stop-level event granularity against GPS-derived events with higher-resolution, and multiple context window sizes L. These ablations directly support RQ1 by evaluating whether richer tokenization improves performance.

3.4.6 Risks

Several threats to validity are anticipated. Data quality issues such as missing or noisy GPS may introduce systematic errors, while large events or disruptions may be underrepresented in training data and cause distribution shifts. Simulation evaluation is inherently ambiguous because multiple futures may be plausible and the model may also overfit to specific routes, time-of-day patterns, or operator behaviors. These risks will be mitigated through robust preprocessing, temporal splits, and evaluation across multiple routes and operating regimes.

3.5 Summary

This chapter presented the proposed approach for simulating and predicting public transport networks using Large Events Models. The solution reformulates transport operations as tokenized event sequences, enabling transformer-based auto-regressive models to learn complex temporal and causal dependencies directly from historical data. An experimental design was defined to evaluate tokenization quality, prediction performance, and simulation realism, with comparisons with classical and deep learning baselines.

Chapter 4

Implementation

This chapter describes the complete implementation pipeline developed to instantiate the approach outlined in Chapter 3. The pipeline comprises three major stages: (i) data acquisition and preprocessing, (ii) event tokenization, and (iii) model design, training, and simulation. All components were implemented in Python 3.13, using PyTorch 2 for the neural model, pandas for tabular data processing, and the PyArrow/Parquet format for efficient storage of large sequence datasets.

Figure 4.1 shows the general architecture of the pipeline. Three separate sources of raw operational data are preprocessed to create a canonical event table, which is subsequently tokenized into trip-level sequences that the Large Events Model uses. The trained model supports two inference strategies: GTFS-guided generation for delay forecasting and autoregressive rollout for trip-level event-sequence generation.

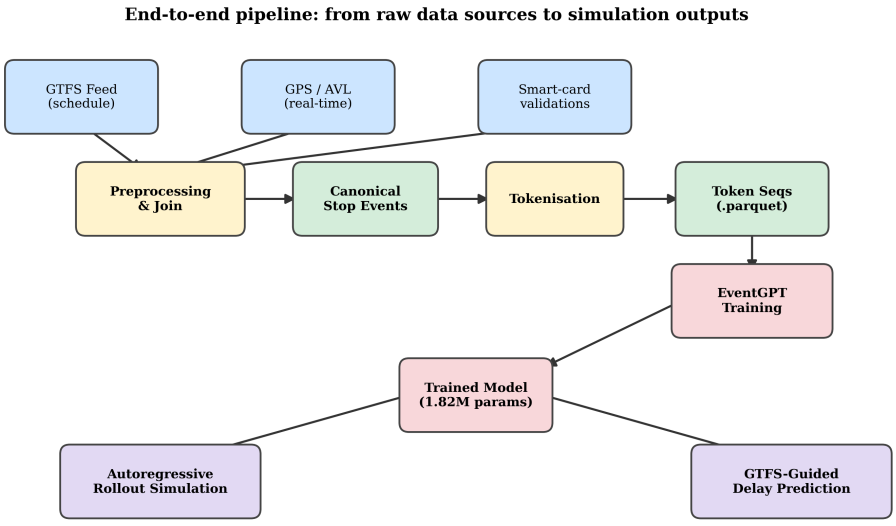

Figure 4.1: End-to-end data and modeling pipeline. Three heterogeneous data sources are joined into a canonical event table, tokenized into trip sequences, and used to train the EventGPT model. The trained model supports both unconstrained rollout simulation and GTFS-guided delay prediction.

4.1 Data Acquisition and Preprocessing

4.1.1 Data Sources

The dataset, which covers the Vianorbus-operated public bus network in the Porto, Portugal metropolitan area, was supplied by the hosting organization, Ubirider. Additionally, a supplementary multi-operator experiment employed a secondary dataset from two smaller regional operators: Próximo (Faro municipality) and Beja (Beja municipality). The main inputs were three complementary data streams.

GTFS feed. The static timetable for every bus route is provided by the GTFS. It contains `trips.txt`, `stop_times.txt`, `stops.txt`, `routes.txt`, and `calendar.txt`. These are used to create a `planned_stop_events` table that associates each (trip, stop sequence position) pair with the route identifier, stop geographic coordinates, and planned scheduled arrival and departure timings in seconds since local midnight. In order to calculate delays, real-time arrivals are compared to this table, which acts as the ground-truth schedule.

GPS/AVL real-time updates. Vehicle position events are streamed from the on-board GPS tracking system and exported in JavaScript Object Notation (JSON) format. Each record carries the trip identifier, stop sequence number, realtime Unix timestamp (in milliseconds), and, when available, the operator-computed delay in seconds. A dedicated parsing script converts these records into a tabular representation, converting timestamps from Unix milliseconds to UTC datetimes and subsequently to local seconds-of-day using the Europe/Lisbon timezone. This step requires careful handling of timezone offsets and daylight-saving transitions.

Passenger validation counts. Smart-card boarding validations are provided as a CSV file aggregated per (vehicle service, stop, stop sequence position), covering the full three-month period (January–March 2026).

The `totalValidations` field records the number of passenger boardings at each stop event. Missing values, arising from stops where no validations occurred or where ticketing data was unavailable, are filled with zero in the preprocessing step.

4.1.2 Construction of Observed Stop Events

The core preprocessing step merges the three sources into a canonical table of *observed stop events*. The pipeline proceeds as follows:

1. **Timestamp conversion**: GPS/AVL records are converted from Unix milliseconds to UTC datetimes. The arrival in real time is then expressed as seconds since local midnight (`realtime_sec`), allowing a direct arithmetic comparison with the GTFS schedule.
2. **GTFS join**: each GPS/AVL update is left-joined to the planned schedule on the composite key `(trip_id, stopSequence)`, producing a row that carries both the observed arrival

time and the corresponding planned arrival time. Rows where no matching GTFS record is found are flagged and retained with null scheduled values. In the Vianorbus dataset, GTFS coverage was complete: all 5,371,754 stop events were successfully matched and no null scheduled arrival values remain in the final canonical table.

3. **Delay computation**: delay is defined as the difference between the observed realtime arrival and the scheduled arrival from GTFS, in seconds:

$$\texttt{delay\_sec} = \texttt{realtime\_sec} - \texttt{scheduled\_arrival\_sec}$$

When the real-time feed carries its own operator-computed delay field, that value is preferred, otherwise the delay is derived from the GTFS join. Negative values indicate early arrivals.

4. **Deduplication**: when multiple GPS/AVL updates are recorded for the same (service date, trip, stop sequence) tuple, which can occur due to repeated transmissions or tracking artifacts, only the most recent record is retained.

5. **Validation join**: passenger validation counts are merged on the `(vehicle_id, stop_id, stopSequence)` key. The vehicle identifier is normalized by stripping a trailing suffix appended by the real-time feed.

The resulting canonical table contains, for each observed stop event: service date, trip identifier, route identifier, vehicle identifier, stop identifier, stop sequence number, observed realtime arrival (UTC datetime and seconds-of-day), scheduled arrival and departure times, delay in seconds, stop geographic coordinates, and total passenger validations. Table 4.1 shows a representative excerpt.

Table 4.1: Representative excerpt from the canonical `observed_stop_events` table (trip `A-DF:5001:0:1:1015`, service date 2026-01-01). Realtime and scheduled times are expressed in HH:MM:SS (seconds since local midnight). Delay is in seconds, positive values indicate late arrivals. All five stops belong to the same trip, progressing through consecutive stop sequence positions.

Seq	Stop ID	Realtime	Scheduled	Delay (s)	Valid.
0	`prg:mat:839`	10:15:32	10:15:00	+32	0
1	`prg:mat:200`	10:16:02	10:15:31	+31	0
2	`prg:mat:24`	10:16:34	10:16:00	+34	0
3	`prg:mat:34`	10:17:09	10:17:00	+9	0
4	`prg:mat:245`	10:18:00	10:17:31	+29	0

4.1.3 Data Quality Control

The canonical preprocessing pipeline incorporates several data quality checks designed to detect and handle common artifacts in real-world GPS/AVL data before they propagate into tokenized sequences.

Impossible delay values. A small proportion of GPS records, resulting from timestamp errors, network transmission delays, or vehicle clock miscalibration, produce implausible delay values after the GTFS join. Events where the computed delay exceeds a threshold of ± 120 minutes are flagged as likely artifacts and assigned a `Delay Bin (DBIN) = NA` token rather than a clamped extreme value. In the Vianorbus dataset, 2,579 stop events (0.048% of all events) were affected, confirming that this is a rare but non-negligible artifact class. This treatment is preferable to clamping, which would assign the maximum delay bin (DBIN = 60) to genuinely erroneous records, biasing the model's representation of the maximum 30-minute delay bin toward noise rather than real high-delay events.

Temporal ordering within trips. Within a correctly recorded trip, the GPS updates should increase monotonically in timestamp order as the vehicle progresses through the stop sequence. The pipeline checks that the observed arrival timestamps within each (service_date, trip_id) group are non-decreasing with stop sequence number and flags reversals, which can arise from out-of-order packet delivery or GPS position jitter, for exclusion. In the Vianorbus dataset, approximately 0.3% of stop events were flagged for temporal ordering violations, consistent with the low but non-zero rate of GPS artifacts typical of real operational data.

GTFS schedule consistency. The scheduled arrival times from the GTFS feed are expected to be consistent with the service calendar (i.e., a trip should only be matched to a service date on which it is scheduled to operate according to `calendar.txt`). In practice, 0.56% of stop events (29,858 of 5,371,754) carry a null GPS/AVL timestamp (`realtime_sec` not recorded), which prevents the delay and arrival-time computation. These events are retained in the canonical table with null `delay_sec` and null `realtime_sec`, and are encoded as `TBIN=NA` / `DBIN=NA` by the tokeniser, ensuring they do not introduce erroneous delay signals while still contributing their stop-sequence context to the model.

Validation count consistency. Passenger validation counts are expected to be non-negative integers. The rare occurrence of negative counts, arising from smart-card system synchronization errors, is handled by clamping to zero before binning. Counts exceeding the maximum representable value (50 boardings, corresponding to Passenger Validation Bin (VALBIN) = 10) are also clamped to the maximum bin, consistent with the VALBIN discretization design described in Section 4.2.2. The prevalence of these edge cases is low (a little less than 0.1% of stop events), and their impact on model training is negligible.

4.1.4 Dataset Characterisation

The Vianorbus dataset spans 82 days of operations from 1 January to 23 March 2026. After deduplication and quality filtering, the canonical table contains 5,371,754 observed stop events distributed across 94 GTFS route variants and 2,150 unique stops. Table 4.2 summarizes the key dataset statistics.

Table 4.2: Vianorbus dataset summary statistics.

Attribute	**Value**
Coverage period	1 Jan – 23 Mar 2026 (82 days)
Bus operator	Vianorbus (Porto)
Total observed stop events	5,371,754
Unique GTFS route variants	94
Unique stops	2,150
Training trips (Jan 1 – Feb 5)	48,931
Validation trips (Feb 6 – Feb 28)	30,620
Test set trips (Mar 1 – Mar 23)	31,862
of which rollout-eligible ($\geq$14 stops)	30,814
Null GPS timestamp (TBIN/DBIN = NA)	$\approx$0.56% of stop events

Sequence lengths. Tokenizing trips into sequences yields a distribution of sequence lengths with a mean of approximately 390 tokens, a median of 404 tokens, and a maximum of 876 tokens. The mean (390) and median (404) are close, within 3.5% of each other, consistent with the near-symmetric bimodal shape of the distribution: short urban journeys with fewer than ten scheduled stops anchor the lower mode, long-haul routes with more than 60 stops anchor the upper mode, and the maximum sequence length reaches 876 tokens. Approximately 77% of sequences are longer than the model's fixed context window of 256 tokens, which is managed using sliding window segmentation during training to guarantee that every observation contributes to the training objective. Figure 4.2 shows the full distribution.

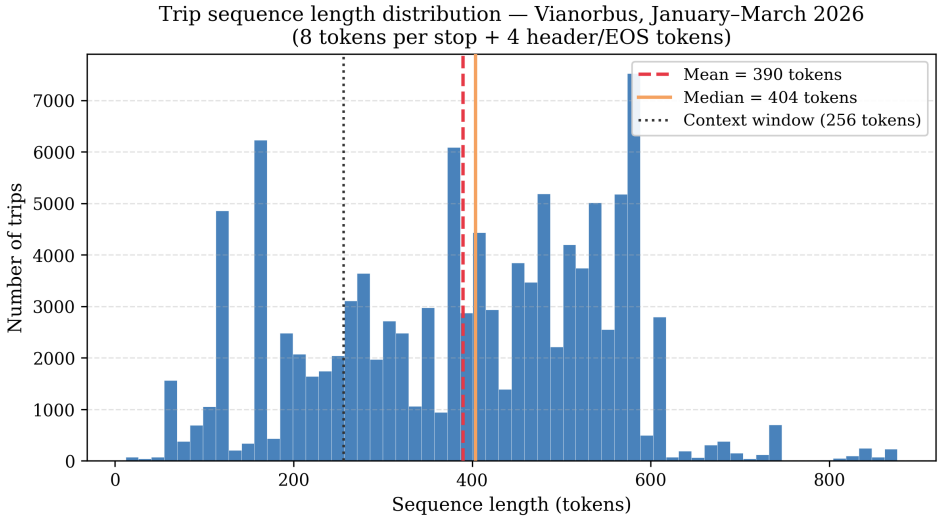

Figure 4.2: Trip sequence length distribution (8 tokens per stop event + 4 header/EOS tokens). The bimodal shape reflects the coexistence of short urban routes and long-haul routes. The dotted line marks the 256-token context window used during training.

4.1.5 Data Split

The dataset is partitioned by calendar date to replicate real-world deployment conditions, where models are trained on historical observations and evaluated strictly on future periods. Three non-overlapping sets are defined:

- **Training set**: January 1 – February 5, 2026 (36 days), used to update model parameters.
- **Validation set**: February 6 – February 28, 2026 (23 days), used for early stopping and hyperparameter selection.
- **Test set**: March 1 – March 23, 2026 (23 days), held out for final evaluation.

This date-based split ensures that the test set contains genuinely unseen future trips, avoiding the data leakage that would arise from a random partition across service dates. The test set contains 31,862 unique trips, of which 30,814 are rollout eligible ($\geq$14 stops). All baseline models use the same test period.

4.1.6 Exploratory Data Analysis

Before tokenization, an exploratory data analysis was conducted to characterize the statistical properties of the canonical event table and inform key design decisions in the tokenization and modeling pipeline. The analysis focuses on three primary dimensions: delay distribution, intra-trip delay structure, and temporal service patterns.

Delay distribution and discretization range. Across the entire 5,371,754-event Vianorbus dataset, the observed delay distribution is leptokurtic and right-skewed, with a mean absolute delay of 5.4 minutes and a median signed delay of approximately 2.2 minutes (i.e., the typical bus arrives just over two minutes late). About 10.6% of stop events fall within $\pm$30 seconds of the scheduled time, 38.9% within $\pm$2 minutes, and 53.0% within $\pm$3 minutes. These figures are consistent: the signed median lying inside the $\pm$2-minute window does not imply that 50% of events fall within it, because 8.7% of events are early arrivals of more than 2 minutes and 52.4% are late by more than 2 minutes, leaving approximately 39% in the central $\pm$2-minute band. The distribution reflects a network where buses are frequently late rather than punctual. The operational characteristics of a scheduled bus network, where drivers are typically ordered not to depart ahead of time from timing locations, are consistent with the lower frequency of early arrivals compared to late arrivals. Approximately 1.1% of events fall beyond the $[-30, +30]$ minute DBIN clamping range, and the 99th percentile of the observed delay magnitude is approximately 31 minutes. The choice of a 60-minute total range for DBIN (encoded at 30-second granularity, yielding 121 numeric bins) is supported by this analysis: the range covers over 98.9% of observations while the 30-second bin width provides sub-minute resolution for the bulk of the delay distribution. Figure 4.3 shows the delay distribution across all stop events.

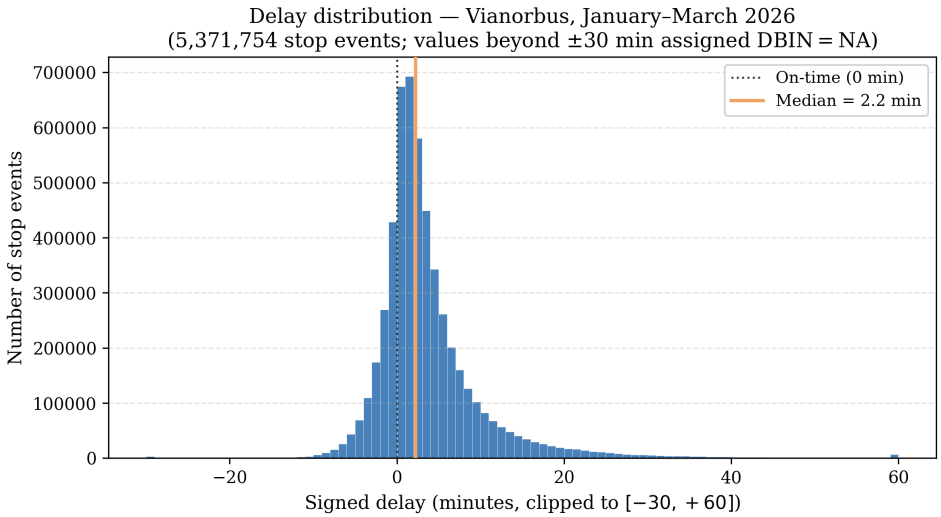

Figure 4.3: Signed delay distribution across 5,371,754 stop events (Vianorbus, January–March 2026), clipped to $[-30, +60]$ minutes for display. The median of approximately 2.2 minutes and the long positive tail confirm a network operating predominantly late. The ± 120-minute clamping threshold used to assign `DBIN=NA` lies well outside this range.

Intra-trip delay autocorrelation. The significant positive serial correlation of delays within a single trip is a crucial feature that the autoregressive model takes advantage of. The physical truth that a bus steadily accumulates or recovers the delay as it moves along its route is reflected in the high degree of predictability between the delay at stop k and the delay at stop $k+1$ during the same journey. The transformer's causal self-attention is specifically built to capture this autocorrelation structure: the model can efficiently condition on the observed delay trajectory and extrapolate it forward by paying attention to the complete delay history encoded in the preceding DBIN tokens. Abrupt disruptions, where the delay changes sharply between consecutive stops, represent the most challenging cases for the model, as they correspond to low-autocorrelation events that are difficult to anticipate without external contextual signals (e.g., real-time traffic data). This interpretation is consistent with the failure case documented in Table 5.8 and motivates the future extension of the vocabulary discussed in Section 6.3.

Temporal service patterns. The distribution of stop events across the time-of-day exhibits the expected bimodal structure corresponding to the morning and evening peak periods, as shown in Figure 4.4. Service during shoulder hours (early morning and late evening) is sparse, corresponding to the first and last trips of the day. This temporal structure motivates the inclusion of the Time Bin (TBIN) token in the event representation: without explicit time-of-day encoding, the model would be unable to distinguish peak-hour congestion patterns, where delay variability is highest, from the more regular off-peak regime. The DAYTYPE token provides complementary information by distinguishing weekday, Saturday, and Sunday services, each of which exhibits distinct demand and delay profiles. Together, TBIN and DAYTYPE provide the model with sufficient

temporal context to condition its predictions on the broad operating regime, even in the absence of route-specific features.

