UNIVERSITÀ DEGLI STUDI DI GENOVA

POLYTECHNIC SCHOOL

DITEN

Department of Naval, Electrical, Electronic and Telecommunications Engineering

MASTER'S THESIS IN

Engineering Technology for Strategy & Security

Driving Scenario Recognition Using Spatio-Temporal Transformers on Vehicular Signals and Traffic Time Series

Supervisors:

Prof. Francesco Bellotti

Prof. Riccardo Berta

Co-supervisor:

Dr. Alessandro Pighetti

Candidate:

Luca Benvenuto

July 2026

UNIVERSITÀ DEGLI STUDI DI GENOVA

SCUOLA POLITECNICA

DITEN

Dipartimento di Ingegneria Navale, Elettrica,

Elettronica e delle Telecomunicazioni

TESI DI LAUREA MAGISTRALE IN

Engineering Technology for Strategy & Security

Riconoscimento di scenari di guida simulati un approccio basato su Transformer spazio-temporali per l'analisi di serie temporali veicolari e del traffico

Relatore:

Prof. Francesco Bellotti

Prof. Riccardo Berta

Co-relatore:

Dr. Alessandro Pighetti

Candidato:

Luca Benvenuto

Luglio 2026

Driving Scenario Recognition Using Spatio-Temporal Transformers on Vehicular Signals and Traffic Time Series

Abstract

Automatic driving scenario recognition is a core capability for advanced driver-assistance systems (ADAS) and autonomous vehicles. Most scenario-defining events, such as cut-ins, braking, and gap-closing, are inherently relational: their recognition requires joint modelling of multiple agents over time.

This thesis introduces a synthetic labelled benchmark of 31,502 scenario windows extracted from multi-agent CARLA simulations, encoded in the ASAM Open Simulation Interface (OSI) format and classified into nine categories in accordance with the ISO 34502 taxonomy. We propose a Spatio-Temporal Transformer (STT), a factorised attention architecture that alternates spatial multi-head attention across agent slots with temporal multi-head attention along the scenario timeline. An intermediate Ego Cross-Attention module distils multi-agent context into an ego-centric representation before temporal processing. The model is trained under a multi-task learning framework combining Focal Loss classification over the nine scenario classes with Huber Loss regression over three ego-vehicle kinematic targets, with task weights balanced via Kendall homoscedastic uncertainty weighting.

On the test partition, the STT achieves a macro-F1 of 0.732 ± 0.005 across three random seeds, matching the Flat Transformer (0.733 ± 0.020) and STGCN (0.730 ± 0.007) while exhibiting substantially lower seed-to-seed variance. The ego-only LSTM baseline achieves a macro-F1 of 0.444, falling 0.288 points below the interaction-aware models, confirming that multi-agent context is essential for relational scenario classes. The regression head evaluation demonstrates that ego-

centric kinematic targets are predicted with comparable accuracy by ego-only and multi-agent models, validating the multi-task design rationale.

Keywords. Driving scenario recognition, Spatio-Temporal Transformer, multi-task learning, CARLA simulation, Open Simulation Interface (OSI), ISO 34502.

Riconoscimento di scenari di guida simulati un approccio basato su Transformer spazio-temporali per l'analisi di serie temporali veicolari e del traffico

Sommario

Il riconoscimento automatico di scenari di guida è un componente fondamentale per i sistemi di assistenza alla guida avanzati (ADAS) e per i veicoli autonomi. La maggior parte degli eventi che definiscono uno scenario, quali tagli di corsia, frenate o avvicinamenti, è intrinsecamente relazionale: il loro riconoscimento richiede la modellazione congiunta del comportamento di più agenti nel tempo.

Questo lavoro introduce un benchmark sintetico etichettato di 31.502 finestre di scenario estratte da registrazioni multi-agente simulate in CARLA, codificate nel formato ASAM Open Simulation Interface (OSI) e classificate in nove categorie in accordo con la tassonomia ISO 34502. Viene proposto uno Spatio-Temporal Transformer (STT), un'architettura ad attenzione fattorizzata che alterna l'attenzione spaziale multi-testa tra gli slot degli agenti con l'attenzione temporale multi-testa lungo la sequenza temporale. Un modulo di Ego Cross-Attention intermedio distilla il contesto multi-agente in una rappresentazione ego-centrica prima dell'elaborazione temporale. Il modello è addestrato con un framework multi-task che combina Focal Loss per la classificazione e Huber Loss per la regressione su tre target cinematici del veicolo ego, con bilanciamento dei pesi tramite la ponderazione di incertezza omoschedastica di Kendall.

Sulla partizione di test, lo STT raggiunge un macro-F1 di 0,732 ± 0,005 su tre run con seed diversi, equiparando il Flat Transformer (0,733 ± 0,020) e lo STGCN (0,730 ± 0,007) con una varianza seed-to-seed nettamente inferiore. La baseline ego-only LSTM registra un macro-F1 di 0,444, inferiore di 0,288 punti rispetto ai modelli interaction-aware, confermando che il contesto multi-agente è indispensabile per le classi di scenari a base relazionale. La valutazione della testa di regressione dimostra

che i target cinematici ego-centrici sono predetti con precisione comparabile da modelli ego-only e multi-agente, validando la razionale del design multi-task.

Parole chiave. Riconoscimento di scenari di guida, Spatio-Temporal Transformer, apprendimento multi-task, CARLA, Open Simulation Interface (OSI), ISO 34502.

List of Figures

List of Table

Acknowledgment

I would like to thank Prof. Francesco Bellotti and Prof. Riccardo Berta for their availability and for their support throughout the internship in their lab. Special thanks go to Dr. Alessandro Pighetti for his hands-on help and guidance throughout this journey. I would also like to thank the ELIOS Lab for providing the resources and environment necessary for carrying out this work.

Table of Contents

1. Introduction

Advanced driver-assistance systems (ADAS) and autonomous vehicles must continuously interpret the dynamic environment around them to anticipate hazards, adapt their behaviour, and satisfy regulatory safety requirements.[1] A critical component of this interpretation is driving scenario recognition, the ability to automatically identify the type of multi-agent interaction unfolding around the ego vehicle from kinematic time-series observations.

Standard taxonomies such as ISO 34502, [2] formalise a set of recurrent scenario categories , braking, lane-change, cut-in, cut-out, and following , that collectively cover the most safety-relevant highway traffic situations. Reliable recognition of these categories in real time would enable an ADAS to select the appropriate response strategy, trigger targeted data logging for scenario-specific model refinement, and support compliance verification with type-approval test requirements.

The key challenge is that most scenario-defining events are fundamentally relational: a cut-in is characterised by the lateral trajectory of a neighbouring vehicle relative to the ego lane, not by any property of the ego vehicle alone. A recognition system must therefore jointly model the behaviour of multiple agents over time , a spatio-temporal problem over a variable-size, permutation-sensitive set of actors. [3]

Obtaining large-scale annotated real-world recordings for supervised training is expensive and logistically demanding. [4] Simulation environments such as CARLA provide a controlled and reproducible alternative: traffic scenarios can be scripted to specification, executed at scale, and automatically labelled at the frame level, without the legal and privacy constraints that encumber naturalistic driving data. The Open Simulation Interface (OSI) standard provides [2] a vendor-neutral format for exchanging simulation ground-truth data [5], enabling interoperability across toolchains and easing the transition from simulation to real-world evaluation.

1.1 Research objectives

This thesis pursues the following research objectives:

O1. Construct a synthetic labelled benchmark of driving scenario windows by generating multi-agent CARLA simulations, extracting and labelling scenario windows according to the ISO 34502 taxonomy, and encoding the resulting dataset in the ASAM OSI / MCAP format.