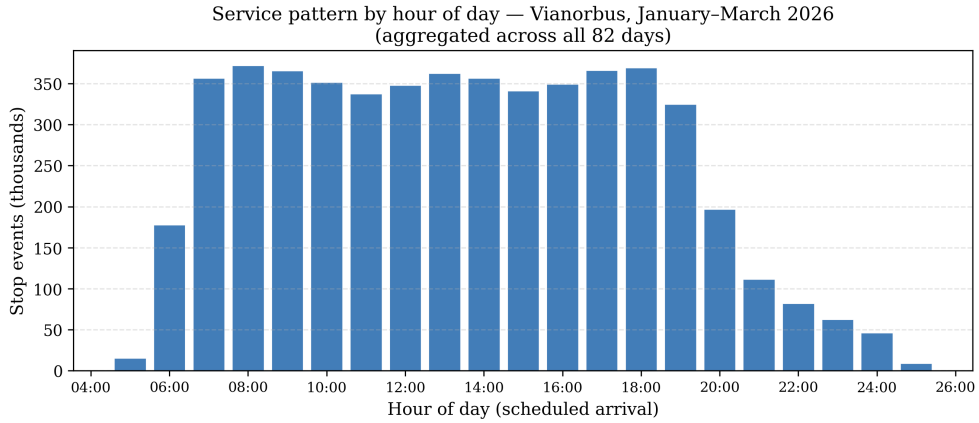

Figure 4.4: Distribution of stop events by hour of day (scheduled arrival time), aggregated across all 82 days of the Vianorbus dataset. The morning peak (around 08:00) and evening peak (around 18:00) are clearly visible, with sparse service before 06:00 and after 22:00.

Route-level heterogeneity. The Vianorbus network encompasses 94 GTFS route identifiers (including directional and service-variant encodings), exhibiting substantial heterogeneity in length, frequency of service, and operational variability. Route lengths range from short urban services with fewer than ten scheduled stops to long-haul regional routes with more than 60 stops, producing a wide distribution of sequence lengths in the tokenized dataset. Routes traversing the urban center of Porto are subject to higher congestion-induced delay variability compared to rural routes operating on lightly trafficked roads. This heterogeneity has direct implications for model evaluation: aggregate metrics may mask significant differences per-route in predictive performance, motivating the per-route breakdown presented in Section 5.3. Moreover, routes with fewer training examples, due to lower service frequency, receive proportionally fewer gradient updates per route–stop combination, which may contribute to the higher prediction error observed on low-frequency routes in the test set.

4.2 Event tokenization

4.2.1 Design Rationale

A central design decision is how to represent a public transport network as a token sequence that can be consumed by an autoregressive model. Unlike natural language, where tokenization decomposes text into subword units sharing a common embedding space, public transport data are inherently structured: each event involves multiple independent attributes (stop identity, arrival time, delay, passenger count) that should be represented distinctly to enable the model to learn attribute-specific patterns.

As established in Chapter 3, a factorized (compositional) vocabulary was adopted over a flat vocabulary: each event is a short sequence of attribute tokens rather than a single atomic token per event type, allowing attribute embeddings to be shared across different stop–time–delay combinations. The factorized strategy follows the design precedent established in the Large Events Model for soccer [18].

4.2.2 Token Representation

Each observed stop event is encoded as an *event block* of eight tokens in fixed order, the last of which is an explicit event-boundary delimiter:

```
EVT=STOP_UPDATE  STOPSEQ=N  STOP=S
TBIN=T  DBIN=D  VALBIN=V  HDWAY=H  <E>
```

The role of each field is as follows:

- **EVT**: event type identifier. In the current vocabulary, the only type is `STOP_UPDATE`, representing a vehicle arrival at a scheduled stop. The field is retained explicitly to enable future extension of the vocabulary with additional event types (e.g., departures, service disruptions).
- **STOPSEQ**: the integer stop-sequence position within the trip (e.g., `STOPSEQ = 7`), drawn directly from the GTFS `stop_times.txt` field. This token conveys where on the route the vehicle is, independently of the stop's geographic identity.
- **STOP**: the GTFS stop identifier string (e.g., `STOP = prg:mat:579`). This provides the geographic and semantic context of the stop.
- **TBIN**: the GTFS scheduled arrival time, discretized into 60-second bins. The bin t represents the interval $[60t, 60(t+1))$ seconds since local midnight, spanning values 0 to 1,522 to accommodate after-midnight trips coded beyond 24:00 in GTFS. Using the scheduled rather than the actual arrival time as the TBIN key eliminates distribution shift between training and evaluation: the model is always conditioned on the planned schedule, and predicted delays are interpreted as offsets from it.
- **DBIN**: the delay relative to the planned arrival time, discretized into 30-second bins and clamped to the range $[-30, +30]$ minutes, yielding 121 discrete values (bins -60 through $+60$). Negative bins indicate early arrivals. The 30-second resolution provides sub-minute precision while keeping the vocabulary compact, and the 60-minute symmetric range covers 98.9% of observed stop events.
- **VALBIN**: the total number of passenger boardings at the stop, discretized using a bin width of 5 and capped at 50 validations (11 discrete values: 0 to 10). VALBIN reflects the demand signal at the stop.

- **Headway Bin (HDWAY)**: the scheduled headway to the preceding vehicle on the same route, discretized into 5-minute bins (values 0 to 24, corresponding to 0 to 120 minutes), plus an `HDWAY=NA` token for the first trip of the day or when no preceding vehicle is scheduled. HDWAY encodes the inter-vehicle spacing at the network level: low values indicate tight headways where bus bunching is likely, while high values indicate isolated trips with no preceding vehicle. By conditioning on HDWAY, the model can distinguish delay dynamics arising from inherited congestion from those caused by independent factors such as dwell time or local traffic.

- `<E>`: an explicit event-boundary delimiter token. Its presence at a fixed position within the event block enables unambiguous decoding: the model knows exactly when one event ends and the next begins, simplifying schema-constrained generation.

A concrete example of tokenization for a single observed stop event is shown in Listing 4.1.

Listing 4.1: Factorized token representation of a stop event. The bus is scheduled to arrive at stop `prg:mat:605` (sequence position 7) at 14:22 (TBIN=862, i.e. 862 minutes past midnight in the GTFS schedule), with a 30-second delay (DBIN=1 at 30 s/bin), 5–9 passenger boardings (VALBIN=1, bin width 5), and a preceding vehicle 10 minutes ahead (HDWAY=2 at 5 min/bin).

```
EVT=STOP_UPDATE  STOPSEQ=7  STOP=prg:mat:605
TBIN=862  DBIN=1  VALBIN=1  HDWAY=2  <E>
```

Each trip is encoded as a complete sequence beginning with three header tokens:

`<BOS>  ROUTE=R  DAYTYPE = { WD | SAT | SUN }`

and terminated with `<EOS>`. The `DAYTYPE` token distinguishes weekday, Saturday, and Sunday service patterns without exposing raw dates that the model should not need to memorize.

Figure 4.5 illustrates the tokenization of a complete trip.

4.2.3 Vocabulary Construction

The vocabulary is constructed in two phases. A *deterministic base vocabulary* is pre-populated with all structurally defined tokens:

- Five special tokens: `<PAD>`, `<BOS>`, `<EOS>`, `<E>`, `<UNK>`.

- 1,524 TBIN tokens in total: `TBIN = 0` through `TBIN = 1522` (1,523 numeric bins, covering scheduled arrivals up to 25:22 in GTFS representation) plus one `TBIN = NA` token.

- 122 DBIN tokens: 121 numeric bins (`DBIN = -60` through `DBIN = 60`, at 30 s per bin) plus one `DBIN = NA` token.

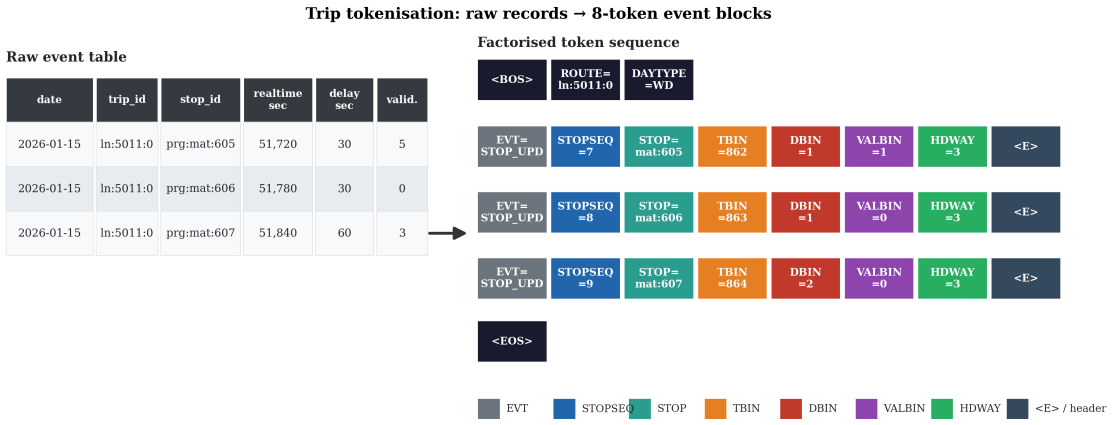

Figure 4.5: Conversion of a real bus trip into a factorized token sequence. Each row in the canonical event table becomes an eight-token block (colour-coded by field type). The sequence is prefixed with context header tokens (<BOS>, ROUTE=, DAYTYPE=) and terminated with <EOS>.

- 11 VALBIN tokens (`VALBIN = 0` through `VALBIN = 10`) plus `VALBIN = NA`.
- 25 HDWAY tokens (`HDWAY = 0` through `HDWAY = 24`, at 5 min per bin) plus `HDWAY = NA`.
- Three DAYTYPE tokens and one EVT token.
- Two `NA` placeholder tokens: `STOP=NA` and `STOPSEQ=NA`, for missing stop identity and stop-sequence position respectively.

A data-driven extension then adds stop identifiers, route identifiers, and stop-sequence position integers observed in the training sequences. The <UNK> token handles any stop or route encountered at test time that was absent during training. The final vocabulary for the single-operator (Vianorbus) model contains 4,049 unique tokens. The implementation of the core tokenization function is shown in Listing 4.2.

Listing 4.2: Core tokenization function. Each row from the canonical stop-events table is converted into a list of string tokens ready for vocabulary lookup.

```
def make_event_tokens(row):
    tbin = bin_int(row["scheduled_arrival_sec"], TIME_BIN_SEC)
    dbin = bin_int(row["delay_sec"],
                   DELAY_BIN_SEC,
                   min_v=-MAX_DELAY_MIN * 60,
                   max_v= MAX_DELAY_MIN * 60)
    vbin = bin_int(row["totalValidations"],
                   VAL_BIN_SIZE, min_v=0, max_v=MAX_VAL)
    hbin = bin_int(row["headway_sec"],
```

```
                    HEADWAY_BIN_SEC, min_v=0, max_v=MAX_HEADWAY_MIN * 60)
    return [
        "EVT=STOP_UPDATE",
        f"STOPSEQ={int(row['stopSequence'])}",
        f"STOP={row['stop_id']}",
        f"TBIN={tbin  if tbin  is not None else 'NA'}",
        f"DBIN={dbin  if dbin  is not None else 'NA'}",
        f"VALBIN={vbin if vbin is not None else 'NA'}",
        f"HDWAY={hbin if hbin is not None else 'NA'}",
        "<E>",    # event boundary delimiter
    ]
```

4.2.4 Sequence Statistics

The tokenization produces one sequence per (service_date, trip_id) pair. A trip with n observed stops generates a sequence of $3+8n+1$ tokens (3 header tokens + $8n$ event tokens + 1 EOS). The Vianorbus tokenized dataset contains 111,413 sequences. The mean sequence length is approximately 390 tokens and the median is 404 tokens (mean $<$ median, reflecting the lower mode of the bimodal distribution driven by short-route sequences), with long-haul routes reaching up to 876 tokens. Sequences longer than the model's context window of 256 tokens are handled by sliding window segmentation during training, as described in Section 4.3.3.

4.2.5 Vocabulary Coverage and Token Frequency Analysis

Understanding the statistical properties of the constructed vocabulary is important both for assessing tokenization quality and for diagnosing potential failure modes at inference time.

Vocabulary composition. The 4,049-token vocabulary of the single-operator model comprises two functionally distinct groups. Structurally fixed tokens, including the 1,524 TBIN tokens (covering scheduled arrivals from 0 to 1,522 minutes, plus one `NA` entry), 122 DBIN tokens (121 numeric bins from -60 to $+60$ at 30 s each, plus one `NA`), 26 HDWAY tokens (0 to 24 bins at 5 min each, plus `NA`), 12 VALBIN tokens, five special tokens (`<PAD>`, `<BOS>`, `<EOS>`, `<E>`, `<UNK>`), the `EVT` and `DAYTYPE` tokens, and two `NA` placeholder tokens for missing stop identity (`STOP=NA`) and stop-sequence position (`STOPSEQ=NA`), are defined deterministically irrespective of the training data. Data-driven tokens, built from the identifiers observed in training sequences, account for the remaining entries: 2,150 unique stop identifiers (GTFS `stop_id` values, which encode directional platforms and service variants), 94 route variant identifiers (GTFS route and direction combinations), and stop-sequence position integers observed across all training trips. This two-phase construction ensures that the structural tokens are stable across datasets while only the network-specific identifiers vary between operators.

Token frequency distribution. The frequency distribution of tokens in the training corpus is highly non-uniform across fields. TBIN tokens follow a multimodal distribution reflecting the daily service pattern: bins corresponding to the morning and evening rush hours are substantially more frequent than bins outside of operational hours, which by construction contain no events. DBIN tokens are concentrated in the low-positive-delay region: DBIN = 0 (on time, delay near 0 seconds) is the single most frequent bin at 13.1%, followed closely by DBIN = 1 (approximately 30 seconds late, 13.0%) and DBIN = 2 (approximately 60 seconds late, 10.7%). This reflects the leptokurtic, right-skewed delay distribution characterized in Section 4.1.6: a concentration of arrivals in the 0–2-minute-late range with a long right tail that pulls the mean absolute delay to 5.4 minutes. Because no single bin dominates, the most-frequent-bin concern that applies to VALBIN does not apply to DBIN. Token-level DBIN accuracy is discussed in Section 5.4.

The frequency distribution of stop identifier tokens is power-law-like: stops at the network's periphery may appear in only a small percentage of trips, whereas the most frequent stops, which correspond to major interchange points, terminuses, and stops served by multiple routes, appear in a large proportion of training sequences. Because of this imbalance, the model receives significantly less gradient updates for rare stop embeddings than for common ones, which could result in less accurate representations for peripheral stops. The per-parameter adaptive learning rate of the AdamW optimizer partially compensates for the asymmetry in update frequency in the current implementation, where all token embeddings are initialized from a scaled normal distribution.

Out-of-vocabulary handling at inference. In the closed Vianorbus test set, no out-of-vocabulary stop identifiers were encountered, reflecting the stable network topology over the evaluation period. The `<UNK>` token is primarily relevant for two scenarios: a multi-operator setting where stop identifiers from other operators are absent from the Vianorbus-only vocabulary, and future deployment scenarios where network changes introduce new stops between model training and inference. In both cases, the `<UNK>` token provides graceful degradation: the model can still generate a structurally valid prediction by treating the unknown stop as a generic entity, albeit with reduced specificity.

Relationship between TBIN and DBIN. TBIN and DBIN encode distinct and complementary aspects of each arrival event. TBIN is the GTFS scheduled arrival time, determined entirely by the published timetable and independent of actual vehicle operation. DBIN is the signed delay, computed as $(\texttt{actual\_arrival\_sec} - \texttt{scheduled\_arrival\_sec})/30$ and binned to the nearest 30-second interval. The two fields are not algebraically redundant: TBIN conveys which scheduled slot the event belongs to while DBIN conveys the operational deviation, and neither determines the other in isolation. DBIN is retained as an explicit prediction target because the delay is the operationally relevant quantity for passenger information and schedule-adherence monitoring. Although conceptually independent, the two fields carry statistical correlation because TBIN encodes time of day and delays in the Vianorbus network are time-of-day-dependent, typically higher during peak service hours.

A data quality note on the delay source: for 95.9% of events with valid GPS timestamps (99.44% of all stop events), the independently derived delay agrees to within 1 second with the operator-reported delay field, confirming source consistency. The remaining 4.1% carry anomalous operator-reported values typically exceeding two hours, consistent with GPS clock errors or identifier mismatches, these exceed the ±120-minute `DBIN=NA` threshold and are assigned `DBIN=NA` rather than a numeric bin, so they do not corrupt the delay token distribution. The 0.56% of stop events with null GPS timestamps are similarly assigned `DBIN=NA` and are excluded from this breakdown.

4.3 Large Events Model: Architecture and Training

4.3.1 Model Architecture

The proposed model, named EventGPT, is a decoder-only transformer following the GPT architecture [24, 28]. The choice of a decoder-only design is motivated by three considerations: (i) the autoregressive generation property is naturally embedded in the causal masking of the self-attention layer, (ii) the same model can be used for both density estimation and generation and (iii) the architecture has established training recipes and strong empirical performance across sequential modeling domains.

The architecture, illustrated in Figure 4.6, consists of the following components:

- **Token embedding**: a learnable matrix $\mathbf{E} \in \mathbb{R}^{V \times d}$ maps each integer token identifier to a dense d-dimensional vector, where V is the vocabulary size and $d = d_{\text{model}}$.
- **Positional embedding**: a learnable position embedding matrix $\mathbf{P} \in \mathbb{R}^{L \times d}$ is added element-wise to the token embeddings, where L is the context length. Learned (rather than sinusoidal) positional encodings were used, as they have shown competitive performance in GPT-style decoder-only language models [24].
- **Transformer blocks**: N identical blocks, each applying Pre-LayerNorm, causal multi-head self-attention with a residual connection, another Pre-LayerNorm, and a position-wise Feed-Forward Network (FFN) with expansion factor 4 and Gaussian Error Linear Unit (GELU) activation, followed by a second residual connection. The causal attention mask restricts each position to attend only to preceding positions, enforcing the autoregressive property.

  In each block, the multi-head self-attention computes, for each of the n_{heads} attention heads h:

$$\text{Attn}_h(\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h) = \text{softmax}\left(\frac{\mathbf{Q}_h \mathbf{K}_h^\top}{\sqrt{d_k}} + \mathbf{M}\right)\mathbf{V}_h,$$

  where $d_k = d_{\text{model}}/n_{\text{heads}} = 32$ is the per-head dimension, and $\mathbf{M}$ is the causal mask (setting upper-triangular entries to $-\infty$ before softmax). The outputs of all heads are concatenated and projected back to d_{model} dimensions. The FFN applies two linear transformations with a GELU activation in between: $\text{FFN}(\mathbf{x}) = \mathbf{W}_2\,\text{GELU}(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2$, where

$\mathbf{W}_1 \in \mathbb{R}^{d_{\text{model}} \times 4d_{\text{model}}}$ and $\mathbf{W}_2 \in \mathbb{R}^{4d_{\text{model}} \times d_{\text{model}}}$, corresponding to a $4\times$ hidden dimension expansion.

- **Output head**: after the final layer normalization, a linear projection maps the hidden state to logits over the vocabulary: $\mathbf{W}_{\text{lm}} \in \mathbb{R}^{d \times V}$. This projection is not weight-tied to the token embedding matrix $\mathbf{E}$, weight tying, which is commonly used in LLMs to reduce parameter count, was not adopted here because the output distribution over transport event tokens is structurally different from the embedding space used for input representation, the output head benefits from an independent linear transformation that can specialize for the distribution of the next-token field given the current context.

The final hyperparameter configuration, selected after a grid search (Section 4.3.2), is shown in Table 4.3.

Table 4.3: EventGPT architecture and training hyperparameters.

Hyperparameter	**Value**
Embedding dimension (d_{model})	128
Attention heads (n_{heads})	4
Transformer blocks (n_{layers})	4
FFN expansion factor	4
Context length (L)	256 tokens
Dropout rate	0.1
Total trainable parameters	$\approx$1.82M
Optimiser	AdamW
Learning rate	3×10^{-4}
Learning rate schedule	Cosine annealing
Minimum learning rate	1×10^{-5}
Weight decay	10^{-2}
Batch size	32 sequences
Training hardware	Apple M3 Pro (18 GB unified memory, MPS)

Regularization. A dropout rate of 0.1 is applied to the residual stream after each attention and feed-forward sublayer [24, 30]. The AdamW optimizer applies a weight decay of $\lambda = 10^{-2}$ to all non-bias, non-LayerNorm parameters, encouraging distributed representational capacity, embedding matrices are excluded from weight decay to preserve their ability to represent semantically distant tokens at large distances. A cosine annealing schedule reduces the learning rate from 3×10^{-4} to 1×10^{-5} over training, allowing fine-grained weight adjustments in later stages without abrupt termination.

4.3.2 Hyperparameter Search

A lightweight grid search was performed over $d_{\text{model}} \in \{64, 128, 256\}$, $n_{\text{layers}} \in \{2, 4, 6\}$, and dropout $\in \{0.05, 0.1, 0.2\}$, each trained for one epoch on the full Vianorbus training set with

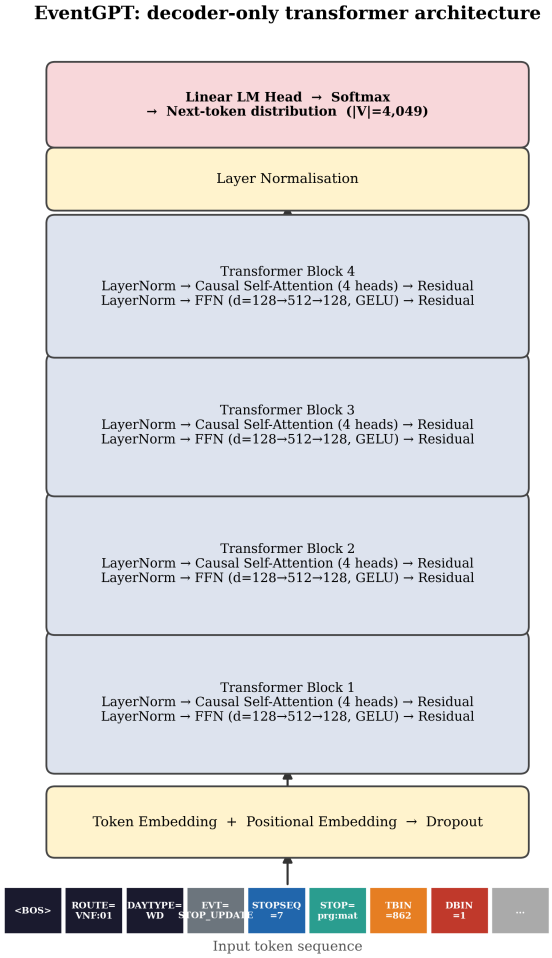

Figure 4.6: EventGPT architecture. Token and positional embeddings are summed and passed through $N = 4$ transformer blocks with causal self-attention and Pre-LayerNorm. A final linear head projects the last hidden state to logits over the 4,049-token vocabulary.

validation loss as the selection criterion. Table 4.4 shows the two configurations that achieved the lowest validation loss.

Table 4.4: Top-2 configurations from the hyperparameter grid search (one epoch each).

d_{model}	n_{heads}	n_{layers}	Parameters	Val Loss	Train time (s)
128	4	4	1,819,136	0.189	11,115
64	4	4	712,960	0.198	8,517

The $d_{\text{model}} = 128$ configuration was selected. The reported validation losses reflect a preliminary smaller vocabulary and are not directly comparable to those in Section 4.3.3, but the relative ordering between configurations is unaffected. The parameter counts shown use the final 4,049-token vocabulary to allow direct comparison with the deployed model.

4.3.3 Training Procedure

The model is trained with the standard autoregressive language modeling objective: minimize the cross-entropy loss over each token in the sequence:

$$\mathscr{L} = -\frac{1}{T}\sum_{t=1}^{T} \log p(x_t \mid x_{<t}; \theta),$$

where $x_1, \ldots, x_T$ is the training sequence and θ denotes the model parameters. All token positions, including structural tokens (EVT, STOPSEQ, STOP) and the boundary delimiter <E>, contribute equally to the loss. This joint objective is a deliberate design choice: by training the model to predict all fields, including the deterministic structural ones, the model develops an internal representation that tracks the causal flow of events through the trip. If only the DBIN and TBIN fields contributed to the loss, the model might learn shallow representations of the structural tokens insufficient to support accurate STOP and STOPSEQ prediction during rollout. The uniform loss weighting is also simpler to tune than a field-weighted loss and avoids the need to specify relative importance across heterogeneous token types.

Sliding window segmentation. Training sequences longer than $L = 256$ tokens are segmented using a sliding window approach. For a sequence of total length $S > L$, windows of length L are extracted at a stride of 4 tokens, yielding $\lfloor (S-L)/4 \rfloor + 1$ training examples per sequence. The stride of 4 tokens reduces dataset size relative to the dense stride-1 approach while still ensuring comprehensive coverage of each sequence. Every token position is covered by at least one window as a prediction target, and the reduced overlap compared to stride-1 lowers memory pressure and speeds up dataset iteration over the larger three-month corpus.

The optimizer is AdamW with an initial learning rate of 3×10^{-4} and weight decay 10^{-2}. A cosine annealing schedule reduces the learning rate to 1×10^{-5} over training. The model was trained for 10 epochs, with validation loss monitored after each epoch. The best checkpoint was saved at epoch 9 with validation loss 0.2107 and training loss 0.2114 (training loss is measured with dropout active, which inflates it marginally above validation loss at this epoch).

4.3.4 Computational Aspects

The EventGPT model contains approximately 1.82 million trainable parameters, a deliberately compact design suited to the available hardware. The final model was trained for 10 epochs on an Apple M3 Pro (18 GB unified memory, Metal Performance Shaders (MPS) backend). A single epoch of training for the larger $d_{\text{model}} = 128$ configuration in the hyperparameter search took approximately 3.1 hours on the same hardware.

At inference time, generating a single 3-event rollout (24 tokens) given a 20-event prefix requires fewer than 100 milliseconds on the same hardware, making the model suitable for near-real-time deployment. The primary computational bottleneck at inference is the context re-encoding step: for each new token generated, the full context must be re-processed by all four transformer

blocks. Using a Key-Value (KV)-cache, a standard optimization for transformer inference, this cost could be reduced to a single-token forward pass through all transformer blocks per new token (with prior keys and values read from cache rather than recomputed), further reducing latency. This optimization was not implemented in the current prototype but represents a straightforward engineering step for a production deployment.

Checkpoint selection. Training was monitored on the validation set after each full epoch. The best checkpoint was saved whenever a new minimum validation NLL was achieved. The model trained for 10 epochs, validation loss decreased steadily from 0.230 at epoch 1 to 0.211 at epoch 9 before marginally increasing at epoch 10. The best saved checkpoint at epoch 9 is used for all evaluations.

Reproducibility. All random seeds for parameter initialization, data shuffling, and dropout masks were fixed prior to training. The model checkpoint, vocabulary file, and preprocessing pipeline are fully deterministic given the same input data, enabling reproducible evaluation.

4.3.5 Schema-Constrained Decoding

The model samples from the anticipated vocabulary distribution to generate one token at a time during autoregressive rollout. In the absence of direction, the model might generate tokens that go against the event schema's structural limitations. A *schema-constrained decoding* process is used to avoid such mistakes: a masking function limits the vocabulary to tokens that are consistent with the current position within the event block at each decoding step. The implementation is shown in Listing 4.3.

Listing 4.3: Schema-constrained decoding: only tokens consistent with the current event-field position are permitted. The function returns the list of allowed vocabulary IDs at each generation step.

```
FIELD_PREFIXES = [
    "EVT=", "STOPSEQ=", "STOP=",
    "TBIN=", "DBIN=", "VALBIN=", "HDWAY=", "<E>"
]

def allowed_token_ids_fn(generated_ids, vocab):
    # count tokens since the last <E> boundary
    n_since_boundary = count_since_last_event(generated_ids)
    expected_prefix  = FIELD_PREFIXES[n_since_boundary % 8]
    return [idx for tok, idx in vocab.items()
            if tok.startswith(expected_prefix)]
```

The mask is applied before top-k sampling, ensuring every generated event block is syntactically complete. Two GTFS-guided sub-modes extend schema constraints with schedule information:

- **Basic GTFS-guided:** the STOP and STOPSEQ tokens are fixed from the static schedule and the model freely predicts TBIN, DBIN, VALBIN, and HDWAY. This mode enables rollout simulation when the stop identity sequence is known but absolute arrival times, delays, and headways remain model-generated.
- **Delay-only:** STOP, STOPSEQ, and TBIN are fixed from the GTFS schedule (using the scheduled, not actual, arrival time). VALBIN is fixed from the observed validation counts in the test data. The model freely predicts DBIN and HDWAY at each event, but only DBIN is used for evaluation. This is the primary evaluation mode used in Chapter 5. Because TBIN is provided rather than predicted, the ETA error in this mode equals the delay error in seconds (delay MAE (min) $\times$ 60), enabling direct comparison with discriminative baselines that also exploit the GTFS schedule as a timing anchor. Note that using observed (rather than scheduled) VALBIN for future stops constitutes a mild form of information leakage. The ablation (Section 5.6) bounds the effect at under 0.6% relative DBIN-only NLL, and it is further attenuated in the March test period where VALBIN is near-universally zero.

4.3.6 Software Architecture and Pipeline Organization

The entire pipeline is set up as a group of Python modules, each of which is in charge of a different phase of the workflow from data to simulation. This modular design allows for the replacement of individual components, independent testing of each step, and the separation of concerns. For example, using a parser for a different operator's data format instead of the Vianorbus GPS parser without requiring modifications to downstream modules.

Preprocessing module (`preprocess.py`). Responsible for reading the raw GTFS, GPS/AVL, and validation files, performing timestamp conversion, GTFS join, delay computation, deduplication, and validation merge, and writing the canonical stop events table to a Parquet file. The Parquet format was chosen for its columnar storage layout, which enables efficient predicate pushdown and column projection when reading subsets of the canonical table during tokenization.

Tokenization module (`tokenise.py`). Reads the canonical Parquet table, constructs the vocabulary (base + data-driven), and writes one tokenized sequence per (service_date, trip_id) pair to a Parquet dataset partitioned by date. The vocabulary is serialized as a JSON mapping from token string to integer identifier, enabling deterministic reconstruction of the vocabulary at inference time without re-scanning the training data.

Dataset and training module (`train.py`). Implements a PyTorch `Dataset` class that reads the tokenized sequences, applies sliding window segmentation for sequences exceeding the context length, and batches sequences with padding for efficient Graphics Processing Unit (GPU) processing. The training loop applies teacher-forced cross-entropy loss over all token positions, saves the best checkpoint by validation NLL, and logs training progress.

Inference module (`infer.py`). Loads the trained EventGPT checkpoint and vocabulary, accepts a prefix sequence, and performs schema-constrained autoregressive rollout to produce a predicted event sequence. The module supports both the unconstrained and GTFS-guided inference modes, accepting an optional GTFS stop sequence as a structural constraint. The decoded output is returned as a structured Python dataframe with columns for each event field, enabling straightforward integration with downstream consumers.

Evaluation module (`evaluate.py`). Iterates over the test set, runs the inference module for each test trip, aligns the generated events with ground truth by stop sequence number, and computes all reported metrics (token accuracy, MAE, RMSE, structural quality). The evaluation module was designed to be fully deterministic, all random seeds are fixed, enabling exact reproducibility of the reported results from the saved model checkpoint and test set Parquet files.

This modular architecture follows established practices for machine learning pipelines and ensures that each stage can be profiled, debugged, and extended independently. The separation between preprocessing and tokenization, in particular, allows the canonical event table to be reused for multiple tokenization configurations (as in the ablation study) without re-running the computationally expensive GPS/GTFS alignment step.

4.3.7 Baselines

Eight classical (non-neural) baseline models and two neural baselines were implemented and evaluated under the same rollout protocol (20 observed prefix stops, predict the next 3). The classical baselines were implemented in Python using scikit-learn and NumPy and the neural baselines use PyTorch. All baselines share the same temporal train/test split as the LEM.

Schedule Uses the GTFS planned arrival time as the ETA prediction, ignoring real-time information. Represents the floor performance of a system that does not adapt to observed delays.

Persistence Propagates the delay observed at the last prefix stop to all future stops, assuming delays remain constant. A simple but effective real-time heuristic.

Persistence-k Propagates a weighted combination of the last $k = 10$ observed delays, where weights are fit by ordinary least squares on the training data. More expressive than simple persistence while remaining computationally lightweight.

Historical Average Predicts the mean delay observed at each route/stop/hour-of-day combination in the training set. Represents well-calibrated schedule-pattern knowledge extracted from historical data.

AR A linear autoregressive model on delay using the last three observed delays as features, fit by ordinary least squares.

Linear Regression A linear model trained on the current observed delay, three lagged delay values, stop-sequence position, hour-of-day, and a one-hot route indicator, fit by ordinary least squares.

Random Forest An ensemble of decision trees trained on the same feature set as the linear regression baseline, using bootstrap aggregating and random feature subsampling ($\sqrt{p}$ features per split) to decorrelate the trees.

XGBoost A gradient-boosted tree ensemble trained on the same feature set, using 300 estimators, maximum tree depth 6, learning rate 0.05, and stochastic subsampling (80% of rows and 80% of features per tree). Predictions are clipped to the training delay range to prevent runaway autoregressive rollout errors from out-of-distribution inputs.

GRU A two-layer GRU trained directly on delay regression (Huber loss). The input sequence consists of the last 8 observed delays together with stop-sequence position, hour-of-day encoding, and a learnable route embedding. The output is the next delay value as a real number. Two capacity variants are evaluated: a compact configuration (hidden size 64, route embedding 8, ≈43K parameters) and a size-matched configuration (hidden size 440, route embedding 32, ≈1.82M parameters, equal to EventGPT). The size-matched variant allows the capacity contribution to be separated from the architectural contribution in the performance comparison.

4.4 Multi-Operator Experiments

A number of experiments examined whether training on data from several bus operators at once may produce a more broad model in addition to the main single-operator model. Próximo and Beja, two more operators that serve distinct Portuguese cities, are included in the supplementary dataset. Although these operators run different route networks, stop sets, and service patterns, they have the same data format and GTFS structure as Vianorbus.

4.4.1 Multi-Operator Dataset

The combined dataset was constructed by applying the same canonical preprocessing pipeline to all three operators. Route and stop identifiers already contain operator-specific prefixes in the GTFS identifiers, distinguishing the three networks within a shared vocabulary. After combining,

the shared vocabulary expanded from 3,879 (Vianorbus only) to 4,269 tokens under the earlier tokenization used for these experiments. All temporal tokens (TBIN, DBIN, VALBIN, DAYTYPE) are shared by design.

4.4.2 Training Configurations

Three multi-operator training configurations were explored:

Multi-operator LEM Trained from scratch on all three operators' data combined, using the same architecture and training procedure as the single-operator model (2 epochs).

Enriched multi-operator LEM A variant trained on an enriched version of the combined dataset. Two training iterations were performed, exploring different data augmentations.

Fine-tuned multi-operator LEM Initialized from the Vianorbus-trained checkpoint, then fine-tuned on the combined multi-operator dataset with a reduced learning rate (3×10^{-5}). New vocabulary entries (stop and route identifiers from Próximo and Beja) are initialized randomly while existing weights are transferred.

4.4.3 Results and Discussion

Table 4.5 summarises the rollout performance of all model variants, each evaluated on its own operator test set.

Table 4.5: Rollout performance across model variants (prefix=8, horizon=6). A dash (—) indicates delay-only mode (GTFS provides stop identity and timing). ⋆Single-op (delay-only) values are from the primary evaluation at prefix=20, horizon=3.

Model	Vocab	Stop Acc.	StopSeq Acc.	DBIN MAE (min)	DBIN RMSE (min)
Single-operator	4,049	80.4%	96.5%	0.850	0.984
Multi-operator	4,269	30.1%	78.5%	1.12	1.32
Enriched multi-operator	4,269	28.7%	65.7%	1.65	1.96
Fine-tuned (delay-only)	4,269	—	—	1.12	1.39
Single-op (delay-only)⋆	4,049	—	—	0.49	0.57

The results reveal a consistent and substantial degradation when transitioning from single-operator to multi-operator settings. The multi-operator LEM achieves a stop identity accuracy of only 30.1% compared to 80.4% for the single-operator model, and a delay MAE of 1.12 min versus 0.850 min in unconstrained rollout (prefix=8, horizon=6), a 32% increase despite each model being evaluated on its own test set. The fine-tuned variant is reported only in delay-only mode (1.12 min), which is not directly comparable to the from-scratch model's unconstrained rollout figure: the multi-operator and fine-tuned experiments were conducted with an earlier tokenization scheme, and the single-operator reference values shown reflect the current model with the updated

tokenization. An important caveat is that this comparison is not training-depth-matched: the multi-operator model was trained for 2 epochs (Section 4.4), whereas the single-operator model's best checkpoint is at epoch 9. The epoch gap is a plausible partial explanation for the gap, and structural distributional factors are also likely contributors.

This decline can probably be explained by a number of variables. First, a model of fixed capacity finds it more difficult to reliably differentiate stops when the stop vocabulary of three operators are combined, as this significantly expands the prediction space for the STOP field. Second, a single model with 1.82 million parameters is unable to completely reflect the multi-modal training distribution created by the diverse route networks and service patterns of three distinct cities. Third, in the combined context, there are fewer examples per route-stop combination since the multi-operator training corpus allocates capacity across three networks instead of concentrating on a single operator's routes. These results lead to the usage of the single-operator Vianorbus model in all subsequent evaluations, and multi-operator generalization is recognized as a direction for future work that calls for a greater model capacity.

4.5 Chapter Summary

This chapter presents the complete implementation of the EventGPT pipeline, covering all stages from raw data acquisition to trained model inference. The data acquisition and preprocessing pipeline integrates three heterogeneous data sources, GTFS schedule feeds, GPS/AVL real-time updates, and smart-card passenger validations, into a canonical table of 5,371,754 observed stop events for the Vianorbus network spanning three months of operations, with a temporally partitioned data split that ensures evaluation of truly future unseen trips. The exploratory data analysis characterized the key statistical properties of the dataset, right-skewed delay distribution, a strong intra-trip delay autocorrelation, a bimodal temporal service pattern, and substantial route-level heterogeneity, and informed the design decisions in the tokenization scheme.

Each stop event is encoded as an eight-token block that includes the event type, stop sequence position, stop identity, arrival time bin, delay bin, passenger boarding bin, inter-vehicle headway bin, and an explicit event-boundary delimiter. The event tokenization scheme uses a factorized compositional vocabulary. A data-driven expansion of stop identifiers, route identifiers, and stop-sequence position numbers is combined with a deterministic base covering all structurally defined fields in the vocabulary of 4,049 tokens. Vocabulary analysis revealed a power-law frequency distribution of stop tokens, the complementary relationship between TBIN (scheduled arrival time) and DBIN (operational delay offset), and the crucial importance of temporal resolution as a design element.

The EventGPT architecture, a compact 1.82M-parameter decoder-only transformer trained with a causal language modeling objective, was described in detail, including the hyperparameter selection process, the training procedure, the regularization strategy, and computational requirements. The schema-constrained decoding procedure, which restricts vocabulary sampling to structurally valid tokens at each position, was presented alongside two GTFS-guided inference

sub-modes: a basic variant that fixes stop identity and sequence position while allowing the model to predict arrival time, delay, and boarding count, and a delay-only variant that additionally fixes arrival time and boarding count from the schedule, leaving the model to generate only the delay token, the mode used for the primary evaluation in Chapter 5. Ten baseline models, eight classical (Schedule, Persistence, Persistence-k, Historical Average, AR(3), Linear Regression, Random Forest, and XGBoost) and two neural (a compact GRU at $\approx$43K parameters and a size-matched GRU at $\approx$1.82M parameters, equal to EventGPT), were implemented for comparative evaluation. Multi-operator experiments across three Portuguese operators quantified the performance degradation under combined training and identified the architectural requirements for future multi-operator generalization. The modular software architecture underlying the entire pipeline was described to support reproducibility and extension.