O2. Design a spatio-temporal Transformer architecture (STT) that can jointly model spatial interactions among agents and temporal dynamics along the scenario timeline, without assuming a fixed ordering of agents or a fully joint attention across the combined spatio-temporal token set.

O3. Train and evaluate the STT under a multi-task learning framework that combines a primary classification objective over nine scenario classes with an auxiliary regression objective over ego-vehicle kinematic targets, using homoscedastic uncertainty weighting to balance the two losses.

O4. Benchmark the STT against a set of competitive baselines spanning the spectrum from ego-only models (LSTM-ego, SVM) to full multi-agent architectures (Flat Transformer, STGCN) and characterise the contribution of multi-agent context to scenario recognition accuracy.

1.2 Thesis Contribution

The work described in this thesis delivers two primary contributions to the field of data-driven driving scenario recognition.

The first contribution is a **synthetic labelled benchmark** comprising 73,829 CARLA-simulated multi-agent scenario recordings covering nine ISO 34502 scenario classes. Each recording is encoded in the ASAM Open Simulation Interface format and packaged as an MCAP file, providing standardised, reproducible access to multi-agent kinematic ground-truth data. The benchmark is partitioned into training, validation,

and test splits totalling 31,502 test windows, and includes class-balanced sampling metadata to address the pronounced class imbalance in the raw recording distribution.

The second contribution is the Spatio-Temporal Transformer (STT), a factorised attention architecture for multi-agent scenario recognition. The STT alternates a spatial multi-head self-attention layer , which relates agent tokens at each individual timestep , with a temporal multi-head self-attention layer , which models the evolution of the ego-centric context vector over time. An intermediate Ego Cross-Attention module distils the spatially-attended multi-agent representation into a single ego-centric token before the temporal encoder, reducing the temporal sequence length from K×T to T. The STT is trained under a multi-task learning framework with Focal Loss for classification and Huber Loss for kinematic regression, with task weights learned via Kendall homoscedastic uncertainty weighting. On the test partition, the STT achieves a macro-F1 of 0.732 ± 0.005 across three random seeds, matching the Flat Transformer and STGCN while exhibiting lower seed-to-seed variance.

Figure 1.1 provides a high-level overview of the two-component system developed in this thesis: the Scenario Generator, which produces labelled OSI-encoded scenario windows from CARLA simulations, and the Scenario Detector, which classifies incoming windows using the trained STT model.

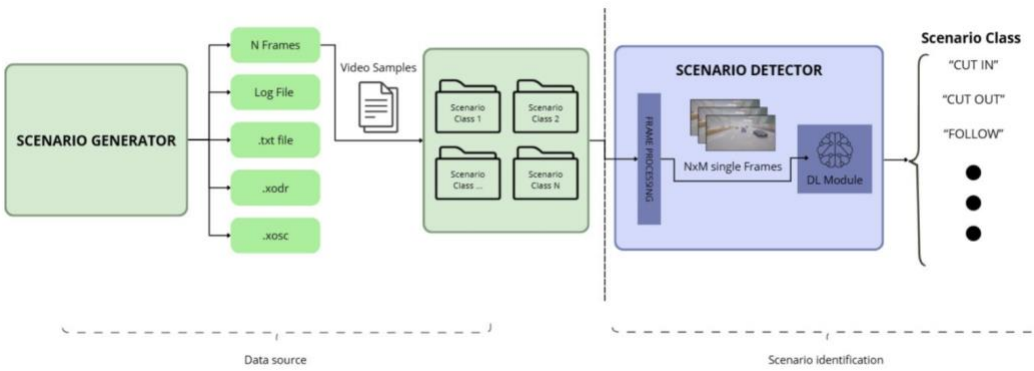

Figure 1. 1 System overview. The Scenario Generator (left) processes raw CARLA simulator recordings through the OSI conversion pipeline to produce standardised labelled scenario windows. The Scenario Detector (right) applies the trained Spatio-Temporal

1.3 Structure of thesis

This chapter motivates the research problem, states the specific objectives pursued in this thesis, and summarises the main contributions. Chapter 2 reviews the relevant background in driving scenario recognition, spatio-temporal modelling, and multi-task learning. Chapter 3 describes the synthetic dataset and its construction pipeline. Chapter 4 details the proposed Spatio-Temporal Transformer architecture and training procedure. Chapter 5 presents the experimental evaluation, and Chapter 6 concludes with a discussion of limitations and directions for future work.

2. Related work

This chapter surveys the background literature across three areas that constitute the technical foundations of this thesis.

2.1 Driving scenario recognition and characterization

The ability to automatically recognise and classify driving scenarios is a foundational capability for advanced driver-assistance systems (ADAS) and autonomous vehicles. A driving scenario, in the sense used throughout this thesis, is a short temporal segment of highway traffic in which the ego vehicle and a set of surrounding agents engage in a characteristic interaction pattern as lane change, cut-in, following, braking, and so forth. Reliable recognition of these patterns is prerequisite to downstream tasks such as risk assessment, adaptive cruise control, and post-hoc evaluation of simulation campaigns.

Early approaches to scenario recognition relied on explicitly engineered features and rule-based decision logic. Kinematic quantities such as headway, time-to-collision, and lane-relative lateral offset were combined through hand-crafted thresholds to trigger categorical labels [6], [7]. While interpretable, such methods are brittle under sensor noise and do not generalise across map geometries or traffic densities.

Machine learning approaches replaced hard rules with statistical models. Hidden Markov Models were among the first to treat manoeuvre recognition as a sequence classification problem, exploiting the Markovian structure of lane-change dynamics [8]. Support vector machines trained on hand-engineered kinematic feature vectors achieved competitive results on structured highway scenarios but required careful feature selection and scaled poorly to multi-agent settings. Xing et al. adopted recurrent neural networks with LSTM cells to model time-series driving behaviour and infer lane-change intention from vehicle sensor signals, demonstrating that learned temporal representations outperform hand-crafted features [9],[10]. Dörr further confirmed that time-series-based machine learning methods consistently surpass rule-

based classifiers on multiclass manoeuvre datasets, though they noted that class imbalance and variable-length segments remain open challenges that existing pipelines address only partially [11].

The central limitation shared by these approaches is that each vehicle's trajectory is treated as an independent time series, whereas in practice the label of a scenario is defined by the interaction between the ego vehicle and its neighbours: a Braking Leading Vehicle scenario, for instance, is intrinsically relational, requiring the model to simultaneously attend to the ego's kinematic state and the rapid deceleration of the leading vehicle. Ego-centric scene understanding, in which the ego vehicle acts as the query and the surrounding agents provide context, has accordingly emerged as a natural paradigm for this family of tasks [10]. The model proposed in this thesis addresses this gap by computing a spatial attention mechanism over all agents at each timestep with a temporal attention mechanism over the resulting ego-centric representation, yielding a unified spatio-temporal encoder tailored to multi-agent highway scenarios.

2.2 Transformer Architectures for Sequence Modelling

The Transformer architecture, introduced by Vaswani et al. in 2017, replaced recurrence and convolution with scaled dot-product multi head self-attention as the primary mechanism for modelling sequential dependencies [12]. The core operation computes, for each element in the sequence, a weighted sum of all other elements through learned query, key and value projections. This design confers three properties that are critical for the scenario recognition task: first, all pairwise interactions within a sequence are computed in parallel, enabling efficient training on long windows; second, the attention score between any two positions is computed directly regardless of their distance, avoiding the vanishing gradient problem that limits RNNs and LSTMs on long sequences; and third, the model is order-agnostic by construction, relying on an explicit positional encoding to reintroduce sequence order information.