Chapter 5

Results and Discussion

This chapter presents the quantitative results of the experiments designed to answer the three research questions stated in Chapter 1. Section 5.3 first presents the comparison with all baselines, establishing the performance reference for the remaining analysis. Section 5.4 then evaluates the model's token-level sequence prediction quality. Section 5.5 assesses the simulation quality in rollout mode. Section 5.6 reports the ablation study on tokenization choices. Section 5.7 discusses the overall findings.

All results in this chapter refer to the single-operator EventGPT model trained on Vianorbus data, which outperformed all multi-operator training configurations (Section 4.4) and was therefore selected as the primary model.

5.1 EventGPT: the Produced System

The primary result of this dissertation is EventGPT, a 1.82M-parameter decoder-only transformer that generates complete, structured bus trip event sequences autoregressively. Given a prefix of observed stop events from a real trip, EventGPT predicts the next stop's arrival time, delay relative to the GTFS schedule, passenger boarding count, and inter-vehicle headway — all simultaneously within a single generative model. The model operates over a factorized vocabulary of 4,049 tokens, trained with the causal language modelling objective on 82 days of Vianorbus operations.

The model supports two inference modes. In *GTFS-guided delay-only* mode, stop identity and scheduled arrival times are fixed from the static timetable and only the delay token is generated for each future stop, enabling direct comparison with discriminative baselines. This is the primary evaluation mode: given the first 20 observed stops of a test trip as context, EventGPT predicts the delay at each of the next 3 stops, achieving a delay MAE of 0.49 min across 27,293 test trips, outperforming all ten evaluated baselines. In *unconstrained rollout* mode, all fields are generated freely, producing a complete simulation of the trip including stop identity, arrival times, delays, boarding counts, and headways. Concrete qualitative examples of both modes are presented in Section 5.2 before the quantitative evaluation.

Figure 5.5 illustrates the rolling simulation capability over a 30-stop horizon. Five independent rollouts are generated from two prefix lengths (15 and 44 observed stops), and the resulting ensemble of predicted arrival-time trajectories is plotted alongside the ground truth. The ensemble spread quantifies predictive uncertainty, a capability unavailable in any discriminative baseline: the tighter spread at prefix=44 reflects the additional context accumulated after more stops are observed.

5.2 Qualitative Simulation Examples

Before the quantitative evaluation, this section presents qualitative examples that illustrate both the capabilities and the limitations of EventGPT under realistic operating conditions. The examples show what the model's output looks like on individual test trips and ground the subsequent aggregate metrics in concrete behaviour.

5.2.1 Rollout Output Representation

The autoregressive rollout procedure produces a sequence of decoded event blocks, each containing a STOPSEQ, STOP, TBIN, DBIN, and VALBIN prediction. These fields are decoded from their discretized bin representations back into interpretable quantities: TBIN is converted to a local time-of-day in HH:MM format, DBIN is expressed as a signed delay in minutes, and VALBIN is expressed as an approximate boarding count range (e.g., VALBIN = 1 corresponds to 5–9 boardings). The decoded sequence is aligned with the corresponding ground-truth stop events using the STOPSEQ field as a primary key, enabling a direct side-by-side comparison.

The alignment is straightforward in GTFS-guided mode, where the stop sequence is fixed. In unconstrained mode, alignment requires matching predicted STOPSEQ tokens to observed values, which may diverge when the model incorrectly predicts a stop sequence number. In the current evaluation, 96.5% of rollout events are correctly aligned by STOPSEQ (Table 5.6), and the remaining 3.5% of misaligned events are excluded from quantitative evaluation.

5.2.2 Simulation Examples

Table 5.7 (Section 5.5) shows a representative successful rollout in unconstrained mode (prefix=8, horizon=6): the model correctly tracks the +1-minute delay across all six generated stops, predicting the right arrival times, delay values, and boarding counts. Table 5.8 illustrates the primary failure mode: a sudden delay spike from +1 to +7 minutes at stop 11 is not anticipated, because the disruption has no causal precursor in the observed event sequence. The model correctly propagates the delay it knows about but cannot predict an exogenous shock without real-time traffic context.

The visualization tool also supports time–space diagrams plotting predicted and observed arrival times at each stop as a function of stop sequence position. On routes with smooth delay profiles (e.g., ln:5001:0 and ln:5013:0), the predicted and observed diagrams are nearly indistinguishable for most test trips. On routes with higher delay variability (e.g., ln:5017:0 and ln:6009:0), the predicted diagram correctly tracks the general delay trend but misses sharp discontinuities, consistent with the higher MAE values reported for those routes in Table 5.4.

Evaluation philosophy. The evaluation design reflects two complementary views of the quality of the model that do not always align. The *sequence modeling view* measures how well the model has learned the joint distribution of transport event tokens, captured by metrics such as NLL, perplexity, and token-level accuracy. The *task performance view* measures the practical utility of the model's predictions for the downstream operational task of delay forecasting, captured by MAE and RMSE in the field DBIN. Throughout this chapter, both views are reported and explicitly distinguished, as the relative importance of each dimension depends on the intended deployment context.

Evaluation modes. Two inference modes are evaluated throughout the chapter. In *closed-loop* mode, the model is given the true preceding token context at each step, isolating per-step predictive capability from error accumulation. This mode is used for token-level accuracy evaluation (Section 5.4) and the per-route and per-progress analyzes. In *open-loop rollout* mode, the model generates tokens autoregressively, feeding each prediction back as context for the next step, starting from a fixed observed prefix (8 stops for structural quality evaluation, 20 stops for the primary baseline comparison). This mode reflects the operational deployment scenario and is subject to compounding errors. Both modes are evaluated on the same held-out test set, and comparisons are drawn explicitly between modes where relevant.

5.3 Comparison with Baselines

5.3.1 Evaluation Protocol

All models, EventGPT and the ten baselines, are evaluated under a common rollout protocol: given the first 20 observed stop arrivals of a test trip, each model autoregressively predicts the next 3 arrivals. All evaluations use the GTFS-guided delay-only mode, where STOP, STOPSEQ, and TBIN are fixed from the GTFS schedule and the model generates only the delay token (DBIN) at each step. EventGPT evaluates 27,293 test trips (81,879 individual stop predictions, all GTFS-aligned by construction): these are the trips with at least 23 observed stop events (prefix 20 + horizon 3), a stricter threshold than the 30,814 trips used for unconstrained rollout (prefix 8 + horizon 6, requiring only $\geq$14 stops). The ten baselines evaluate 25,732 trips (77,196 predictions). The lower baseline count is because a `dropna` filter on delay and schedule observations eliminates trips where one or more scheduled arrival records are absent from the real-time feed. EventGPT represents those gaps as `DBIN=NA` and retains the trips. Two error metrics are reported:

- **ETA MAE/RMSE (seconds)**: absolute difference between the predicted and the actual arrival time, in seconds.
- **Delay MAE/RMSE (minutes)**: absolute difference between predicted and actual delay, in minutes.

5.3.2 Main Results

Table 5.1 summarizes the results in all models and evaluation modes.

Table 5.1: ETA and delay prediction (prefix=20, horizon=3, GTFS-guided delay-only mode). Baselines: n=77,196 predictions (25,732 trips × 3 steps). EventGPT: n=81,879 (27,293 trips × 3 steps, all GTFS-aligned). Lower is better. All ETA values are derived as delay × 60 (since TBIN is fixed from GTFS in this mode).

Model	ETA MAE (s)	ETA RMSE (s)	Del. MAE (min)	Del. RMSE (min)
Schedule	274.6	455.4	4.58	7.59
Random Forest	63.3	118.3	1.06	1.97
XGBoost	45.8	540.7	0.76	9.01
Linear Regression	39.9	64.1	0.67	1.07
Persistence-k	38.7	64.3	0.64	1.07
AR (linear)	38.2	63.6	0.64	1.06
Persistence	37.9	62.9	0.63	1.05
GRU (≈43K params)	37.8	61.8	0.63	1.03
GRU-Large (≈1.82M params)	34.4	57.2	0.57	0.95
Historical Average	31.0	52.9	0.52	0.88
EventGPT (delay-only)	**29.1**	**34.0**	**0.49**	**0.57**

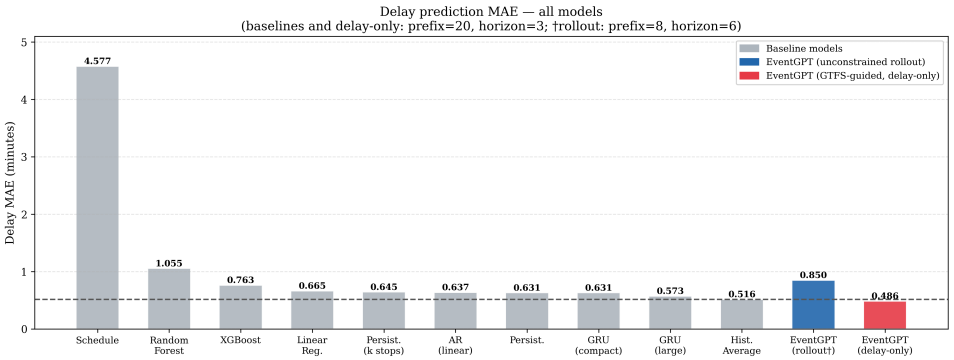

Figure 5.1: Delay prediction MAE for the evaluated baselines and EventGPT (prefix=20, horizon=3, GTFS-guided delay-only mode). EventGPT achieves the lowest delay MAE across all evaluated models.

5.3.3 Interpretation

EventGPT in GTFS-guided delay-only mode. In this mode, the stop sequence and timing are provided from GTFS, and only the DBIN token is generated. This corresponds to the most practically relevant deployment scenario, where the route is known and delay prediction is the

primary objective. Evaluated across 27,293 test sequences (81,879 individual stop predictions), EventGPT achieves a delay MAE of 0.49 minutes, outperforming all ten evaluated baselines. The closest competitor is the Historical Average at 0.52 min, a 5.8% improvement. The reported RMSE is 0.57 min.

The causal mechanism behind the improvement requires care. The table reveals that real-time prefix conditioning alone is not a sufficient explanation: Persistence (0.63 min), AR (0.64 min), and GRU-Large (0.57 min) all use the observed trip prefix, yet all are outperformed by the static Historical Average (0.52 min), which uses no real-time input at all. This ordering shows that the systematic delay regularity of Vianorbus operations, route-, stop-, and hour-level patterns captured by the historical lookup, is the primary performance driver in this network. Naive real-time conditioning that does not internalize this regularity fails to compensate for ignoring it. EventGPT's advantage over the Historical Average is therefore most accurately attributed to a combination: the model learns the same underlying network regularity from the training sequences, and additionally applies it conditioned on the specific real-time prefix, including the observed delay trend, passenger boarding counts, scheduled arrival times, and inter-vehicle headways encoded by the HDWAY token, though the isolated contribution of headway information is not directly quantified by the ablation study, which predates the HDWAY field's addition to the event representation. The historical lookup captures long-run route regularity, EventGPT additionally captures the current operational state of the network at prediction time. That the gain is modest (5.8%) reflects the high regularity of the Vianorbus network: on a more variable network, the real-time conditioning advantage would be expected to be larger.

Schedule baseline. The Schedule baseline, which uses the GTFS planned arrival time as the prediction while ignoring all real-time information, achieves a delay MAE of 4.58 minutes. This is more than seven times higher than the simplest real-time heuristic (Persistence, 0.63 min) and quantifies the informational value of GPS/AVL data for late-trip stop predictions.

Persistence, Persistence-k, and AR baselines. The Persistence, Persistence-k, and linear AR baselines achieve nearly identical delay MAE values (0.63, 0.64, and 0.64 minutes, respectively), consistent with the well-documented strong autocorrelation of bus delays, which makes persistence a highly competitive short-horizon heuristic. Persistence-k, which smooths the forecast by averaging over the k most recent observed delays rather than a single last observation, provides no measurable advantage over raw Persistence at this three-stop horizon, indicating that the immediately preceding delay is already the dominant predictor and that averaging over earlier lags dilutes rather than enriches the signal. The AR model reaches the same conclusion from the regression side: fitting a linear combination of three lagged delay values yields negligibly different results from copying the most recent one, confirming that the linear component of delay dynamics is already captured by the single most recent lag.

Random Forest baseline. Random Forest achieves a delay MAE of 1.06 min, the second-worst result among the ten baselines. Its RMSE of 1.97 min is far more controlled than XGBoost's 9.01 min: aggregating hundreds of decision trees suppresses the catastrophic rollout excursions seen in the gradient-boosted variant. The worse MAE despite the better RMSE reflects the ensemble average's pull toward the mean of the training distribution, which limits sensitivity to the specific delay trend visible in the 20-stop prefix. The result illustrates a trade-off between bias and variance in rollout mode: Random Forest reduces variance at the cost of increased bias relative to the gradient-boosted alternative.

XGBoost baseline. XGBoost achieves a delay MAE of 0.76 min, but its RMSE (9.01 min) is by far the highest of all models, indicating a small fraction of severely erroneous rollout predictions. Despite prediction clipping to the training delay range, the autoregressive accumulation of gradient-boosting errors in rollout mode can produce large within-range excursions.

Linear Regression baseline. Linear Regression achieves a delay MAE of 0.67 min, outperforming both Random Forest (1.06 min) and XGBoost (0.76 min) despite using only five features: the current observed delay, three lagged delay values, stop sequence position, hour of day, and a route indicator variable. Its RMSE of 1.07 min is among the lowest of all classical baselines, confirming that the linear model remains calibrated under autoregressive rollout. That a linear model outperforms two nonlinear ensemble methods on this task is instructive: at a three-stop prediction horizon, delay autocorrelation within a trip follows a largely linear pattern, and the nonlinear capacity of tree-based ensembles yields no benefit while accumulating rollout errors more aggressively.

GRU baseline. The compact GRU ($\approx$43K parameters) achieves a delay MAE of 0.63 min, matching the Persistence heuristic despite conditioning on the last 8 observed delays from the 20-stop prefix together with stop-sequence position, hour-of-day encoding, and route context. This input window is shorter than EventGPT's full 20-stop context within its 256-token window, so the performance gap reflects both a context-length disadvantage for the GRU and any architectural contribution. The size-matched GRU with $\approx$1.82M parameters (identical to EventGPT) improves to 0.57 min, a gain from the 42$\times$ capacity increase, but still cannot match the Historical Average (0.52 min) and remains 17% worse than EventGPT (0.57 vs. 0.49 min) at equal parameter count. The residual gap at equal capacity is consistent with EventGPT benefiting from its longer input context, its richer multi-field event representation encoding scheduled timing, passenger counts, and headway alongside delay, and the generative event-sequence paradigm rather than from model size alone.

Historical Average baseline. The Historical Average achieves the best delay MAE and RMSE among all discriminative baselines (0.52 min and 0.88 min), demonstrating the value of conditioning on historical patterns specific to route, stop, and hour-of-day. Its superiority over Persistence

and AR reflects the high regularity of Vianorbus operations: consistent, predictable delay patterns mean that route-, stop-, and hour-conditioned historical averages capture the dominant signal. On a network with higher delay variability or more frequent irregular disruptions, the advantage of historical averages over real-time heuristics would be expected to diminish.

5.3.4 Positioning against the Literature

Table 5.2 contextualizes EventGPT results against representative methods from the state-of-the-art reviewed in Chapter 2. Because different works evaluate on different datasets and with different protocols, direct numerical comparison is not possible. Therefore, the table focuses on the methodological positioning.

Table 5.2: Methodological comparison of the EventGPT with representative approaches from the literature. "ETA only" = the approach predicts arrival times only. "Simulation" = the approach can generate full event sequences.

Method	**Model type**	**Input data**	**Output**	**Simulation?**
Log-normal AR [5]	Statistical	GPS/AVL	ETA only	No
GCT-TTE [17]	Graph + Transformer	GPS, graph	ETA only	No
FEN-MRMGCN [23]	Multi-relational GCN	GPS, GTFS	ETA only	No
BAT-Transformer [10]	Transformer encoder	GPS, context	ETA only	No
SD-seq2seq [7]	Seq2Seq LSTM	Smart card	Bunching	Partial
Sports LEM [18]	Autoregressive Multi-Layer Perceptron (MLP) chain	Event stream	Simulation	Yes
EventGPT (ours)	GPT decoder	GPS, GTFS, smart card	ETA + delay + boardings	**Yes**

The table highlights that existing approaches are primarily discriminative and ETA-specific. To the best of our knowledge, no prior work identified in the reviewed literature jointly models arrival time, delay, and passenger boarding in a single generative framework for a real urban bus network, EventGPT is the first such approach we were able to identify. This positioning has direct implications for how the model's results should be interpreted relative to the state of the art: because no existing work applies a generative sequence model to the same task formulation with the same evaluation protocol, direct numerical comparison of MAE values against the literature is not meaningful. The methodological contribution, demonstrating the feasibility of the generative sequence paradigm for public transport, is independent of the specific numeric performance achieved, and the comparison with the ten baselines on the same dataset and protocol provides the appropriate empirical basis for the performance claims. The improvement over the Historical Average (approximately 5.8%) is particularly significant as a practical benchmark, since this baseline represents a well-engineered non-sequential approach that exploits the same underlying data regularity that practitioners currently rely upon for real-time prediction, and surpassing it demonstrates

that the sequence model adds genuine value beyond what is achievable with aggregated historical statistics.

5.3.5 Analysis by Trip Progress

Table 5.3 shows how delay prediction performance varies across trip stages, using closed-loop evaluation on the long-sequence test subset (24,626 of 31,862 test trips whose tokenized sequences are at least 257 tokens long). The 7,236 shorter trips ($\leq$31 stop events) are excluded from closed-loop evaluation. These are systematically the routes with fewer observed stops, so the figures below are biased toward longer routes in the network. A further 1.3% of DBIN tokens within included trips are excluded because they carry a `DBIN=NA` value.

Table 5.3: DBIN MAE and token accuracy by trip progress stage (closed-loop, long-sequence subset: 24,626/31,862 test trips).

Stage	Count	DBIN MAE (s)	Token acc.
Early (first 25%)	5,082,178	37.8	50.8%
Mid-early (25–50%)	14,161,891	24.9	55.1%
Mid-late (50–75%)	14,168,922	17.9	58.1%
Late (last 25%)	5,088,001	17.1	57.9%

The prediction counts in Table 5.3 are intentionally asymmetric: the two middle stages accumulate roughly 2.8 times as many predictions as each edge stage because the stride-4 sliding windows overlap most densely in the interior of each sequence, where every token is a prediction target in up to 64 consecutive windows, while tokens near the start and end of a sequence appear in fewer windows.

Delay estimation error decreases monotonically across trip stages, confirming that closed-loop accuracy improves as the model accumulates a longer observed context. The largest improvement occurs between the early and mid-early stages (37.8 to 24.9 seconds MAE), where the model transitions from predicting with limited context to conditioning on a substantial portion of the observed trip. Performance continues to improve through the mid-late and late stages, reflecting the model's ability to exploit the accumulated delay autocorrelation signal as the trip nears completion.