For the temporal dimension of the model, positional information is injected using the sinusoidal encoding proposed in the original transformer paper. Each position index (t) is mapped to a (d) dimensional vector through alternating sine and cosine functions at exponentially spaced frequencies. This deterministic encoding has the desirable property of generalising to sequence lengths unseen during training. The survey by Church et al. (2025) provides a comprehensive analysis of positional encoding strategies for time-series transformers and confirms that sinusoidal encoding remains competitive when the sequence length distribution is heterogeneous and the training set is finite, as is the case here[13].

The effectiveness of self-attention for time series has been extensively documented. The survey by Wen et al. reviews transformer variants across time series forecasting, classification and anomaly detection, identifying the self-attention mechanism as the principal contributor to performance gains over LSTM baselines on irregular and long-horizon time series [14]. The survey by Foumani on deep learning for time series classification reports that transformer-based architectures consistently outperform convolutional and recurrent architectures when the sequence exhibits multi-scale temporal dependencies and the number of classes is large [15]. PatchTST further showed that partitioning the time series into non-overlapping patches before applying self-attention substantially improves representational efficiency, as each patch aggregates local context before global attention is applied [16]. Although this thesis does not use patch tokenisation (the temporal sequences are already short and the per-timestep representation already encodes rich spatial context from the preceding spatial encoder stage), the theoretical argument from PatchTST supports the use of attention as the temporal pooling mechanism rather than mean aggregation or recurrence.

Within the classification head, a Temporal Attention pooling layer replaces the common alternative of simply taking the last hidden state or computing a uniform mean over timesteps. For variable duration scenarios, the informative frames are not uniformly distributed: the peak kinematic event may occur at any point within the window. The pooling layer assigns a learned scalar weight to each timestep through a

two-layer MLP followed by a masked softmax, ensuring that uninformative padding frames receive near-zero weight and that the final representation is dominated by the most discriminative moments of the scenario. This design is consistent with the self-attention pooling mechanisms used in sequence-level classification tasks in NLP and audio [17], and is a natural extension of the transformer's attention mechanism to the aggregation stage.

The choice of encoder-only (bidirectional) versus decoder-only (causal) attention is architecturally significant for classification tasks. Decoder-only models such as GPT apply a causal mask that prevents each position from attending to future timesteps, a constraint motivated by autoregressive generation but unnecessary and restrictive for discriminative classification, where the full sequence is available at inference time. Encoder-only models such as BERT apply bidirectional attention, allowing each position to integrate information from both past and future timesteps, which is directly beneficial for scenario recognition: the discriminative signal for a cut-in event may appear in the lateral velocity trajectory of an adjacent vehicle several timesteps before or after the most salient ego-vehicle response. The STT therefore uses bidirectional self-attention in both the spatial and temporal stages, with no causal masking applied.

Efficient attention variants such as Linformer [18] and Performer [19]approximate the softmax attention matrix with low-rank factorizations, reducing the quadratic memory cost of standard attention to linear in the sequence length. These approximations are motivated primarily by very long sequences (thousands to tens of thousands of tokens). In the setting of this thesis, temporal sequences span at most 110 timesteps and spatial attention is computed over at most five agent slots; the computational cost of exact softmax attention is negligible, and introducing approximation errors is undesirable given the modest sequence lengths. Standard scaled dot-product attention is therefore used throughout, consistent with the TimeSformer and s2TNet architectures reviewed in Section 2.3.

Pre-training on large unlabelled corpora followed by task-specific fine-tuning , the paradigm established by BERT and GPT-2 and later extended to time series by models

such as TimesFM [20] and MOMENT [21] , offers a potential avenue for improving generalisation when annotated data is scarce. In the context of this thesis, the labelled dataset of 31,502 windows is sufficient to train the STT from random initialisation without pre-training. Nevertheless, the availability of large unlabelled CARLA corpora suggests that a masked autoencoding pre-training stage, analogous to MAE for images or TimeMAE for time series, could improve performance on the hardest classes (CINF, COUTF) by providing the encoder with a richer feature initialisation before supervised fine-tuning. This direction is identified as a priority for future work in Section 6.3.

2.3 Spatio-temporal Transformer

Traffic scene modelling requires representing two distinct relational structures simultaneously: the spatial relationship among agents at each individual timestep, and the temporal evolution of each agent's state across successive timesteps. The challenge is that both structures are dynamic: the set of present agents varies between recordings, and the relevant spatial relationships change within a scenario, and the two dimensions are not independent since the temporal dynamic of one agent influence and re influenced by those of all others.

The canonical prior work on spatio-temporal modelling for traffic is the Spatio-Temporal Graph Convolutional Network (STGCN) proposed by Yu et al., which represents agents as nodes in a graph and applies graph convolutions for spatial aggregation interleaved with gated temporal convolutions . STGCN established that joint modelling of the two dimensions outperforms temporal-only or spatial-only baselines, and it remains a strong reference architecture in the literature. However, graph convolutional approaches require a fixed or explicitly constructed adjacency matrix, which imposes a strong inductive bias on which agent-agent connections are informative. For the scenario recognition setting, the relevant spatial relationships are not known a priori: a cut-in (CINF) scenario requires attending to a laterally adjacent vehicle, while a braking scenario (BF) requires attending to the longitudinally

preceding vehicle. Constructing a single fixed graph that captures both interaction types uniformly is non-trivial and may introduce spurious edges that dilute the attention signal.

Factorised spatio-temporal attention addresses this limitation by decoupling spatial and temporal attention into separate, alternating stages. Bertasius et al. demonstrated this principle for video understanding: their TimeSformer model applies spatial self-attention across patch tokens within each frame, followed by temporal self-attention across corresponding positions over time, achieving state of the art performance on action recognition benchmarks while requiring substantially fewer floating-point operations than joint space-time attention. The key theoretical insight is that this factorisation reduces the computational cost from:

$$O[(T \cdot K)^2] \quad \text{to} \quad O[TK^2 + KT^2]$$

where T is the number of times stamps, and K is the number of agents [22]. The s2TNet architecture applies an analogous factorised design to trajectory prediction, confirming that the decomposition generalises from image grids to unstructured agent sets, while the STGformer and the graph transformer of Zhou et al. further extend this approach to large-scale traffic forecasting, reporting consistent improvements over STGCN-family baselines [23], [24].

The spatial encoder in the proposed model applies multi-head self-attention over the k vehicle slots at each timestep independently, training each vehicle a token without imposing any a priori graph structure. This design is permutation-invariant with respect to the ordering of non-ego vehicles: the spatial attention weights are learned end to end, allowing the model to discover which agent relationships are most predictive for each class. Vehicle slots are disambiguated from the ego vehicle through a learned vehicle embedding (an index-based lookup added to the input projection), which provides the model with ordinal slot information without constraining the semantic role of any non-ego slot.

Following the spatial encoder, an Ego Cross-Attention layer extracts the ego-centric representation for each timestep. The ego vehicle token serves as the query, while the full set of agents' tokens, including the ego, serves as keys and values. This asymmetric cross-attention design formalises the ego-centric perspective: the output is a representation of the ego vehicle conditioned on the entire spatial context, capturing which surrounding agents are most salient for understanding the ego's current situation. The ego centric paradigm is theoretically motivated by the definition of the scenario labels themselves, all of which are described relative to the EV. Empirically, cross-attention-based ego conditioning has been adopted in recent multi-agent prediction models to achieve selective, context-aware feature aggregation without requiring explicit pairwise feature concatenation [25]. The ego cross-attention output is combined with the masked mean of all agent representations, fusing the ego-query perspective with a global scene summary.

Stochastic Depth [26] is applied as a regularisation technique at every encoder layer, with drop rate increasing linearly from the first to the last layer [27]. During training, each layer is randomly bypassed with a probability proportional to its depth, allowing the model to adaptively adjust its effective depth per mini batch. The linearly increasing schedule concentrates the regularisation pressure on the deeper layers, which are more prone to overfitting, while preserving the integrity of the early feature extraction stages. This is particularly relevant given the relatively modest dataset size compared to the model capacity (approximately 9.5 million trainable parameters).

Table 2.1 summarises the spatio-temporal architectures discussed above, highlighting for each the spatial and temporal modelling mechanism, its application domain, and the limitation that motivates the design choices adopted in this thesis.

Method	Spatial / temporal mechanism	Domain	Gap addressed
STGCN	Graph convolution (fixed adjacency) + gated temporal convolution	Traffic forecasting	Needs a fixed graph; agent-agent relevance not known a priori for scenario recognition
TimeSformer	Spatial attention over patches per frame, then temporal attention across positions	Video / action recognition	Built for regular image-patch grids, not unstructured agent sets
s2TNet	Factorised spatial + temporal attention over agents	Trajectory prediction	Confirms factorisation generalises to agent sets (no citation no. found , verify)

STGformer	Graph-based spatial attention + transformer temporal modelling	Large-scale traffic forecasting	Improves on STGCN-family, but still graph-based, network-scale
Graph Transformer	Heterogeneous graph attention + transformer temporal modelling	Large-scale traffic forecasting	Improves on STGCN-family, but still relies on a road-network graph
This thesis	Self-attention over vehicle slots (no fixed graph) + Ego Cross-Attention + temporal transformer with attention pooling	Highway scenario recognition	No fixed graph needed; explicit ego-centric attention matches the ego-relative scenario labels

Table 2. 1 Comparison of spatio-temporal modelling approaches for multi-agent traffic and video understanding tasks.

2.4 Multi-Task Learning and Training Strategies

Multi-task learning (MTL) is a training paradigm in which a shared encoder is simultaneously optimised for multiple related objectives, with the objective of improving generalisation through the joint learning of complementary representations [30]. The key theoretical argument is that auxiliary tasks act as inductive biases, constraining the hypothesis space to solutions that explain multiple data sources at once. For the scenario recognition setting, the three auxiliary regression targets, peak lateral velocity of the ego vehicle, mean ego speed, and peak ego deceleration, are physically meaningful quantities that are directly observable from the input features and that vary systematically across scenario classes. The intuition is that a model which correctly predicts, for instance, that peak lateral velocity is high is effectively forced to represent the ego's lateral motion explicitly, which is precisely the discriminative feature distinguishing LCF (lane change) from FLF_B3 (following). The auxiliary regression loss thus penalises representations that ignore kinematically relevant dimensions of the input, acting as a form of task-specific regularisation.

The weights assigned to the classification and regression losses are learned automatically using the homoscedastic uncertainty weighting method proposed by Kendall [31]. In the homoscedastic formulation, each task is associated with a learned scalar log-variance parameter $s_i = \log(\sigma_i^2)$. The combined loss becomes the sum over tasks of $(e^{-s_i} \cdot L_i + 0.5 \cdot s_i)$, where L_i is the task-specific loss. This formulation has a clear interpretation: if the model is uncertain about a task (large σ_i), the precision

weight e^{-s_i} is small and the task contributes less to the gradient; the regularisation term $0.5 \cdot s_i$ prevents the model from trivially suppressing all tasks by setting $\sigma_i \to \infty$. The key practical advantage over a fixed weighting hyperparameter such as lambda is that the relative task importance is learned from data: as the classification loss converges faster or slower than the regression loss during training, the uncertainty weights adapt accordingly. [32] Validated this approach on simultaneous regression and classification tasks in scene geometry estimation, showing that learned uncertainty weighting consistently outperforms grid-searched fixed weights [31]. This justification removes the need to tune a critical hyperparameter on the validation set.

The classification loss uses Focal Loss, introduced by Lin et al. for one-stage object detection [33]. Focal Loss modifies the standard cross-entropy by multiplying each sample's loss by $(1 - p_t)^{\gamma}$, where p_t is the predicted probability assigned to the correct class. When a sample is correctly classified with high confidence, p_t is close to 1 and the factor is close to 0, downweighing the loss contribution of easy examples. When the model makes a hard error (p_t small), the factor is close to 1, maintaining full gradient signal. With $\gamma = 1.5$, as used in this model, the loss concentrates learning on the minority classes and ambiguous boundary cases, a choice directly motivated by the data: as documented in Section 3.5, the dataset exhibits a four-fold class imbalance between the most frequent class (LCF, approximately 15,800 recordings) and the least frequent (BF, approximately 3,800 recordings). Standard cross-entropy assigns equal implicit importance per sample, causing the model to overfit on the majority classes. Focal Loss addresses this without requiring explicit prior knowledge of the class frequency, complementing the WeightedRandomSampler that rebalances the mini-batch composition at the data loading level. Class-frequency-derived weights are additionally passed to the Focal Loss as per-class multipliers, providing a further correction layer. The combination of balanced sampling, per-class weights, and focal modulation represents a defence-in-depth strategy against the class imbalance documented in the dataset.

MixUp augmentation is applied during training [34]. Given a pair of training samples (x_i, y_i) and (x_j, y_j), MixUp constructs a virtual example by interpolating both the input tensor and the one-hot label vectors with a mixing coefficient λ drawn from a Beta(α, α) distribution with α = 0.3. The resulting soft labels impose a Lipschitz continuity constraint on the decision boundary: the model must predict interpolated confidence scores for interpolated inputs, preventing overconfident, high-curvature decision boundaries in the feature space. Zhang et al. demonstrated that MixUp consistently reduces test error and improves calibration on classification benchmarks without requiring architectural changes or additional data [34]. In the MTL context, the regression targets for the mixed sample are taken from the dominant sample (the one with $\lambda \geq 0.5$), as the three regression targets peak lateral velocity, mean speed, and peak deceleration are not additively composable: the maximum of a linear combination is not the linear combination of the maxima. This conservative choice ensures that the regression supervision is always physically consistent.

The optimiser is AdamW with weight decay 2×10^{-2}, a decoupled L2 regularisation scheme that avoids the interaction between the adaptive learning rate scaling and the weight norm penalty present in standard Adam with L2 regularisation [35]. The learning rate follows a One-Cycle schedule, warming up from:

$$\frac{lr_{max}}{div\ factor} \text{ to } lr_{max}$$

over the first 30% of training steps (pct_start=0.3) and then annealing to

$$\frac{lr_{max}}{div\ factor} = 3\times10^{-7}$$

using a cosine curve.

Smith showed that a single large learning rate cycle followed by aggressive annealing achieves faster convergence and higher final accuracy than step-decay or cosine-without-warmup schedules, by combining the exploration benefits of a high peak learning rate with the fine-grained convergence of the low final rate [36]. Gradient clipping at max_{norm}=0.5 is applied before each parameter update to prevent gradient

explosion, which is particularly relevant for the deep stacked attention layers where gradient norms can spike during early training when the attention distributions are still near-uniform.

At inference time, the model applies Test-Time Augmentation (TTA) over five stochastic forward passes with dropout active (model.train() mode), averaging the resulting logits. This approach, related to MC Dropout [37], leverages the dropout and stochastic depth layers as approximate Bayesian inference, producing a more robust prediction by marginalising over the dropout distribution. For the three regression targets, the same five-pass average is applied to reg_{pred}. TTA is computationally affordable at inference time: five passes over a sequence of at most 110 timesteps add negligible latency, and the technique empirically reduces both classification error and regression MAE on held-out data.