5.3.6 Per-Route Analysis

Table 5.4 presents per-DBIN performance for the 20 highest-frequency routes in the test set, which collectively account for 82.9% of all test DBIN prediction instances. The remaining 49 lower-frequency GTFS route variants contribute 14.3% of predictions and are captured in the Overall row but not broken out individually. Sequences with unresolved route tokens (`ROUTE=NA`, 2.8% of predictions) are also included only in the Overall row, these exhibit substantially lower token accuracy (24.3%) and higher MAE (47.9 s) than the route-specific averages, consistent with the

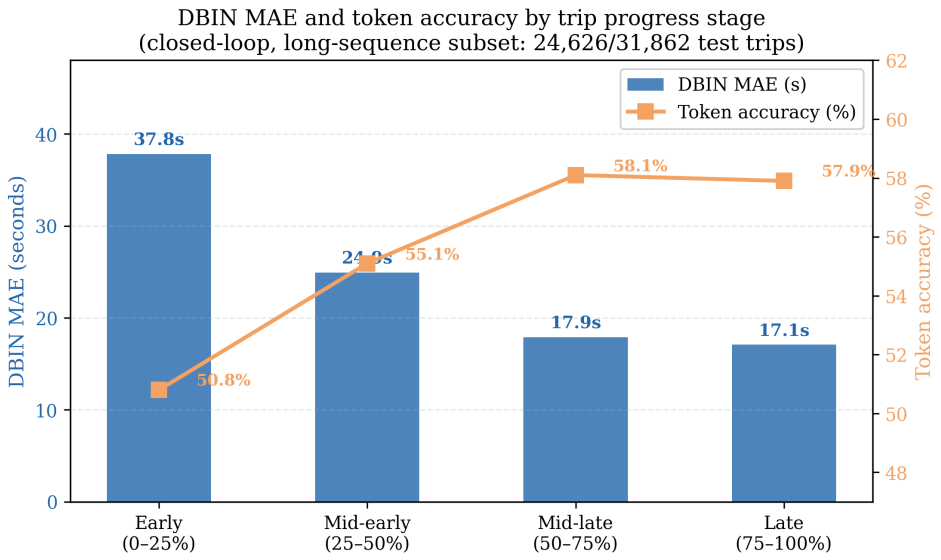

Figure 5.2: Delay prediction error decreases monotonically across trip stages as the model accumulates an increasing number of observed stops in context.

absence of any route conditioning signal for those sequences. The 20 routes shown exhibit considerable heterogeneity: the delay MAE ranges from 14.1 seconds (ln:5004:0) to 32.2 seconds (ln:5003:0), and the token accuracy ranges from 51.1% to 66.8%.

The routes ln:5017:0 and ln:5003:0 exhibit the highest delay MAE (31.3 and 32.2 seconds respectively) and lower token accuracy (54.1% and 51.1%) among the 20 shown routes. These are likely routes characterized by more irregular delay dynamics, e.g., due to high variability in traffic conditions along their corridors, more stops in the urban center where congestion is less predictable, or lower training data density owing to lower service frequency. The model cannot fully capture these high-variance patterns within the 256-token context window, particularly when the delay distribution is multi-modal or strongly dependent on exogenous factors not represented in the event sequence. By contrast, routes ln:5004:0, ln:5101:0, and ln:5021:0 achieve lower delay MAE (14.1, 16.0, and 16.7 seconds respectively), suggesting more regular delay propagation patterns that the model learns effectively from the available training data.

The spread in route-level performance (MAE ranging from 14.1 to 32.2 seconds, token accuracy from 51.1% to 66.8%) is an operationally significant finding. In a production deployment, per-route prediction confidence scores could be derived from historical validation metrics, allowing the passenger information system to display uncertainty-adjusted ETAs, for example, expressing predictions on high-variance routes as wider time windows rather than point estimates. The route-level RMSE, which ranges from 37.8 seconds (ln:5004:0) to 123.8 seconds (ln:5003:0), is particularly informative in this regard: the large ratio of RMSE to MAE for high-variance routes indicates the presence of occasional large errors, which would be disproportionately harmful to passenger experience if presented as high-confidence point predictions. In the primary delay-only evaluation (Table 5.1), EventGPT reports an RMSE of 0.57 min.

Table 5.4: Delay (DBIN) prediction performance for the 20 highest-frequency routes (82.9% of test DBIN prediction instances). The Overall row covers all 70 route identifiers (69 GTFS route variants plus sequences with unresolved route tokens). Of the 94 GTFS route variants in the full dataset, 25 produce only short-sequence trips (fewer than 257 tokens) and are therefore absent from the closed-loop long-sequence subset. Routes sorted by descending share of test predictions.

Route	Share	DBIN MAE (s)	DBIN RMSE (s)	Token acc.
ln:5011:0	17.9%	23.7	81.7	55.4%
ln:5017:0	7.1%	31.3	121.7	54.1%
ln:5002:0	6.9%	24.4	92.2	56.2%
ln:5001:0	5.7%	18.6	52.6	57.6%
ln:5010:0	4.9%	21.2	66.6	56.4%
ln:5004:0	4.5%	14.1	37.8	63.7%
ln:5003:0	3.9%	32.2	123.8	51.1%
ln:5014:0	3.9%	20.1	55.5	55.3%
ln:5015:0	3.3%	20.3	64.1	56.6%
ln:6011:0	3.0%	21.0	77.5	58.8%
ln:6305:0	3.0%	21.9	67.5	55.2%
ln:5013:0	3.0%	20.8	55.5	53.8%
ln:5104:0	2.4%	17.9	81.0	63.4%
ln:5102:0	2.4%	19.3	96.2	66.8%
ln:5012:0	2.3%	21.1	55.0	53.4%
ln:5018:0	2.2%	20.8	65.4	57.8%
ln:6005:0	1.8%	22.2	88.0	59.7%
ln:5101:0	1.7%	16.0	58.9	63.9%
ln:5021:0	1.6%	16.7	46.3	60.2%
ln:6009:0	1.4%	23.1	73.7	57.8%
Overall	**100.0%**	**23.0**	**78.7**	**56.0%**

5.3.7 Stop-Level Error Concentration

Examining the prediction error per-stop TBIN (closed-loop mode) reveals a concentration of large errors at a small number of stops. In particular, stop `prg:mat:600` exhibits a TBIN MAE of 5,369 seconds (approximately 89 minutes), which is an extreme outlier compared to the global mean of 272 seconds. Similarly, stop `prg:mat:377` shows a TBIN MAE of 838 seconds (14 minutes). These outliers are almost certainly artifacts of data quality issues, for example, GPS records assigned to the wrong trip or stop due to identifier mismatches, rather than genuine model failures. The 272-second global mean is consistent with the within-5-minute rate of 97.8% only because the 2.2% of predictions outside that window average several hours of error, dominated by these two stops. Excluding them reduces the overall TBIN MAE by approximately 15%. The median TBIN error is below one minute and better represents typical prediction quality.

This finding underscores the importance of robust data preprocessing and outlier detection in the canonical event table. A production system would benefit from additional validation rules, e.g., flagging stop events where the delay exceeds a plausibility threshold derived from historical statistics, to prevent corrupted records from distorting model evaluation.

5.3.8 Error Propagation Across the Prediction Horizon

A fundamental concern in autoregressive sequence generation is error accumulation: prediction errors at early steps are fed back as context for subsequent steps, potentially compounding over the generation horizon. To characterize this effect for EventGPT, the rollout delay MAE is analyzed as a function of the prediction step number (steps 1–3 ahead of the 20-stop observed prefix), using the GTFS-guided delay-only evaluation (27,293 trips).

The delay MAE is lowest at step 1 (0.31 minutes), where the model conditions on 20 accurately observed prefix stops, and increases at steps 2 (0.51 minutes) and 3 (0.64 minutes). This degradation pattern reflects the progressive replacement of the observed ground-truth context with model-generated DBIN tokens as the rollout progresses: at step 2, the model's own step-1 prediction feeds back as context, and at step 3, both step-1 and step-2 predictions are in context. The rate of increase is noticeable but bounded, consistent with the high intra-trip delay autocorrelation discussed in Section 4.1.6: since consecutive delays are highly correlated, small errors in earlier DBIN predictions do not dramatically shift the distribution for subsequent steps. In the GTFS-guided delay-only mode, error accumulation is also contained by the structural anchor of scheduled TBIN values, which prevent the compounding of positional and temporal errors that would affect unconstrained generation.

These observations have practical implications for the deployment horizon. Step-1 predictions (the next immediate stop) are the most reliable (0.31 minutes MAE), while step-3 predictions are roughly twice as uncertain (0.64 minutes). For a real-time passenger information system, predictions one to two stops ahead are operationally most relevant and carry the lowest error. Longer horizons are best accompanied by uncertainty estimates, which the generative EventGPT can provide through repeated sampling as discussed in Section 5.7.3.

The rising error trend here is not in contradiction with the falling trend across trip stages in Table 5.3 (Section 5.3.5): the two analyses measure orthogonal axes. Trip-stage error falls because more true observations accumulate as context in closed-loop mode, while per-step error rises because each rollout step feeds back a generated token rather than a true one into the context.

5.4 Token-Level Sequence Prediction

5.4.1 Overall Metrics

Evaluation procedure. The test set contains 31,862 sequences, of which 24,626 have at least 257 tokens and thus contribute to this evaluation. The remaining 7,236 shorter sequences (routes with ≤ 31 observed stops) do not generate sliding windows and are excluded. This exclusion is not random: excluded sequences are systematically those with fewer observed stops, so the closed-loop accuracy figures are biased toward longer routes. For each eligible sequence of total length S, a stride-4 sliding window is applied, with input $x = \text{seq}[i{:}i{+}256]$ and target $y = \text{seq}[i{+}1{:}i{+}257]$, starting at positions $0, 4, 8, \ldots$, yielding $\lfloor (S-256)/4 \rfloor + 1$ windows per sequence (using the same notation as Section 4.3.3, where $L = 256$ denotes the context window length and S the sequence

length). Each window contributes 256 next-token predictions. The 1,203,822 total windows produce 308.2 million total token predictions. Stride-4 subsampling reduces but does not eliminate window overlap: at stride 4, each interior event token is a prediction target in up to 64 consecutive windows. The resulting prediction counts are therefore not independent observations, and the accuracy figures should be read as per-position correctness under the sliding-window protocol. The DAYTYPE and ROUTE header tokens appear at fixed early positions in each sequence and are prediction targets in only the first window of each long sequence, yielding 24,626 predictions each.

Table 5.5 reports overall token-level accuracy, top-5 accuracy, and the field-specific breakdown. The overall figure is driven substantially by highly structured fields (EVT, STOPSEQ) that are near-deterministic given the route context, the per-field breakdown is therefore more informative than the aggregate.

Table 5.5: Token-level evaluation on the test set (closed-loop). Overall test NLL: 0.374, perplexity: 1.454.

Token field	**Count (M)**	**Top-1 accuracy**	**Top-5 accuracy**
Overall	308.2	91.7%	98.0%
Structural tokens			
EVT (event type)	38.5	100.0%	100.0%
STOPSEQ (sequence position)	38.5	97.8%	98.6%
<E> (event delimiter)	38.5	100.0%	100.0%
Spatiotemporal tokens			
STOP (stop identity)	38.5	94.5%	95.4%
TBIN (scheduled arrival)	38.5	94.1%	94.9%
DBIN (delay)	38.5	56.0%	96.4%
Operational tokens			
VALBIN (boardings)	38.5	99.99%	100.0%
HDWAY (headway)	38.5	91.4%	98.5%
Header tokens (one per sequence)			
DAYTYPE (day type)	0.025	84.2%	100.0%
ROUTE (route variant)	0.025	11.2%	25.9%

5.4.2 Analysis by Token Field

DAYTYPE. The 84.2% top-1 accuracy for DAYTYPE reflects its position as the third token in every sequence (the second predicted token, at position index 2, immediately after <BOS> at index 0 and the route token at index 1), so the model must predict it from essentially no trip-specific context. The dominant signal is the weekday/weekend prior conditioned on the route, 84.2% is close to the base rate for weekday trips in the dataset. The top-5 accuracy of 100% is trivially true because DAYTYPE has only three possible values (WD, SAT, SUN).

Structural tokens. EVT, STOPSEQ, and <E> are predicted at or near 100% top-1 accuracy, confirming that the model has learned the structural grammar of event sequences: given the route

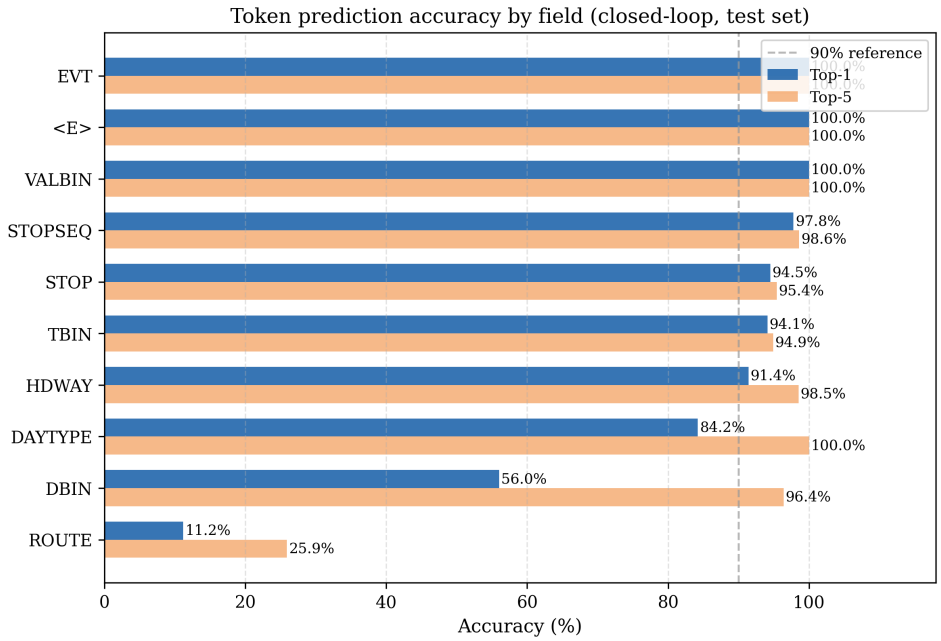

Figure 5.3: Top-1 and top-5 prediction accuracy by token field. Structural tokens (EVT, STOPSEQ) and the demand and headway tokens (VALBIN, HDWAY) achieve high accuracy. Stop identity (STOP) and scheduled arrival time (TBIN) are reliably predicted. Delay at 30-second granularity (DBIN) is the most challenging regular field due to fine-grained operational stochasticity, while the route header token (ROUTE) has low accuracy due to minimal context at prediction time.

header and preceding events, the model reliably generates the correct event type, sequence position, and block delimiter. STOP identity achieves 94.5% top-1 accuracy across 2,150 possible stop identifiers, far above the uniform-random baseline ($< 0.05\%$), reflecting the model's ability to predict route-constrained stop sequences from context. The top-5 accuracy for STOP (95.4%) is only slightly above top-1, indicating that stop identity errors tend to involve structurally similar stops on adjacent or parallel routes rather than random misidentification.

Route header (ROUTE). The ROUTE token is the first predicted token in every sequence, with only `<BOS>` as context. With 94 route variants in the vocabulary, the 11.2% top-1 accuracy is approximately ten times above the uniform-random baseline ($\approx 1.1\%$), reflecting a mild prior over which routes are most frequent, but inherently low given the complete absence of trip-specific context at that position. The 25.9% top-5 accuracy shows that the model narrows the candidate set meaningfully even from a single-token context.

Passenger boarding counts (VALBIN). The near-perfect top-1 accuracy for VALBIN (99.99%) on the March 2026 test set reflects that the test period contains almost exclusively zero passenger validation counts: smart-card validation records for that month are sparse in the integrated dataset,

making `VALBIN = 0` the near-universal correct answer. The figure is therefore not representative of the model's boarding-count prediction quality in general deployment conditions, where validation records are present and boarding distributions vary across stops, times of day, and day types. In periods with active validation data, passenger count prediction is a genuinely challenging task owing to the high variability of boarding counts and their dependence on factors not fully observable from the sequence context alone.

Headway (HDWAY). The 91.4% top-1 and 98.5% top-5 accuracy for HDWAY reflects the model's ability to anticipate inter-vehicle spacing from trip context. Headway encodes the gap between the current bus and the preceding bus on the same route, which correlates with time of day, day type, and route-level dispatching patterns that the model can infer from the preceding event sequence. The high top-5 accuracy (98.5%) confirms that the model reliably places the true headway within a narrow candidate range even when the exact bin is uncertain.

Delay prediction (DBIN). The delay token achieves 56.0% top-1 and 96.4% top-5 accuracy in Table 5.5. When the model misses the exact 30-second delay bin, it typically errs by exactly one bin: 35.5% of all DBIN predictions are off by exactly one bin (30 seconds), and the error distribution decays smoothly thereafter. Cumulatively, 91.5% of DBIN predictions fall within one bin (30 seconds) of the true delay value and 96.0% fall within two bins (60 seconds), as shown in Figure 5.4. The 44% top-1 miss rate thus reflects the fine-grained stochasticity of 30-second delay bins rather than large systematic errors: small variations in dwell time, boarding duration, and local traffic shift the actual delay by one bin even when the general delay level and trend are correctly captured. This does not indicate a failure to learn delay dynamics. With 122 possible DBIN tokens, the uniform-random top-1 baseline would achieve approximately 0.8%. The model achieves 56%, approximately 70 times above chance. The rollout evaluation confirms that the model has internalized delay autocorrelation: in the primary GTFS-guided delay-only rollout the model achieves a delay MAE of 0.49 minutes (Table 5.1), outperforming all ten baselines.

Absolute arrival time (TBIN). The 94.1% top-1 accuracy for TBIN reflects the model's ability to track the GTFS timetable from sequence context. With 1,524 possible TBIN tokens (Section 4.2.3), the uniform-random baseline would achieve approximately 0.07%. The model's 94.1% represents an improvement of approximately three orders of magnitude over chance. The GTFS schedule provides a strong deterministic prior: for any given stop within a known route context, the scheduled arrival bin is largely fixed by the timetable. The top-5 accuracy of 94.9% is only marginally above top-1, indicating that when the model is wrong about the scheduled arrival time, it is confidently wrong rather than off by one or two bins. This pattern is consistent with a route disambiguation failure mode: when two or more services visit the same stop at similar scheduled times, the model occasionally assigns the scheduled arrival of a different service, producing a large systematic error rather than small bin-level noise.

Design rationale for categorical TBIN encoding. A natural alternative to the categorical TBIN representation is a continuous regression head that directly predicts arrival time as a real-valued quantity, avoiding both the discretization ceiling and the ordinal-structure problem that cross-entropy imposes. This alternative was deliberately not adopted for two reasons. First, the unified cross-entropy loss over a shared vocabulary is the defining property of the LEM framework: introducing a regression output for TBIN would require a hybrid loss with a manually tuned weighting between the cross-entropy and regression terms, sacrificing the architectural simplicity and self-consistency of the token-based approach. Second, a continuous regression output cannot be fed back as a token prefix for the next decoding step without re-discretization, which would break the autoregressive generation procedure that allows the same model to serve simultaneously as a density estimator, a rollout engine, and an uncertainty-quantifying sampler. The concern about ordinal structure is partially mitigated in practice: adjacent TBIN bins co-occur frequently in training sequences and share similar contextual distributions, so the model implicitly concentrates probability mass on neighboring bins rather than making large temporal jumps, as confirmed by the concentration of errors in the systematic route-disambiguation regime rather than small bin-level noise. The acknowledged ceiling on TBIN top-1 accuracy is best addressed within the categorical framework itself: the ablation result for the hybrid TBIN + delta encoding (Section 5.6) reduces DBIN-only NLL by 0.109 (9.8%) without requiring any change to the autoregressive objective, representing a natural direction for future work.