3. Dataset

This chapter describes the synthetic benchmark constructed for this thesis. Section 3.1 provides an overview of the dataset and its generation procedure. Section 3.2 specifies the Omega-Prime data model and the MCAP file architecture used for storage. Section 3.3 details the CARLA-to-Omega-Prime conversion pipeline that transforms raw simulator logs into standardised ground-truth scenario files. Section 3.4 discusses known dataset limitations and compliance constraints relative to the Omega-Prime specification. Section 3.5 presents the descriptive statistics of the resulting benchmark.

3.1 Dataset Overview

The dataset used in this thesis consists of simulated highway driving scenarios recorded using the CARLA open-source autonomous driving simulator[38]. A total of 73,829 individual recordings were generated, yielding 216,715 output frames at a fixed sampling frequency of 10 Hz. The scenarios are distributed across three distinct CARLA maps and cover nine highway manoeuvre classes, collectively forming the scenario taxonomy of this work.

Each recording captures the behaviour of a designated ego vehicle and a variable number of surrounding traffic participants across a short temporal window. The nine manoeuvre classes represent the fundamental interaction patterns characterising highway driving:

- **Braking Leading Vehicle (BF):** the vehicle immediately ahead of the ego decelerates abruptly, requiring a reactive response from the ego.
- **Leading Vehicle Cut-In (CINF):** an adversary vehicle from an adjacent lane merge into the ego's lane, reducing the available longitudinal gap.
- **Leading Vehicle Cut-Out (COUTF):** the vehicle currently occupying the lean ahead the ego departs into an adjacent lane.
- **Ego Vehicle Lane Change (LCF)**: the ego vehicle itself executes a lateral lane change manoeuvre.

- **Free Ride (FRF)**: the ego vehicle travels freely on the highway without interacting with a leading vehicle.
- **Ego Pulling Away from Leading Vehicle (FLF_B1)**: the ego vehicle accelerates away from a previously close leading vehicle, increasing the inter-vehicle gap.
- **Approaching Leading Vehicle (FLF_B2)**: the ego vehicle decelerates towards a slower leading vehicle, progressively closing the inter-vehicle gap.
- **Following Leading Vehicle (FLF_B3)**: the ego vehicle maintains a stable following distance behind a leading vehicle.
- **Following Distant Leading Vehicle (FLF_B4)**: the ego vehicle follows a leading vehicle at a larger, more comfortable distance.

The raw CARLA recorder logs produced during the simulation campaign were converted into the Omega-Prime format through a dedicated pipeline describe in Section 3.3. The resulting data format and its internal structure are introduced in Section 3.2.

3.2 Omega prime format & data architecture

The dataset is stored in the **Omega-Prime** format, an open data model and file format for ground truth road traffic data developed by the Institute for Automotive Engineering at RWTH Aachen University, within the EU-funded SYNERGIES project [39]. Omega-Prime is the successor of OMEGA Format and is grounded in three established open standards: MCAP, ASAM OSI, and ASAM OpenDRIVE. [39]

MCAP is an open-source, serialisation-agnostic container file format designed for robotics and autonomous driving application [40]. It stores multiple channels of timestamped heterogeneous data with embedded schemas, ensuring long-term readability and forward compatibility. Its row-oriented, append-only design supports fast writes, while optional LZ4 or Zstandard compression further reduces storage footprint. Each Omega-Prime file is an MCAP archive exposing two primary topics:

- ***ground_truth*:** which carries a stream ASAM OSI *GroundTruth* message at 10HZ;
- ***ground_truth_map*:** which stores the static road network description.

ASAM OSI (Open Simulation Interface) is a standard published by ASAM e.V. that defines a Protocol Buffers-based interface for the exchange of simulation environment data[41]. It characterises the complete simulated environment through a GroundTruth message encompassing dynamic actors, traffic infrastructure and environmental state. The Omega-Prime format mandates OSI version 3.7.0 and enforces additional quality constraints on field completeness and accuracy.

ASAM OpenDRIVE is an XML-based standard for the description of static road networks[42]. It encodes road geometry, lane topology, signals and junction logic. In Omega-Prime, the .xodr map file is embedded directly into the MCAP archive under the /ground_truth_map topic, yielding fully self-contained recordings.

All spatial data adheres to the ISO 8855:2011 right-handed coordinate system, with the X axis pointing forward, Y pointing left and Z pointing upward. Each GroundTruth message carries the following structured information.

At Root level the message includes: country_code (set to 0 to signal the absence of real-world geographic reference, as per ISO 3166-1 numeric); version (OSI 3.7.0); proj_frame_offset (position (0,0,0), yaw 0.0 , no projection offset for simulated data); timestamp (seconds and nanoseconds, derived from the CARLA simulation clock via Packet 0); host_vehicle_id (ego actor identifier); and map_reference (name of the corresponding OpenDRIVE map file).

For each moving object (one entry per dynamic actor per frame), the message provides: actor id; bounding box dimensions (length, width, height, in metres); world-frame position (x, y, z); orientation (roll, pitch, yaw, in radians, bounded to [-π, π]); linear velocity and acceleration vectors in the world frame; OSI type enum (derived from CARLA blueprint, e.g. VEHICLE); vehicle_classification.type (CAR, HEAVY_TRUCK, DELIVERY_VAN, BUS, MOTORBIKE); and

vehicle_classification.role (UNKNOWN, CIVIL, POLICE, AMBULANCE, FIRE, PUBLIC_TRANSPORT), resolved via a per-blueprint lookup table.

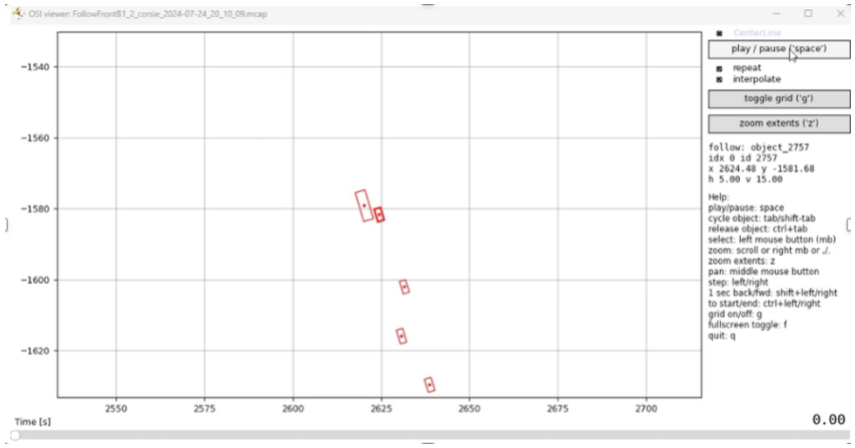

Figure 3. 1 OSI Viewer visualisation of a recorded scenario window. Each rectangle represents a moving object (vehicle) at a given timestep; axes are in metres in the OSI right-handed coordinate frame.

3.3 CARLA to Omega-Prime Conversion Pipeline

The conversion from raw CARLA recorder logs to Omega-Prime MCAP files is performed by dedicated pipeline. The pipeline accepts two inputs, a CARLA .log binary recorder file and the corresponding. xodr map, and proceeds through seven sequential stages.

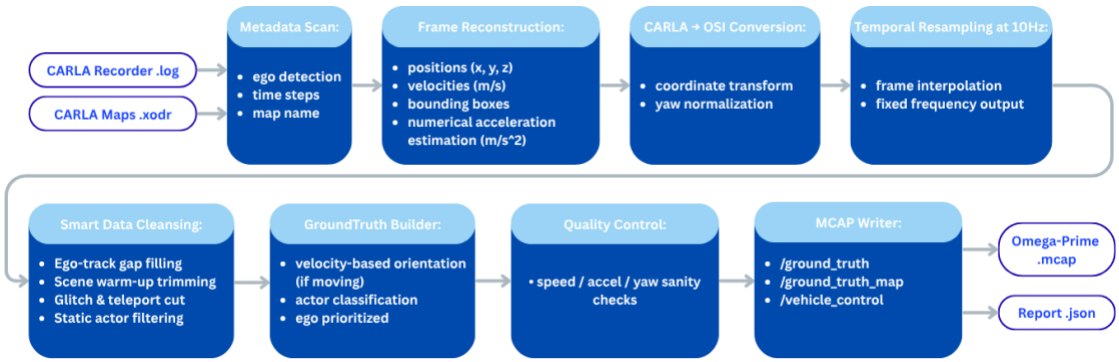

Figure 3. 2 CARLA to Omega-Prime conversion pipeline overview. Raw CARLA binary recorder logs are processed through five sequential stages , metadata scan, frame reconstruction, coordinate conversion, temporal resampling, and MCAP serialisation , to produce standardised OSI-encoded scenario windows.

In Metadata Scan, the pipeline opens the binary log and performs a lightweight single-pass scan to extract metadata without constructing full frame objects. It reads the map name from the file header, actor metadata from EVENT_ADD packets (including blueprint identifiers and, from CARLA 0.9.14, base_type and special_type attributes), per-actor frame counts used to measure actor persistence, frame timestamps from FRAME_START packets, and spawn positions of traffic light actors. The estimated recording frequency is computed from the median inter-frame interval of the timestamp sequence.

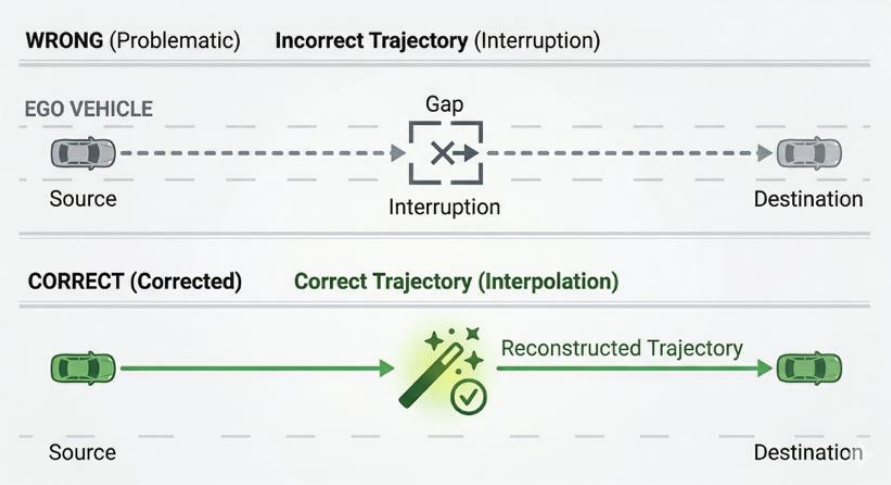

Figure 3. 3 Trajectory gap filling via linear interpolation. When a recording gap is detected, the pipeline reconstructs the missing kinematic states by linear interpolation between the last valid and first valid frames (bottom).

In Frame Reconstruction, the full-pass second iteration reconstructs the complete kinematic state of every actor per frame. Positions are parsed from Packet 6 and converted from centimetres to metres. Linear velocities are sourced from Packet 12 (kinematics), already expressed in m/s by the CARLA recorder. Bounding box extents are read from Packet 13 and converted from half-extents in centimetres to full dimensions in metres; when bounding box data is absent, default dimensions of 4.0 x 1.8 x 1.5 m are applied. Acceleration vectors are not stored natively in the CARLA recorder format and are therefore estimated numerically from successive velocity differences.

Since CARLA uses the Unreal Engine left-handed coordinate system, whereas OSI adopts the ISO 8855 right-handed convention. In CARLA to OSI Coordinate Conversion, the coordinate transform negates the Y axis for all position and velocity vectors and inverts the sign of yaw and pitch angles accordingly. Angular velocity components are converted from degrees/s to rad/s with the Y component negated. A noteworthy challenge arose from the ordering of Euler angles in the binary log: the CARLA recorder stores rotations via FQuat::Euler(), which returns angles in (roll, pitch, yaw) order as defined by Unreal Engine. The official CARLA documentation incorrectly depicts this ordering as (pitch, yaw, roll). The pipeline follows the C++ source code rather than the documentation, ensuring correct angle extraction , a discrepancy that would otherwise produce erroneous orientation values for all actors.

CARLA simulations do not guarantee a strictly fixed frame rate; the inter-frame interval may vary due to simulator load. The Temporal Resampling detects missing or irregular frames by comparing recorded timestamps against the nominal interval and fills gaps through linear interpolation of positions and velocities, and spherical linear interpolation (SLERP) for orientation angles. The output is resampled to a uniform 10 Hz grid, producing fixed-length, evenly spaced time series suitable for sequence modelling.

At the and for each reconstructed frame is assembled into an OSI GroundTruth message. The ego vehicle is identified using a tiered heuristic: actors carrying role_name == "hero" or "ego" take priority; if absent, an explicitly supplied actor ID is used; otherwise, the vehicle appearing in the greatest number of frames is selected as a fallback. Actor types are mapped from CARLA blueprint identifiers to OSI MovingObjectType and VehicleClassificationType enums. Vehicle roles (police, ambulance, fire, public transport) are resolved via a dedicated lookup table keyed on blueprint ID. Traffic light actors are placed in the traffic_light list rather than moving_object, as mandated by the Omega-Prime specification. The ego vehicle is written first in each moving object list.

Before serialisation, each frame is validated against plausibility thresholds in the Quality control block: speeds exceeding 100 m/s, accelerations exceeding 50 m/s², and yaw rates exceeding 10 rad/s are flagged as anomalous. Orientation values are guaranteed to lie within [-π, π] by construction. A QC statistics accumulator collects per-run counts of anomalies and is reported at the end of each conversion job.

MCAP Writer: Validated OSI GroundTruth messages are serialised using Protocol Buffers and written to an MCAP file under the /ground_truth topic at 10 Hz. The embedded map is written once per file to /ground_truth_map. The output file is a self-contained, standards-compliant Omega-Prime recording.

3.4 Dataset Criticalities and Compliance Notes

Several aspects of the dataset require explicit discussion in relation to the Omega-Prime specification requirements.

The Omega-Prime specification requires each traffic light entry to include a source_reference field that links the actor to its corresponding OpenDRIVE signal identifier. This information is not available in the CARLA binary recorder: Packet 7 stores only the actor's DatabaseId, elapsed_time, is_frozen, and colour state. No cross-reference to the OpenDRIVE signal graph is exposed at the recorder level. Therefore, traffic_light.source_reference and traffic_light.classification.icon are absent from all dataset files. This is a structural limitation of the CARLA recorder format rather than of the pipeline.

To explicitly indicate the absence of real-world geographic grounding, every message carries country_code = 0 , the ISO 3166-1 numeric code reserved to signal no country affiliation, as opposed to, for example, 276 (Germany) or 840 (United States) , and proj_frame_offset = position(0,0,0), yaw=0.0, signalling that the simulation coordinate origin has no alignment to any real-world geographic projection.