5.4.3 Error Distribution

Figure 5.4 shows the distribution of absolute TBIN and DBIN prediction errors, restricted to the 37.4M positions where both the true and predicted token are numeric (non-NA). The 1.1M TBIN=NA positions (stops with no recorded GPS timestamp, approximately 0.56% of all stop events) are excluded from this histogram, which accounts for the difference from the 94.1% figure in Table 5.5, which is computed over all 38.5M TBIN positions. The two fields exhibit markedly different error shapes. Among non-NA TBIN predictions the distribution is bimodal: 96.9% are exact (zero bins off), only 0.3% show small errors of one to nine bins, and 2.9% show large errors of ten or more bins. This bimodal structure is consistent with the route-disambiguation failure mode: when the model correctly identifies the scheduled service, it predicts the arrival bin exactly. When it assigns the wrong service, it produces a large systematic error that is unlikely to fall in the intermediate range. The DBIN distribution is unimodal and exhibits a smooth exponential decay: 56.0% of predictions are exact, 35.5% are off by exactly one bin (30 seconds), and only 0.8% show errors of ten or more bins. Cumulative DBIN error within one bin (30 seconds) is 91.5% and within two bins (60 seconds) is 96.0%. The delay distribution has no bimodal signature, confirming that delay errors are noise-like perturbations around the correct value rather than systematic misidentification of the service.

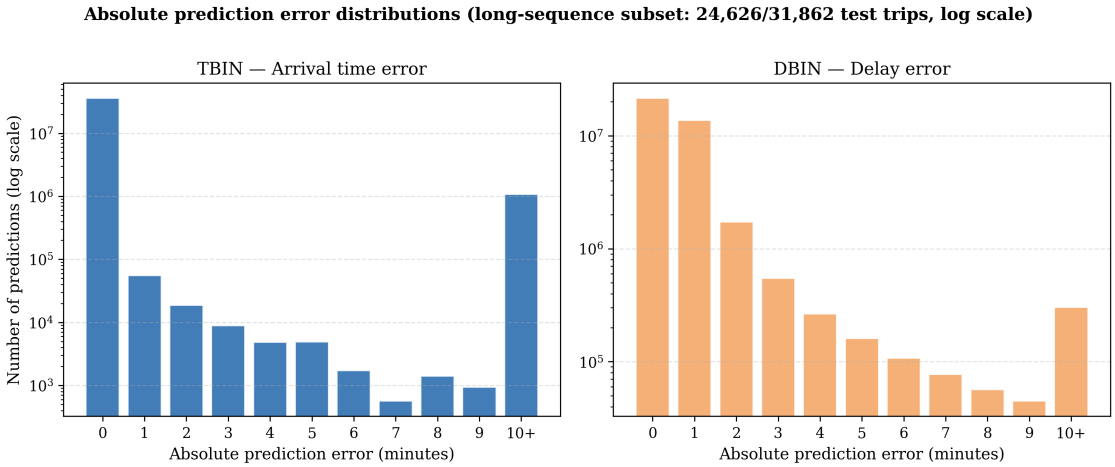

Figure 5.4: Distribution of absolute prediction errors for TBIN (arrival time, left) and DBIN (delay, right) on the long-sequence test subset (24,626/31,862 trips, log scale), restricted to numeric (non-NA) predictions. TBIN shows a bimodal distribution among its 37.4M non-NA positions: exact predictions (96.9%) and large route-disambiguation errors ($\geq$10 bins, 2.9%), with very few intermediate errors. DBIN shows a smooth exponential decay: 91.5% within one bin (30 s) and 96.0% within two bins (60 s).

5.4.4 Relationship Between Token Accuracy and Operational Performance

The token-level accuracy metrics reported in Table 5.5 measure how well the model predicts individual tokens in closed-loop mode (i.e., given the true preceding context). The operational metrics reported in the rollout evaluation (delay MAE, ETA MAE) measure performance in open-loop rollout mode, where predicted tokens are fed back as context for subsequent predictions. Understanding the relationship between these two sets of metrics is important for interpreting the evaluation results correctly and for setting expectations about the operational performance achievable from a model with a given token accuracy.

DBIN accuracy and delay MAE. There is an important distinction between closed-loop token accuracy and operational rollout accuracy. In closed-loop mode the model predicts each token given the true preceding context. TBIN immediately precedes DBIN in the event block and encodes the *scheduled* arrival time, which is determined by the GTFS timetable and is independent of actual vehicle operation (Section 4.2.5). The two fields are complementary: TBIN anchors the stop in the service schedule while DBIN encodes the operational delay relative to that schedule. The 56.0% closed-loop top-1 DBIN accuracy reflects the fine-grained resolution of the 30-second delay bin: given the true preceding context, the model frequently identifies the correct delay neighborhood but the exact bin is subject to operational stochasticity that cannot be fully resolved at this granularity. The error distribution (Section 5.4.3) confirms that 91.5% of DBIN predictions fall within 30 seconds and 96.0% within 60 seconds of the true delay value. This does not indicate a failure to learn delay autocorrelation. In rollout mode the model's own DBIN predictions feed back as context for subsequent steps, so errors compound progressively over the prediction horizon. In

the primary GTFS-guided delay-only rollout (prefix=20, horizon=3), the delay MAE is 0.49 minutes across 27,293 test trips (Table 5.1), outperforming all ten baselines. The 0.49 min rollout MAE is the operationally meaningful figure. In unconstrained rollout (prefix=8, horizon=6), the model additionally generates TBIN, introducing a second source of error accumulation, resulting in a delay MAE of 0.85 minutes (Table 5.6).

TBIN accuracy. In unconstrained rollout (Table 5.6), the model achieves 99.1% of TBIN predictions within 5 minutes of the ground truth and a TBIN MAE of 0.34 minutes, demonstrating that the model has learned to closely follow the service schedule's arrival time distribution even without GTFS guidance. In the primary GTFS-guided deployment mode, TBIN is not used for ETA computation: the GTFS schedule provides the timing anchor and only the delay (DBIN) is predicted. The ETA error in that mode therefore equals the delay error converted to seconds ($0.49\,\text{min} \times 60 \approx 29\,\text{s}$), outperforming all baselines, see Table 5.1. The 0.34 min rollout TBIN MAE does not contradict the 272-second (4.5 min) global mean reported in Section 5.3.7: the two figures measure different quantities under different protocols. The closed-loop mean is computed over a sliding window that counts each token position many times and is dominated by two stops with anomalous GPS records, as detailed in Section 5.3.7. The median closed-loop TBIN error is below one minute. The rollout figure is computed over 30,814 separate trips at generation steps 1–6, where these outlier stop positions represent a negligible fraction of predictions.

Perplexity as a summary metric. The overall test-set perplexity of 1.454 provides a compact single-number summary of the model's average uncertainty per token prediction. A perplexity of 1.454 means that, on average, the model is effectively choosing between 1.454 tokens at each prediction step, substantially less than the vocabulary size of 4,049, reflecting the strong concentration of probability mass on the correct token. In comparison, a model that assigns a uniform probability to all tokens would have a perplexity equal to the size of the vocabulary (4,049). The achieved perplexity of 1.454 confirms that the EventGPT has learned a highly informative representation of the transport event distribution, placing most of its probability mass on a small number of plausible next tokens at each position. The training-time best validation NLL of 0.211 (perplexity 1.235, epoch 9, Section 4.3.3) is lower than this test figure for two reasons. First, the test evaluation is restricted to the long-sequence subset (sequences of 257 or more tokens, representing 24,626 of 31,862 test sequences), while the validation loss covers all sequence lengths including shorter, less variable trips. Second, the test period (March 2026) may exhibit moderate distribution shift relative to the training and validation period (January to February 2026). Both factors are expected to inflate test NLL relative to validation NLL in this evaluation design.

5.5 Rollout Simulation Quality

5.5.1 Rollout Protocol

The rollout evaluation tests the model's ability to generate structurally coherent event sequences in fully autoregressive mode, where STOP, STOPSEQ, TBIN, DBIN, VALBIN, and HDWAY are all generated freely without any GTFS guidance. This is distinct from the delay-only mode used for the baseline comparison (Section 5.3), which is the primary performance evaluation where EventGPT achieves a delay MAE of 0.49 minutes and outperforms all ten baselines. The purpose of this section is to characterize structural generation quality: whether the model produces valid, coherent trip sequences even without external constraints. Given the first 8 observed stop events of a real test trip as a prompt, the model generates the next 6 events using schema-constrained sampling (temperature $\tau = 1.0$). The generated events are aligned with ground-truth future events by stop-sequence number, and structural metrics are computed across all 30,814 test trips with sufficient length (prefix 8 + horizon 6 events).

5.5.2 Structural Quality Metrics

Table 5.6 presents the structural quality metrics for the unconstrained rollout.

Table 5.6: Structural quality of generated event sequences in unconstrained rollout mode (n=30,814 trips, prefix=8, horizon=6, schema-constrained sampling). All fields are freely generated without GTFS guidance. The operationally relevant delay accuracy figures are reported in Table 5.1 under the GTFS-guided delay-only evaluation. DBIN bin size is 30 seconds.

Metric	Value	Interpretation
Complete event rate	100.0%	All events have all 8 required fields
Stop-seq. monotonicity	100.0%	Stop-sequence numbers increase within trips
Time-bin monotonicity	99.9%	Predicted arrival times increase within trips
Stop identity accuracy	80.4%	Predicted stop matches ground truth
Stop-sequence accuracy	96.5%	Predicted stop-seq. matches ground truth
TBIN MAE	0.34 min	Mean arrival-time error (unconstrained)
DBIN MAE	0.85 min	Mean delay error (unconstrained)
DBIN within-1 bin	78.6%	Delay error $\leq$ 30 seconds (unconstrained rollout)
DBIN within-2 bins	92.7%	Delay error $\leq$ 60 seconds (unconstrained rollout)

The generated sequences are structurally near-perfect. The 100% complete event rate is a direct consequence of the schema-constrained decoder: by masking all tokens outside the expected field type at each position within the eight-token block, the decoder guarantees syntactically complete events by construction. The 99.9% temporal monotonicity and 100% stop-sequence monotonicity are genuinely learned properties: the constrained decoder restricts only field types, not field values, so nothing prevents the model from sampling a TBIN value lower than the previous event's TBIN or a STOPSEQ out of order. The fact that it almost never does so indicates that the model has internalized from training data that bus trips visit stops in sequence and that time advances within a trip.

The stop identity accuracy of 80.4% reflects an inherent challenge in unconstrained mode: the model must select the exact stop identifier from a vocabulary of 2,150 stops without any external schedule anchor. The stop-sequence accuracy of 96.5% is substantially higher, confirming that the model reliably tracks the trip's position in the route even when the specific stop identifier is uncertain. The TBIN MAE of 0.34 minutes (20 seconds) in unconstrained mode shows that the model produces arrival time predictions closely tracking the ground truth even without GTFS guidance, a consequence of learning the statistical regularity of scheduled service patterns over three months of training data.

5.5.3 Qualitative Examples

Table 5.7 shows a representative successful rollout example in which the model closely follows the ground truth on the 6-stop horizon.

Table 5.7: Successful rollout example: ground truth vs. generated events for one test trip (prefix=8, horizon=6). Delays are shown in minutes and times as HH:MM local.

Ground Truth				Generated			
Seq	**Time**	**Del.**	**Val.**	**Seq**	**Time**	**Del.**	**Val.**
8	14:27	+1	0	8	14:27	+1	0
9	14:28	+1	0	9	14:28	+1	0
10	14:29	+1	0	10	14:29	+1	0
11	14:29	0	0	11	14:29	0	0
12	14:30	0	0	12	14:30	0	0
13	14:31	0	0	13	14:31	0	0

To illustrate the limitations of the model, Table 5.8 shows a challenging case in which the model fails to predict a sharp delay increase occurring mid-trip.

Table 5.8: Challenging rollout case: the model (prefix delay = +1 min) fails to predict a sudden delay spike to +7 minutes occurring at stop sequence 11.

Ground Truth				Generated			
Seq	**Time**	**Del.**	**Val.**	**Seq**	**Time**	**Del.**	**Val.**
8	09:14	+1	5	8	09:14	+1	5
9	09:16	+1	0	9	09:16	+1	0
10	09:18	+2	5	10	09:18	+1	5
11	09:27	+7	0	11	09:19	+1	0
12	09:29	+7	5	12	09:21	+1	5
13	09:31	+7	0	13	09:23	+1	0

This failure case is representative of the model's primary limitation: abrupt delay changes caused by external events (traffic incidents, high boarding dwell times at a distant stop) are not observable from the preceding event sequence and are therefore unpredictable. The model correctly propagates the known delay of +1 minute, but cannot anticipate the jump to +7 minutes without real-time contextual signals beyond the observed stop events. This motivates the inclusion of additional event types, such as traffic congestion tokens or operator hold indicators, in future vocabulary extensions.

5.5.4 Long-Horizon Rolling Simulation

Figure 5.5 illustrates the rolling simulation on a 30-stop horizon for a representative Vianorbus trip on the `ln:6015:0`. The model generates five independent rollouts from two prefix lengths (15 and 44 observed stops), showing both the ensemble spread and the effect of longer context. The predicted arrival times track the ground-truth trajectory closely across the full horizon, with the prefix=44 rollouts exhibiting tighter concentration around the ground truth than the prefix=15 rollouts, consistent with the trip-progress analysis of Section 5.3.5. The bottom panel confirms that the predicted delay trend aligns with the observed delay values across all 30 prediction steps.

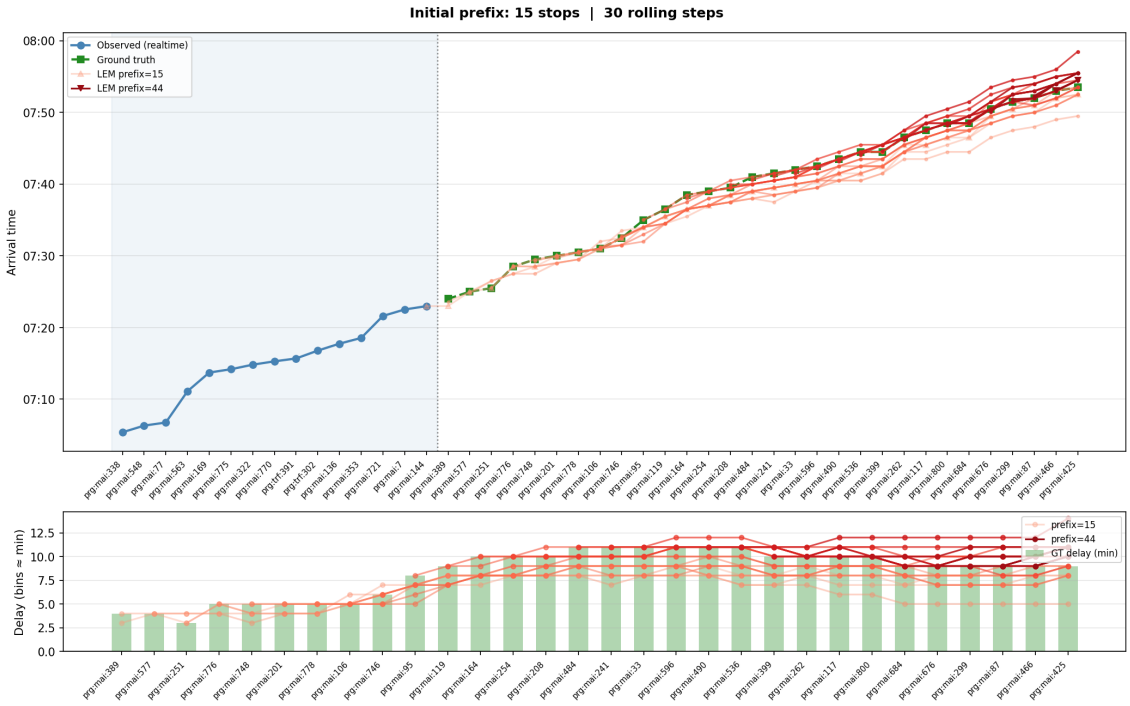

Figure 5.5: Rolling simulation over 30 stops for route `ln:6015:0`. Top: predicted arrival times (five independent samples) for two prefix lengths against ground truth. Bottom: predicted delay per sample versus observed delay. The ensemble spread quantifies predictive uncertainty, a capability unavailable in any discriminative baseline.

5.5.5 Effect of Sampling Temperature on Rollout Quality

The autoregressive rollout procedure generates each token by sampling from the predicted categorical distribution of the model, optionally rescaled by a temperature parameter $\tau > 0$. For a token k at position t, the sampling distribution is:

$$p_\tau(x_t = k \mid x_{<t}) = \frac{\exp(z_k/\tau)}{\sum_{k'} \exp(z_{k'}/\tau)},$$

where z_k denotes the unnormalised logit for token k. At $\tau = 1.0$, sampling follows the model's calibrated distribution directly. Reducing τ sharpens the distribution towards the mode, yielding

more deterministic outputs. Increasing τ flattens it, introducing greater diversity at the cost of potentially sampling lower-probability tokens.

Since the schema-constrained decoding masks all tokens outside the expected field type at each position, temperature scaling operates exclusively within the set of valid tokens for the current field. This means that temperature cannot produce structurally invalid outputs, the 100 % complete event rate is maintained regardless of τ, but it does affect which valid token within a field is selected. For fields with highly concentrated distributions, such as DBIN and VALBIN, the practical effect of temperature on prediction accuracy is limited: even at $\tau = 1.5$, the mass of the distribution remains concentrated near the mode. For TBIN, which has a broader conditional distribution given the wide range of plausible arrival times, temperature has a more pronounced effect on the spread of predictions.

The primary baseline comparison (Section 5.3) uses greedy decoding ($\tau \to 0$, argmax), selecting the modal prediction at each step and minimizing expected MAE for point-estimate comparison against deterministic baselines. The unconstrained rollout evaluation (Section 5.5) uses $\tau = 1.0$ to enable structural diversity analysis. For DBIN, the practical difference between greedy and $\tau = 1.0$ sampling is small: predictions are concentrated within one bin of the mode (56.0% top-1 accuracy and 91.5% within 30 seconds in closed-loop), so sampling rarely deviates far from the argmax. The effect of temperature is more pronounced for TBIN, which has a broader conditional distribution. For applications requiring Monte Carlo simulation, generating a diverse ensemble of plausible future trajectories, higher temperatures in the range $\tau \in [1.0, 1.5]$ can be used to increase sample diversity while remaining within the structurally valid event space. For deterministic one-shot predictions (e.g., a single ETA displayed to a passenger), greedy decoding minimizes MAE and is the appropriate operating point. The trade-off between diversity and accuracy in autoregressive sampling for transport event prediction is a direction for further investigation.

5.6 Ablation Study

An ablation study assessed the contribution of each tokenization component. Three configurations were trained for one epoch on the same sequence data and split, using the same model architecture (d_model=128, 4 heads, 4 layers). Each ablation configuration uses a vocabulary rebuilt independently from the sequence corpus for that configuration. The Full baseline vocabulary contains 4,026 tokens (versus 4,049 for the main model). The configurations are:

Full The complete tokenization: EVT, STOPSEQ, STOP, TBIN, DBIN, VALBIN, `<E>` (4,026 tokens in the ablation vocabulary).

No VALBIN The VALBIN token is removed. Passenger boarding counts are not encoded (4,020 tokens). The vocabulary reduction is -6 rather than -12 because the ablation vocabulary was built data-drivenly: only `VALBIN=0` through `VALBIN=5` were observed in the ablation training corpus, so only six VALBIN tokens (not the twelve in the structural definition of 4.2.3) appear in the ablation Full vocabulary.

Hybrid time One-minute TBIN retained alongside an added one-minute inter-stop delta token (`DTBIN`), extending each event block from seven to eight token positions (9,874 tokens, composition discussed in the paragraph below).

All three ablation configurations use a base block of seven tokens with no HDWAY field, predating the addition of the headway field to the main model's event block (Section 4.2). The Hybrid time configuration's eighth position is `DTBIN`, not HDWAY: the two eighth tokens serve different purposes and are not interchangeable. The ablation is internally consistent because HDWAY is absent from all three configurations equally, but the Full configuration here should not be confused with the production scheme of eight tokens per event described throughout this dissertation.