Range accuracy verification. The Omega-Prime specification defines minimum accuracy thresholds: position 0.2 m, orientation 0.035 rad, velocity 0.1 m/s, and

acceleration 0.1 m/s². For synthetic data, these thresholds are verified through internal kinematic consistency rather than against independent sensor measurements. The maximum deviation between recorded velocity vectors and displacement-derived speed across the full dataset is 0.053 m/s, below the 0.1 m/s threshold. Orientation values are bounded within [-π, π] by construction. Roll and pitch, extracted using the correct FQuat::Euler() field order, are consistently near 0° on flat road surfaces , well within the 0.035 rad threshold.

3.5 Descriptive Statistics

The dataset comprises 73,829 MCAP recordings totalling 216,715 frames sampled at 10 Hz, distributed across three CARLA simulation maps and nine scenario classes. The class distribution is shown in Figures 3.4**.**

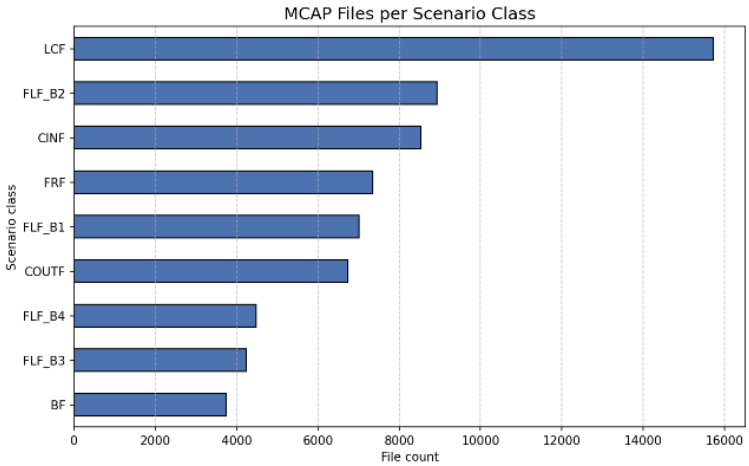

Figure 3. 4 Class recording distribution across the nine ISO 34502 scenario classes.

The distribution is markedly imbalanced: the LCF class (ego lane change) dominates with approximately 15,800 recordings, four times the count of the least represented class, BF (braking leading vehicle) at approximately 3,800 recordings. The three most represented classes (LCF, FLF_B2, CINF) account for approximately 44% of all files. This imbalance reflects the relative frequency of these manoeuvres in the simulation

campaign design and represents a non-trivial challenge for classifier training, addressed in Chapter 4.

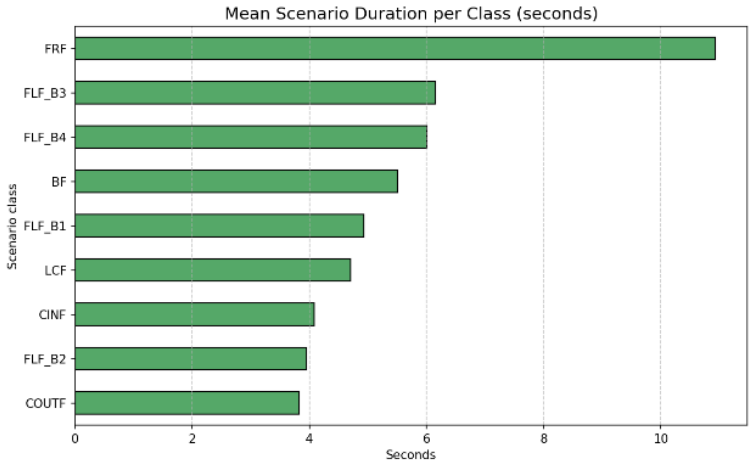

Figure 3. 5 Mean scenario duration per class in seconds (right) across the nine ISO 34502 scenario classes.

In Figure 3.5 shows the mean scenario duration per class: Free Ride (FRF) exhibits by far the longest mean duration, at approximately 11 seconds, reflecting the absence of a discrete triggering event and the naturally sustained character of unobstructed highway travel. The following-type classes (FLF_B3, FLF_B4) show mean durations of approximately 6 seconds, consistent with their steady-state interaction dynamics. Event-driven classes, COUTF and FLF_B2, are the shortest at approximately 3.8 and 4.0 seconds respectively, consistent with the transient and well-delimited nature of cut-out and approach manoeuvres. The LCF class has a mean duration of 4.8 seconds. The combination of class imbalance in recording count and variability in temporal length has direct implications for input sequence design and loss weighting in the proposed model, as discussed in Chapter 4.

4. System Architecture

This chapter presents the complete modelling system developed in this thesis. Section 4.1 describes the preprocessing workflow that converts Omega-Prime MCAP recordings into fixed-size tensors ready for model training, covering feature extraction, sharded storage, class-imbalance handling, and the end-to-end pipeline summary. Section 4.2 details the architecture of the Spatio-Temporal Transformer (STT), from input projection through spatial encoding, Ego Cross-Attention, temporal encoding, and temporal attention pooling to the multi-task output heads. Section 4.3 documents the training configuration, including the multi-task loss, data augmentation strategy, optimiser settings, and evaluation protocol.

4.1 Workflow

4.1.1 From Omega-Prime Recordings to Model-Ready Tensors

The architecture described in this chapter consumes the Omega-prime dataset introduced in Chapter 3 through an intermediate preprocessing stage that converts variable-length MCAP recording into fixed-shape numerical tensors suitable for batched GPU training. Each scenario recording, originally a sequence of OSI GroundTruth messages sampled at 10Hz, is transformed into a tensor X of shape [T, K, F], where T is the number of timesteps in the resampled recording, K is the maximum number of tracked vehicles per timestep, and F is the number of per-vehicle kinematic and categorical features extracted from each moving_object entry (position, velocity, acceleration, orientation, bounding box, and classification fields). A Boolean validity mask M of shape [T, K] accompanies every tensor, marking which vehicle slots are populated at each timestep; slots corresponding to vehicles not present, not yet spawned, or already despawned are masked out rather than zero-padded silently, ensuring that the attention mechanisms described in Section 4.2 never attend to fictitious agents.

A structural convention enforced throughout the pipeline, inherited directly from the GroundTruth Builder stage of Chapter 3 (Section 3.3, Stage 5), is that the ego vehicle

always occupies slot index 0 in the K dimension. This convention removes the need for the model to search for the ego vehicle at training or inference time and allows the Ego Cross-Attention layer (Section 4.2.3) to address it by a fixed index.

4.1.2 Class Imbalance Handling at the Sampling Level

As documented in Section 3.5, the class distribution is imbalanced by a factor of approximately four between the most frequent class (LCF) and the least frequent (BF). Before any loss-level correction is applied (Section 4.3.1), the data loading workflow itself performs class balancing through a WeightedRandomSampler: each training sample is assigned a sampling weight inversely proportional to the frequency of its class, computed once per epoch from the per-shard label counts. This produces mini-batches whose class composition is approximately uniform regardless of the underlying corpus imbalance, complementing rather than substituting the loss-level reweighting strategy described later in this chapter.

4.1.3 End-to-End Pipeline Summary

Figure 4.1 summarises the complete workflow, from the CARLA recorder logs described in Chapter 3 through the Omega-Prime conversion pipeline, the sharded NPZ preprocessing stage described above, and finally the spatio-temporal transformer model and its multi-task training loop described in Sections 4.2 and 4.3. The clear separation between the data-engineering stage (Section 3 and & 4.1) and the modelling stage (Sections 4.2 & 4.3) allows the two to be developed, validated, and iterated upon independently.