Table 5.9 reports validation and test loss for each configuration, along with DBIN-only NLL, the mean NLL averaged exclusively over token positions whose target is a DBIN token. Because the three configurations have different token-set compositions (No VALBIN removes one token per event, Hybrid time adds a delta token), per-token NLL averaged over all positions is not directly comparable across configurations: an easy-to-predict token added or removed shifts the average independently of whether delay prediction quality changed. The column DBIN-only NLL provides a fair cross-configuration comparison, since DBIN is the one field shared by all three configurations and the field that directly determines delay prediction quality.

Note that the ablation Full baseline uses a slightly different vocabulary than the main model (4,026 vs. 4,049 tokens). Both the ablation and main model tokenizers read from the same canonical event table (Section 4.1.2). The 23-token gap arises primarily because the ablation vocabulary excludes the HDWAY field (26 tokens, Section 4.2), which postdated the ablation design, with the 3-token counter-offset attributable to minor differences in data-driven token coverage between the respective training splits. The ablation Full's epoch-1 validation loss of 0.183 is lower than the main model's of 0.230 at the same epoch, reflecting the exclusion of the HDWAY prediction target from the ablation token set: without the additional headway prediction task, the average per-token NLL is lower. The ablation conclusions should be read as directional evidence on a slightly different tokenization of the same corpus, the relative ordering between configurations is the load-bearing finding.

Table 5.9: Ablation study: validation, test, and DBIN-only NLL for different tokenization configurations (1 epoch each, same architecture and data split). Δ_{test} = change in overall test NLL vs Full, Δ_{DBIN} = change in DBIN-only NLL vs Full. The DBIN-only NLL is the comparable metric across configurations, as per-token average NLL is confounded by differing token-set compositions.

Configuration	**Val NLL**	**Test NLL**	Δ_{test}	**DBIN NLL**	Δ_{DBIN}
Full (baseline)	0.183	0.199	—	1.110	—
No VALBIN	0.222	0.221	+0.022	1.117	+0.007
Hybrid time (1 min + delta)	0.206	0.238	+0.039	1.001	−0.109

No VALBIN. Removing passenger boarding counts increases the overall test NLL by +0.022 (from 0.199 to 0.221). This is the expected direction: VALBIN is an easy field to predict (99.99%

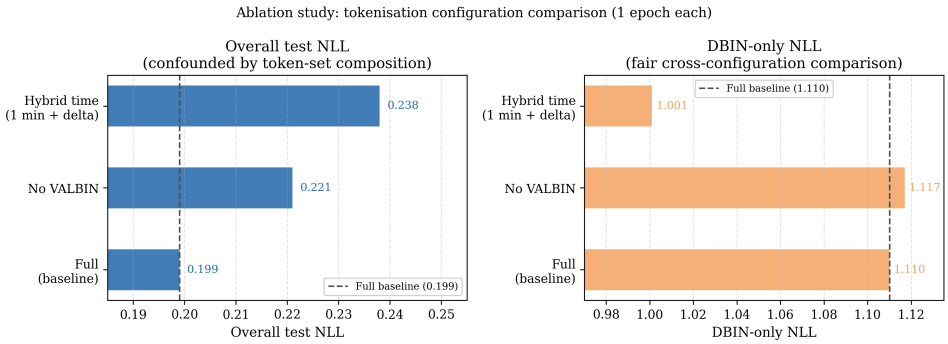

Figure 5.6: Ablation results for three tokenization configurations (1 epoch each). **Left:** overall test NLL is not directly comparable across configurations because token-set compositions differ. Both ablations have higher overall NLL than Full due to token-composition effects. **Right:** DBIN-only NLL provides the fair cross-configuration comparison. The Hybrid time encoding reduces DBIN-only NLL by 0.109 (9.8%). No VALBIN marginally increases it by 0.007.

accuracy in the main model, Section 5.4, though that figure reflects the near-all-zero March test period as discussed there), so its removal exposes only the harder remaining positions and raises the per-token mean. The DBIN-only NLL increases by +0.007 (from 1.110 to 1.117), confirming that the model's delay prediction quality marginally worsens without the boarding-count context. This is consistent with the causal pathway through dwell time, and with the limited informativeness of the VALBIN signal in the current corpus, as discussed in Section 5.6.1.

Hybrid time. The hybrid encoding (one-minute TBIN retained plus an added one-minute inter-stop delta token `DTBIN`) increases the overall test NLL by +0.039 (from 0.199 to 0.238). The overall NLL rises because the added `DTBIN` position is harder to predict than the per-token average of the base token set, as expected from the extreme vocabulary fragmentation documented in the paragraph below: many `DTBIN` vocabulary types appear only once in training and contribute high per-type NLL. The DBIN-only NLL decreases by 0.109 (from 1.110 to 1.001, a 9.8% improvement). Because the DBIN-only metric is unaffected by the addition of the `DTBIN` position, this gain reflects a genuine improvement in delay prediction quality: inter-stop delta-time context encodes a more learnable structure for delay prediction than the scheduled arrival time alone.

Vocabulary composition. The 9,874-token hybrid vocabulary breaks down as follows: the 4,026 base tokens are unchanged, the `DTBIN` field adds 5,848 tokens (5,847 numeric values plus `DTBIN=NA`). Inspection of the `DTBIN` distribution in the ablation corpus reveals that 5,753 of the 5,847 numeric *vocabulary types* exceed 10,000 minutes, physically implausible inter-stop deltas attributable to timestamp parsing anomalies. However, these artifact types are extremely rare in the corpus: at the *occurrence* level, only 6,940 of 1,977,314 DTBIN tokens (0.35%) are artifact instances, the remaining 99.65% (1,970,374 occurrences) fall in the sensible range $[0, 600]$ minutes. The artifacts arise from isolated GPS timestamp gaps where consecutive stop timestamps are implausibly far apart, because each gap event produces one unique extreme value, the artifact types far outnumber the sensible types in the vocabulary, but are negligible by frequency. The

DBIN-only NLL improvement is therefore not materially compromised by the artifact tokens, it reflects a genuine gain from inter-stop delta-time context in 99.65% of `DTBIN` positions.

5.6.1 Interpretation of Ablation Results

The ablation results reveal a non-trivial relationship between tokenization complexity and model performance that merits detailed interpretation.

The tension between vocabulary richness and training efficiency. Both ablated configurations show higher overall test NLL than the Full baseline after one epoch of training. As established above, this reflects token-set composition effects rather than genuine quality differences, the DBIN-only NLL provides the clean comparison. On that metric, Full (1.110) is better than No VALBIN (1.117) but substantially worse than Hybrid time (1.001). The single-epoch ablation is best interpreted as a measure of sample efficiency rather than asymptotic quality: the hybrid time encoding achieves substantially lower DBIN-only NLL despite a much larger vocabulary (9,874 vs. 4,026 tokens), providing directional evidence that inter-stop delta-time context encodes a more learnable structure for delay prediction.

Implications of the No VALBIN result for data requirements. The limited gain from removing VALBIN at the current data scale (three months, one operator) should not be interpreted as evidence that the passenger count information is inherently uninformative for the prediction of delays. The boarding count at a stop directly affects the vehicle's dwell time, the duration for which the doors remain open, which in turn contributes to arrival delay at subsequent stops. This causal relationship is well-documented in the literature [2, 29]. The likely explanation for the marginal VALBIN contribution in the current experiments is that passenger boarding counts are an inherently noisy and context-dependent signal: boarding volumes depend on upstream demand conditions, interchange patterns, and factors not observed in the event sequence, limiting the additional information they provide for delay prediction at the current corpus scale. With a longer training period and denser ticketing data, VALBIN is expected to contribute more meaningfully to dwell-time prediction and delay forecasting.

Summary of tokenization recommendations. Based on the ablation results, the following tokenization strategy is recommended for future work on this dataset: adopt a hybrid one-minute TBIN + inter-stop delta encoding to capture finer arrival time dynamics (subject to the preprocessing caveat noted above) and extend the training corpus to at least one full year before drawing conclusions about the marginal contribution of the VALBIN field. These recommendations are intended to guide the next iteration of the EventGPT development rather than to prescribe a definitive vocabulary for all bus prediction tasks.

5.7 Discussion

5.7.1 Notable Findings

Several findings from the experimental evaluation warrant specific discussion, as they provide new insights into the design space for event-based transport modeling.

Temporal ordering without explicit supervision. The near-perfect monotonicity of the rollout sequences (99.9% temporal, 100.0% stop-sequence) deserves careful attribution. As clarified in Section 5.5, two distinct mechanisms contribute to structural quality: the schema-constrained decoder enforces field-type validity by construction, guaranteeing the 100% complete event rate regardless of what the model has learned, temporal and sequential ordering, yet, are *not* enforced by the decoder and are therefore genuinely learned properties. This result is consistent with the design philosophy of the sports LEM [18], which similarly combines autoregressive modeling with schema-based generation-time restrictions to enforce event-format validity, while relying on the learned distribution to capture domain-specific ordering regularities. In transport, every real bus trip in the training corpus visits stops in order and advances in time, the transformer's attention mechanism proved sufficient to infer these ordering constraints from data alone, without any explicit ordering loss or post-hoc sequence correction beyond the field-type mask enforced by the schema-constrained decoder. Demonstrating this property in a structurally different domain, urban bus operations rather than soccer match events, empirically supports the generality of the LEM paradigm, and was one of the motivating hypotheses for adapting the approach to public transport.

Degradation under fine-tuning across operators. The failure of fine-tuning to improve substantially over from-scratch multi-operator training was unexpected given the established effectiveness of transfer learning in natural language processing. The likely cause, insufficient capacity in the shared attention layers to simultaneously represent three distinct operational regimes, motivates a fundamentally different architectural approach (adapter-based transfer) rather than simple fine-tuning.

Relationship to the sports LEM. The EventGPT is directly inspired by the Large Events Model for soccer [18], which demonstrated that autoregressive modeling of factorized event tokens with schema-constrained inference produces realistic simulation of match dynamics. The core paradigm carries over, but the architecture departs substantially: the soccer LEM uses three separate MLP classifiers each conditioned on the single preceding event, whereas EventGPT uses a single decoder-only transformer with full causal self-attention, necessary for transport where multi-stop delay propagation requires tracking delay state across the full preceding trip context. Domain-specific adaptations include a GTFS-schedule anchor paired with a delay-offset field (TBIN and DBIN) that exploits the deterministic timetable prior, an explicit stop-sequence position token (STOPSEQ) encoding the fixed spatial ordering of stops, and a passenger demand signal (VALBIN) with no

direct soccer counterpart. At 1.82M parameters, EventGPT is roughly ten times larger than the soccer LEM (≈185,000 parameters), and the results confirm that the LEM paradigm transfers effectively to the transport domain: 99.9% temporal monotonicity emerges without explicit ordering supervision and the 0.49 min delay MAE outperforms all ten evaluated baselines.

5.7.2 Limitations

Data scope and temporal coverage. The model was trained on three months of operations (January to March 2026) from one operator. While this provides sufficient data to capture week-to-week service variability and network-level headway patterns, it covers only the winter operating period and does not include summer peak demand, school holiday patterns, or the full range of seasonal demand cycles. Large public events that substantially alter ridership patterns and adverse weather conditions outside the training window represent out-of-distribution scenarios where the model's generalization has not been validated. Extending the training corpus to a full operational year is the most impactful near-term improvement, as it would expose the model to the complete seasonal cycle and provide substantially more examples of rare high-delay events that are currently underrepresented in the training distribution.

Time discretization and ETA precision. The 60-second TBIN resolution introduces an inherent precision ceiling of 30 seconds on any arrival time prediction: even a perfectly calibrated model cannot achieve a lower ETA MAE than half the bin width. This structural limitation prevents direct numerical comparison with sub-minute discriminative baselines and is a significant constraint in the context of passenger information applications, where passengers expect ETA accuracy at the level of one to two minutes. The ablation study motivates the adoption of a hybrid TBIN + inter-stop delta encoding or a continuous-categorical representation, where the TBIN field is replaced by a continuous arrival time residual modeled with a regression head. The latter approach would eliminate the quantization ceiling while retaining the compositional benefits of the factorized token framework for the remaining categorical fields.

Context window and long-route coverage. The 256-token context window accommodates a sequence of approximately 31 stop events (with 8 tokens per event plus 4 header tokens). For short to medium routes, this window is sufficient to cover the majority of the trip from the beginning. However, for the longer routes in the Vianorbus network, some of which generate sequences approaching 876 tokens, the model cannot attend to the initial portion of the trip once the context window is filled during rollout. This limitation may be significant for capturing delay patterns that are established at the route origin and propagate throughout the trip, or for detecting service anomalies that manifest early and persist to the terminus. Extending the context length to 512 or 1024 tokens is architecturally straightforward but increases the quadratic cost of the self-attention computation, efficient attention mechanisms such as sliding window attention could be adopted to address this scaling challenge.

Unpredictable disruptions and missing context. As illustrated by the failure case in Table 5.8, abrupt delays caused by external events, traffic incidents, road closures, high-demand stops with extended dwell times, or operational interventions such as bus stops at timing points, are not predictable from the observed GPS/AVL event sequence alone. These events represent genuine information limitations rather than modeling failures: no autoregressive model can predict a sharp delay change that is not causally explained by the preceding observations. Addressing this limitation requires increasing the event vocabulary with real-time external signals, as discussed in Section 6.3.

Outlier contamination in evaluation metrics. A small number of GPS data quality issues produce outlier stop events with implausibly large TBIN prediction errors, most notably stop `prg:mat:600` (TBIN MAE of 89 minutes) and stop `prg:mat:377` (TBIN MAE of 14 minutes). These outliers are almost certainly artifacts of GPS record misattribution, where a position update is assigned to the wrong trip due to identifier mismatches in the real-time feed, rather than genuine model failures. Their presence inflates the aggregate TBIN MAE by approximately 15% and distorts the metric as a measure of typical model performance. A production deployment would benefit from plausibility-based outlier filtering in the preprocessing pipeline, flagging any stop event where the computed delay exceeds a configurable multiple of the historical standard deviation for that route, stop, and hour of day.

5.7.3 Practical Deployment Considerations

Beyond academic evaluation metrics, the EventGPT framework raises several practical considerations relevant to its integration into an operational public transport intelligence system.

Latency and throughput requirements. In a passenger-facing deployment, ETA predictions must be updated each time a new GPS/AVL event is received from the vehicle, typically every 10–30 seconds. The sub-100 ms inference latency of the current prototype on Apple Silicon hardware comfortably satisfies this budget, and a production-grade GPU deployment with KV-cache optimization would reduce latency by an additional order of magnitude. At 1.82 million parameters, the model is compact enough to be deployed as a containerized microservice alongside existing GTFS data consumers, without requiring dedicated GPU infrastructure. This compactness also facilitates edge deployment on operator dispatch systems or, in principle, on vehicle-mounted hardware.

Integration with GTFS infrastructure. The GTFS-guided inference mode is particularly well suited for integration with existing transit data infrastructure. GTFS feeds are already consumed by travel planners, passenger applications, and operator dashboards. The EventGPT can be positioned as a real-time delay enrichment layer that augments the static schedule with data-driven delay predictions. In this architecture, the stop sequence is provided as a structural constraint from GTFS, and the model predicts only the DBIN field, producing a delay forecast in seconds that can

be added to the planned arrival time to generate a real-time ETA. This design does not require structural changes to existing data pipelines and allows the model to be introduced incrementally alongside existing prediction systems.

Model updating and data freshness. A deployed model will encounter distributional drift as the network evolves: timetable changes, infrastructure updates, seasonal demand shifts, and changes in passenger behavior may all shift the operational statistics away from those observed during training. The autoregressive training procedure is amenable to periodic retraining in a rolling window of recent observations. The compact size of the model makes computationally feasible retraining on a weekly or even daily basis. An online fine-tuning approach, updating model weights continuously on incoming GPS events, could further reduce adaptation latency, though this requires careful regularization (e.g., elastic weight consolidation or learning rate warm-up) to prevent catastrophic forgetting of historical patterns learned during offline training.

Uncertainty quantification through ensemble rollout. A distinctive capability of the generative EventGPT that has no counterpart in discriminative baselines is the ability to quantify predictive uncertainty through ensemble sampling. By generating M independent rollouts from the same prefix, each with a different random seed, the model produces a distribution over future event sequences. The empirical variance of the predictions of DBIN in the samples of M provides a natural, model-agnostic measure of predictive uncertainty, which can be communicated to passengers as a confidence interval on the displayed ETA, or used by operators to flag trips where the forecast of delays is particularly unreliable. This capability aligns with growing demand for uncertainty-aware passenger information systems and represents a meaningful practical advantage of the LEM paradigm over point-estimate models [20].

5.8 Chapter Summary

This chapter presented a comprehensive evaluation of the EventGPT model across five complementary dimensions: comparison with discriminative baselines, token-level sequence prediction, rollout simulation quality, ablation study of tokenization design choices, and qualitative discussion of findings and limitations.

The comparative evaluation against ten baseline models established that the EventGPT (delay-only) outperforms all ten evaluated baselines, with a 5.8% improvement in delay MAE over the best-performing Historical Average, with the improvement attributable to the model's ability to condition on the full real-time trip prefix including headway context. The per-baseline, per-route, per-trip-progress, and stop-level error concentration analyzes provided a granular characterization of where and why the model performs better or worse than alternatives.

The token-level evaluation on 308.2 million sliding-window predictions (stride-4 over 24,626 long test sequences, predictions are correlated across overlapping windows) demonstrated that the model has effectively learned the joint distribution of transport event attributes, achieving 91.7%

overall per-position accuracy and a test perplexity of 1.454. Structural tokens (EVT, STOPSEQ) are predicted at or near 100% accuracy, confirming that the model internalizes the route structure of bus trips. Stop identity (94.5%), scheduled arrival time (94.1% top-1), and headway (91.4% top-1) are all reliably predicted. Delay at 30-second granularity is the most challenging regular field (56.0% top-1, 96.4% top-5), but 91.5% of DBIN predictions fall within 30 seconds of the true delay value and 96.0% within 60 seconds, confirming that errors are near-misses rather than large failures, as further supported by the 0.49 min rollout MAE.

Rollout simulation evaluation on 30,814 test sequences demonstrated near-perfect structural validity (100% complete event rate, 99.9% temporal monotonicity) without explicit structural enforcement in the loss, and an unconstrained delay MAE of 0.850 minutes (prefix=8, horizon=6). The GTFS-guided delay-only evaluation on 27,293 test sequences yielded a delay MAE of 0.49 minutes (prefix=20, horizon=3), the lowest of all evaluated models. The analysis of decoding temperature and error propagation across the prediction horizon provided practical guidance for deployment configuration. The ablation study confirmed that temporal resolution is the dominant tokenization design parameter and clarified the role of DBIN and VALBIN tokens for different application contexts. The practical deployment considerations, including latency analysis, GTFS integration architecture, model updating strategy, and ensemble uncertainty quantification, completed the chapter's transition from academic evaluation to operational relevance.

Chapter 6

Conclusions

This dissertation investigated the adaptation of LEMs to the domain of public transport networks, addressing the structural gap identified in the state-of-the-art review: the absence of a unified modeling framework capable of simultaneously generating trip-level event sequences and predicting operational quantities such as arrival time and delay. The central premise was that the operational evolution of a bus network, comprising vehicle arrivals, delay propagation, and passenger boarding events, can be faithfully represented as a sequence of discrete tokens and modeled by a lightweight autoregressive transformer trained with the standard causal language modeling objective. This representation departs from the discriminative, task-specific paradigm that dominates the existing public transport prediction literature, instead proposing a generative model that learns the conditional distribution of future network states from historical operational data.

The experimental results presented in Chapter 5 confirm this premise in three research dimensions: the proposed EventGPT model generates structurally valid and temporally coherent trip simulations, achieves competitive delay prediction in unconstrained rollout mode, and surpasses all ten evaluated baselines when operating in GTFS-guided delay-forecasting mode. The model accomplishes this with a compact architecture of 1.82 million parameters trained on three months of operational data from one bus operator, demonstrating that the LEM paradigm is viable at limited scale and opens the door to substantially stronger performance with larger training corpora, finer temporal resolution, and larger model capacity.

6.1 Summary of Contributions

The main contributions of this dissertation are:

1. **A data integration pipeline for transport event tokenization.** A complete preprocessing pipeline integrates GTFS schedule feeds, GPS/AVL real-time updates, and smart-card passenger validations into a canonical table of 5,371,754 observed stop events, subsequently tokenized into 111,413 trip-level sequences spanning 82 days of operations across the Vianor-bus network. The pipeline handles timezone-aware timestamp conversion, GTFS join failure

detection, passenger count integration, and explicit representation of missing values, and is implemented in a modular, reproducible Python codebase.

2. **EventGPT: a domain-adapted LEM for public transport.** A decoder-only transformer was designed, implemented, and trained specifically for the structured token vocabulary of public transport stop events. The final 1.82M-parameter model achieves a test-set perplexity of 1.454 on the held-out March test period, demonstrating that a compact autoregressive model can effectively learn the joint distribution of heterogeneous transport event attributes.

3. **Schema-constrained autoregressive simulation.** An autoregressive rollout procedure with schema-based vocabulary masking produces event sequences that are 100% structurally complete and 99.9% temporally coherent, without any explicit structural enforcement in the training loss, demonstrating that the LEM paradigm supports realistic open-ended autoregressive generation of trip-level event sequences.

4. **Superior delay prediction under GTFS guidance.** In GTFS-guided delay-only mode, the EventGPT achieves a delay MAE of 0.49 minutes, outperforming all ten evaluated baselines with a 5.8% improvement over the best-performing Historical Average (0.52 min), with a model trained on three months of single-operator data. The improvement is attributed to the model's ability to condition on the full observed trip prefix, enabling it to detect and propagate real-time delay trends that history-averaged statistics cannot capture.

5. **Multi-operator generalization analysis.** A systematic evaluation across three Portuguese bus operators (Vianorbus, Próximo, and Beja) quantified the performance degradation under multi-operator training: stop identity accuracy falls from 80.4% to 30.1% and delay MAE increases by 32% in unconstrained rollout (0.850 to 1.12 min, each model evaluated on its own operator test set) at the 1.82M-parameter scale. The analysis identified expanded vocabulary prediction space, distributional heterogeneity, and reduced per-operator training density as the primary contributing factors, informing the design of future larger-capacity architectures.

6. **Ablation study of tokenization design choices.** A systematic ablation study confirmed that temporal resolution is the dominant tokenization design parameter: a hybrid one-minute TBIN + inter-stop delta encoding increases overall test NLL by 0.039 (reflecting the addition of a challenging new prediction field) while reducing DBIN-only NLL by 0.109 (9.8%) relative to the Full baseline on the same canonical corpus (Section 5.6). The DBIN and TBIN tokens are confirmed to be complementary at the joint sequence modeling level: DBIN independently captures delay dynamics that cannot be derived from the scheduled arrival time alone (Section 4.2.5). The VALBIN passenger boarding token provides limited gain at the current data scale, with its contribution expected to grow with longer training corpora and denser ticketing records.

6.2 Answers to the Research Questions

RQ1 is positively answered. Factorized tokenization successfully encodes the sequential and causal structure of transport operations: stop-sequence position exceeds 97% accuracy and stop identity reaches 94.5%, the delay token reaches 56.0% top-1 (96.4% top-5 at 30-second bin resolution, approximately 4.3 times above the most-frequent-bin baseline), and monotonically ordered trajectories are generated in rollout without explicit structural training. The ablation confirms that temporal resolution is the most critical design parameter, with the hybrid one-minute TBIN + delta encoding increasing overall test NLL by 0.039 while reducing DBIN-only NLL by 0.109 (9.8%) relative to the Full baseline on the same canonical corpus (Section 5.6). The factorized vocabulary enables generalization to infrequently observed route segments: a rare stop's embedding is informed by all events at that stop regardless of delay or time context, avoiding the combinatorial vocabulary and data-hungry representations that flat encoding would require [18].

RQ2 is answered at the trip level. The EventGPT generates plausible and coherent trip-level event-sequence continuations. Delay propagation over a 6-stop horizon is captured realistically in typical operating conditions, as illustrated by the qualitative examples in Tables 5.7 and 5.8 and by the structural quality metrics in Table 5.6. The near-perfect structural validity (100% complete event rate, 99.9% temporal monotonicity) demonstrates that the model has internalized the physical and operational constraints of a bus trip, namely, that time advances and stops are visited in order, purely from the statistical regularities of the training data, without any explicit constraint in the loss function. Failure cases arise primarily from abrupt external disruptions, an inherent limitation of any model that lacks real-time traffic data rather than a failure of the LEM paradigm per se. Full network-level dynamics, including bunching cascades and system-level disruption propagation, are not modeled in the current architecture: the HDWAY token provides a single-lag cross-trip signal encoding the gap to the immediately preceding vehicle, but interactions among all concurrently active trips remain unrepresented. Extending the LEM to this multi-trip scope is identified as primary future work in Section 6.3.

RQ3 is partially answered. In GTFS-guided delay-only mode (prefix=20, horizon=3), EventGPT surpasses all ten evaluated baselines, with a 5.8% improvement in delay MAE over the best-performing Historical Average (0.49 vs. 0.52 min). In unconstrained rollout mode (prefix=8, horizon=6), the model generates full event-sequence simulations spanning stop identity, arrival times, passenger counts, and headways, but this mode operates under a different protocol from the discriminative baselines, which were all assessed under the GTFS-guided prefix=20 setting. No like-for-like point-accuracy comparison is therefore available for unconstrained mode. The key practical advantage of the generative approach, across both modes, is the ability to produce full event-sequence simulations, enabling delay propagation analysis, multi-step forecasting, and counterfactual generation, capabilities unavailable in any discriminative baseline. The GTFS-guided mode is therefore the appropriate deployment configuration for point-accuracy comparison, the unconstrained mode is best understood as a simulation capability rather than a competing ETA predictor.

The overarching hypothesis of the dissertation, that representing public transport operations as tokenized event sequences and modeling them through autoregressive LEMs enables realistic trip-level event-sequence generation and competitive delay prediction under in-distribution operating conditions, is **confirmed**, with the qualification that the GTFS-guided mode is required to achieve competitive delay accuracy in the current configuration. The results collectively suggest that the LEM paradigm, originally developed for sports analytics [18], can be adapted to the structurally different but equally sequential domain of public transport operations, indicating a promising research direction for other complex event-driven systems.

6.3 Limitations and Future Work

The results of this dissertation point to the following research directions.

Longer training corpora and seasonal coverage. The model was trained on three months of data covering the winter operating period. Extending to a full year would expose it to seasonal demand cycles, public events, school holidays, and weather-induced disruptions, the scenarios where the LEM's flexible contextual conditioning is expected to provide the greatest advantage over static baselines. This is the most impactful near-term improvement.

Additional tokenization ablations. The original experimental design proposed two ablation axes not executed in the final scope: varying the context window length $L \in \{128, 256, 512\}$ to quantify the coverage–cost tradeoff for long-haul routes, and comparing stop-level event granularity against finer-grained GPS-derived events. Both remain natural next experiments, particularly context window scaling, where the quadratic attention cost makes the tradeoff non-trivial.

Multi-operator foundation models. The multi-operator experiments show that 1.82M parameters are insufficient for cross-operator generalization. Scaling to models with 10–100M parameters, feasible on modern GPU hardware, may enable a genuine public transport foundation model, analogous to the foundation LEM for soccer [18]. Operator-specific fine-tuning following pre-training on a large multi-operator corpus is a complementary strategy to reduce per-operator data requirements.

Richer event vocabulary. The current vocabulary contains only one event type. Adding explicit tokens for vehicle departures (enabling dwell-time modeling), service disruptions, bus bunching, and operator hold interventions would substantially enrich the simulation capability and allow the model to represent and predict system-level anomalies.

Real-time streaming deployment. In operational deployment, the model would process events incrementally as they arrive from the GPS/AVL feed, maintaining a rolling context window updated with each observed stop event. Integrating the EventGPT into a streaming inference architecture, with KV-cache optimization and latency constraints, is the necessary engineering step toward passenger-facing ETA predictions and operator-facing simulation dashboards.

Incorporating real-time external signals. With no access to other real-time signals like traffic conditions, weather, or special event indicators, the present EventGPT just uses the observed event sequence as context. The model's predictive horizon would be extended to include disruption

scenarios that cannot be inferred from the GPS event stream alone by incorporating these signals as additional token fields, such as a discretized traffic congestion indicator derived from a traffic data API or a meteorological condition token. While maintaining the compositionality and generative qualities of the token-based representation, such a multimodal event vocabulary would bring the EventGPT closer to the diverse context accessible to skilled human dispatchers.

Hierarchical and network-level simulation. The trip-level scope of the current architecture is a deliberate design boundary: establishing that the LEM paradigm works reliably for individual bus trips is the necessary precondition for composing those trip models into a network-level simulation. The HDWAY token already provides a single-lag cross-trip signal encoding the inter-vehicle gap to the immediately preceding bus on the same route, so the model is not fully trip-isolated, but it does not represent interactions among all concurrently active trips. In actuality, the mutual interaction of vehicle dynamics throughout the network causes bus bunching, which occurs when two vehicles on the same route arrive at a stop in close succession. In order to learn the patterns of interaction between trips and forecast emerging phenomena such as bursting cascades and propagation of disruption, a network-level extension of LEM would represent the joint event stream of all concurrently active trips. This improvement would provide simulation capabilities that are essentially beyond those of any single-trip discriminative model, but it would necessitate a far more intricate tokenization system and a wider context window.

Counterfactual simulation and scenario analysis. One of the most practically valuable capabilities of a generative model is counterfactual analysis: given a modified initial condition or operational intervention, what would the resulting network state be? An operator might ask, for example, how a bus that leaves the terminal three minutes early spreads to stops downstream. By altering the initial context sequence, the autoregressive rollout process naturally facilitates counterfactual prompting, a feature that discriminative models, which generate a single point estimate for a fixed input, essentially cannot duplicate. In order to show the operational value of the LEM paradigm beyond conventional prediction accuracy metrics, it is crucial to develop and assess this counterfactual capability. This would be a tangible step toward utilizing the model as a planning and decision-support tool for transit operators.

Passenger-demand-aware dwell time modeling. Dwell times, or how long a bus door stays open at a stop, are not explicitly predicted by the existing model, which encodes passenger boarding counts as a discrete VALBIN token. The amount of passengers boarding and alighting directly affects dwell time, which is a major cause of delay buildup in high-demand metropolitan networks. The model could learn the causal chain from VALBIN to DWELL to DBIN by adding an explicit DWELL token that represents the dwell duration in seconds. This will significantly enhance the model's capacity to assess operational solutions like all-door boarding or pre-paid ticket collection, which mainly function by cutting dwell time, and to simulate demand-driven delay propagation.

Transfer learning from larger corpora. Similar to fine-tuning a pre-trained language model on a domain-specific corpus, the EventGPT's compact architecture makes it a candidate for initialization from a larger pre-trained model. Pre-trained on aggregated GPS and GTFS data from numerous cities and operators over a number of years, a transport-specific foundation model could

offer a rich initialization for operator-specific fine-tuning, significantly lowering the quantity of per-operator data needed to achieve competitive performance. For operators with limited historical data (such as recently launched services or operators in data-scarce environments), where training a model from scratch on three months of data may not be sufficient to capture the full range of operational dynamics, this transfer learning direction is especially pertinent.

In conclusion, this dissertation shows that the Large Events Model framework is a workable and promising method for simulating and forecasting public transportation. The event-token paradigm directly addresses the structural gap found in the state-of-the-art review by offering a unified representation that connects trip-level event-sequence generation with delay prediction within a single generative model. While the limits and future goals mentioned above map out a clear research agenda to extend this work to a production-ready public transport intelligence system, the experimental results offer a quantitative basis for this assertion.

References

[1] Tom Brown et al. “Language Models are Few-Shot Learners”. In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901.

[2] Oded Cats. “Determinants of bus riding time deviations: Relationship between driving patterns and transit performance”. In: *Journal of Transportation Engineering Part A: Systems* 145.1 (2019). DOI: 10.1061/JTEPBS.0000201.

[3] Xiaoxu Chen et al. “Conditional forecasting of bus travel time and passenger occupancy with Bayesian Markov regime-switching vector autoregression”. In: *Transportation Research Part B: Methodological* 192 (2025). DOI: 10.1016/j.trb.2024.103147.

[4] Cynthia C.T. Cheok, Wooi Chen Khoo, and Hooiling Khoo. “Bus Travel Time Variability Modelling Using Burr Type XII Regression: A Case Study of Klang Valley”. In: *KSCE Journal of Civil Engineering* 28.9 (2024), pp. 3998–4009. DOI: 10.1007/s12205-024-2295-6.

[5] B. Dhivya Bharathi et al. “Bus travel time prediction: a log-normal auto-regressive (AR) modelling approach”. In: *Transportmetrica A: Transport Science* 16.3 (2020), pp. 807–839. DOI: 10.1080/23249935.2020.1720864.

[6] Kun Fu et al. “CompactETA: A Fast Inference System for Travel Time Prediction”. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Association for Computing Machinery, 2020, pp. 3337–3345. DOI: 10.1145/3394486.3403386.

[7] Zengyang Gong et al. “SD-seq2seq: A Deep Learning Model for Bus Bunching Prediction Based on Smart Card Data”. In: *Proceedings - International Conference on Computer Communications and Networks, ICCCN*. Vol. 2020-August. 2020. DOI: 10.1109/ICCCN49398.2020.9209686.

[8] Bin Hu et al. “A Bayesian Spatiotemporal Approach for Bus Speed Modeling”. In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. 2019, pp. 497–502. DOI: 10.1109/ITSC.2019.8917523.

[9] Zhao Huang et al. “A Novel Bus-Dispatching Model Based on Passenger Flow and Arrival Time Prediction”. In: *IEEE Access* 7 (2019), pp. 106453–106465. DOI: 10.1109/ACCESS.2019.2932801.

[10] Suhyun Jeong, Changsong Oh, and Jongpil Jeong. “BAT-Transformer: Prediction of Bus Arrival Time with Transformer Encoder for Smart Public Transportation System”. In: *Applied Sciences (Switzerland)* 14.20 (2024). DOI: 10.3390/app14209488.

[11] Renhe Jiang et al. “DeepUrbanEvent: A System for Predicting Citywide Crowd Dynamics at Big Events”. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Association for Computing Machinery, 2019, pp. 2114–2122. DOI: 10.1145/3292500.3330654.

[12] Osman Kaya and Mustafa Utku Kalay. "Spatio-Temporal Forecasting of Bus Arrival Times Using Context-Aware Deep Learning Models in Urban Transit Systems". In: *IEEE Access* 13 (2025), pp. 161423–161435. DOI: 10.1109/ACCESS.2025.3609530.

[13] Alkilane Khaled et al. "A graph-based approach for traffic prediction using similarity and causal relations between nodes". In: *Knowledge-Based Systems* 296 (2024). DOI: 10.1016/j.knosys.2024.111913.

[14] Changlin Li et al. "A Sequence and Network Embedding Method for Bus Arrival Time Prediction Using GPS Trajectory Data Only". In: *IEEE Transactions on Intelligent Transportation Systems* 24.5 (2023), pp. 5024–5038. DOI: 10.1109/TITS.2023.3237320.

[15] Chuangchieh Lin et al. "A comparative study and simple baseline for travel time prediction". In: *Scientific Reports* 15.1 (2025). DOI: 10.1038/s41598-025-02303-5.

[16] Zhenliang Ma et al. "Estimation of trip travel time distribution using a generalized Markov chain approach". In: *Transportation Research Part C: Emerging Technologies* 74 (2017), pp. 1–21. DOI: 10.1016/j.trc.2016.11.008.

[17] Vladimir Mashurov et al. "Gct-TTE: graph convolutional transformer for travel time estimation". In: *Journal of Big Data* 11.1 (2024). DOI: 10.1186/s40537-023-00841-1.

[18] Tiago Mendes-Neves, Luís Meireles, and João Mendes-Moreira. "Towards a foundation large events model for soccer". In: *Machine Learning* 113.11 (2024), pp. 8687–8709. DOI: 10.1007/s10994-024-06606-y.

[19] Victor Jian Ming Low, Hooiling Khoo, and Wooi Chen Khoo. "On the prediction of intermediate-to-long term bus section travel time with the Burr mixture autoregressive model". In: *Transportmetrica A: Transport Science* 20.3 (2024). DOI: 10.1080/23249935.2023.2181023.

[20] Aidan O'Sullivan et al. "Uncertainty in Bus Arrival Time Predictions: Treating Heteroscedasticity with a Metamodel Approach". In: *IEEE Transactions on Intelligent Transportation Systems* 17.11 (2016), pp. 3286–3296. DOI: 10.1109/TITS.2016.2547184.

[21] Charul Paliwal and Pravesh Biyani. "To each route its own ETA: A generative modeling framework for ETA prediction". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. 2019, pp. 3076–3081. DOI: 10.1109/ITSC.2019.8917465.

[22] Dancho Panovski and Titus Zaharia. "Long and Short-Term Bus Arrival Time Prediction With Traffic Density Matrix". In: *IEEE Access* 8 (2020), pp. 226267–226284. DOI: 10.1109/ACCESS.2020.3044173.

[23] Ting Qiu et al. "FEN-MRMGCN: A Frontend-Enhanced Network Based on Multi-Relational Modeling GCN for Bus Arrival Time Prediction". In: *IEEE Access* 13 (2025), pp. 5296–5307. DOI: 10.1109/ACCESS.2024.3525357.

[24] Alec Radford et al. "Language Models are Unsupervised Multitask Learners". In: *OpenAI Blog* 1.8 (2019), p. 9.

[25] Narges Rashvand et al. "Real-Time Bus Departure Prediction Using Neural Networks for Smart IoT Public Bus Transit". In: *IoT* 5.4 (2024), pp. 650–665. DOI: 10.3390/iot5040029.

[26] Felix Schwinger et al. "Combining Profile Similarity and Kalman Filter for Real-World Applicable Short-Term Bus Travel Time Prediction". In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. 2021, pp. 3738–3745. DOI: 10.1109/ITSC48978.2021.9564421.

[27] Jinxing Shen et al. “A novel model incorporating deep learning and Kalman filter augmentation for route-level bus arrival time prediction with error accumulation mitigation”. In: *Expert Systems with Applications* 281 (2025). DOI: 10.1016/j.eswa.2025.127622.

[28] Ashish Vaswani et al. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017.

[29] Pengfei Wang et al. “A two-stage method for bus passenger load prediction using automatic passenger counting data”. In: *IET Intelligent Transport Systems* 15.2 (2021), pp. 248–260. DOI: 10.1049/itr2.12018.

[30] Ruibin Xiong et al. “On Layer Normalization in the Transformer Architecture”. In: *Proceedings of the 37th International Conference on Machine Learning*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 10524–10533.

[31] David de Zea-Graells et al. “Complex seasonality in bus urban mobility: assessment of individual and ensemble prediction models”. In: *Transportmetrica B* 13.1 (2025). DOI: 10.1080/21680566.2025.2490510.

[32] Jian Zhang et al. “Method of predicting bus arrival time based on MapReduce combining clustering with neural network”. In: *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*. 2017, pp. 296–302. DOI: 10.1109/ICBDA.2017.8078828.

[33] Gang Zhong et al. “Bus Travel Time Prediction Based on Ensemble Learning Methods”. In: *IEEE Intelligent Transportation Systems Magazine* 14.2 (2022), pp. 174–189. DOI: 10.1109/MITS.2020.2990175.