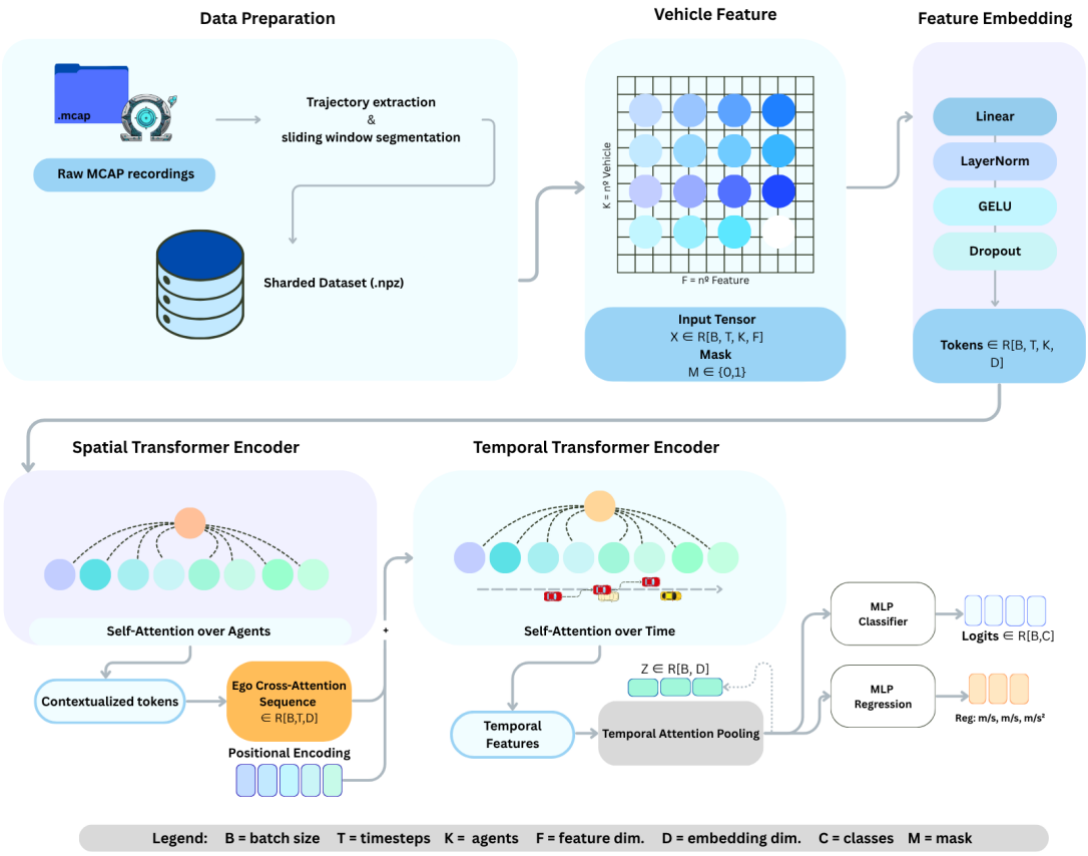

Figure 4. 1 End-to-end training and inference pipeline, from raw CARLA recorder logs to scenario class predictions.

4.2 Spatio-Temporal Transformer Architecture

4.2.1 Spatio-Temporal Design

The model, named SpatialTemporalTransformerV2MTL, implements the factorised spatio-temporal attention paradigm motivated in Section 2.3: rather than computing a single joint attention operation over the full (T × K) set of vehicle-timestep tokens, the architecture alternates a spatial encoding stage (attention across vehicles within a single timestep) with a temporal encoding stage (attention across timesteps of a single, ego-centric representation).This factorisation reduces the computational cost from quadratic in (T·K) to the sum of a term quadratic in K and a term quadratic in T, and reflects the structural assumption, justified in Section 2.3, that spatial interactions (which vehicle affects the ego right now) and temporal dynamics (how the ego-centric situation evolves) are governed by largely separable mechanisms.

4.2.2 Input Projection and Vehicle Embedding

Each per-vehicle, per-timestep feature vector of dimension F is first projected to the model's working dimension d_{model} = 256 through a linear layer followed by layer normalisation, a GELU non-linearity, and dropout (rate 0.15). A learned vehicle embedding, implemented as an nn.Embedding indexed by the vehicle's slot position (0 to K−1, with K = max_vehicles = 64), is added to the projected features.

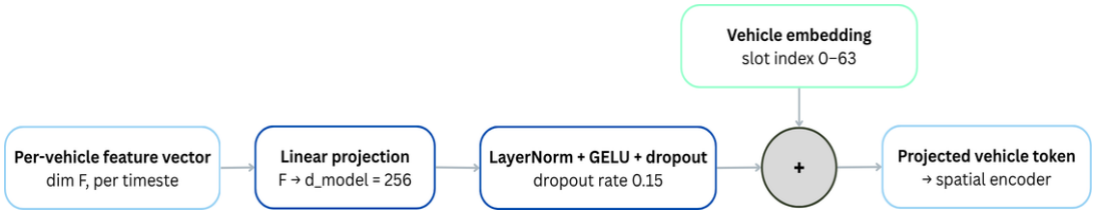

Figure 4. 2 Input projection and vehicle embedding. Each per-vehicle, per-timestep feature vector of dimension F is mapped to the model working dimension d_model via a linear projection layer, yielding a K×T tensor of embedded tokens.

This embedding supplies the model with a fixed positional reference within the vehicle-slot dimension, distinguishing the ego slot (always index 0) from the

remaining agent slots, without imposing any further constraint on which agent occupies which non-ego slot, a deliberate design choice that preserves permutation invariance with respect to the ordering of non-ego vehicles while still letting the model learn that slot 0 carries a privileged role.

4.2.3 Spatial Encoder

The spatial encoder consists of three stacked standard Transformer encoder layers (nn.TransformerEncoderLayer), each with 8 attention heads, a feed-forward dimension of

1024, dropout 0.15, pre-layer normalisation (norm_first=True, motivated in Section 2.2 by the training-stability findings of Xiong et al. [43]), and GELU activation.

Figure 4.3 shows, at each timestep independently, the B·T instances of the K-vehicle set are processed as a batch of token sequences of length K, with a key-padding mask derived from the validity mask (M) ensuring that absent vehicle slots are excluded from the attention computation and do not contaminate the representations of valid slots. Stochastic Depth (Section 2.3, Huang et al., 2016 [26]) is applied to the residual branch of each spatial layer, with a drop probability increasing linearly across the three layers (from drop_path/3 to drop_path, with drop_path = 0.15), regularising the deeper layers more aggressively than the shallower ones.

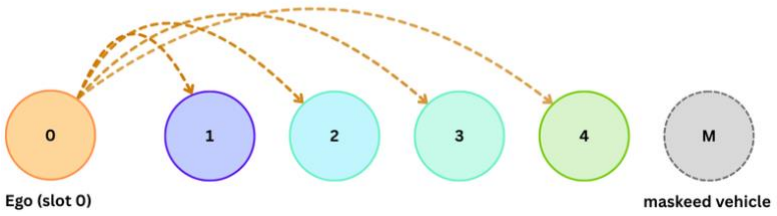

Figure 4. 3 Spatial encoder. Three stacked Transformer encoder layers process all K vehicle tokens jointly at each timestep independently, producing a spatially-attended representation for each vehicle slot that captures pairwise inter-agent relationships.

4.2.4 Ego Cross-Attention

Following the spatial encoder, an Ego Cross-Attention module (Section 2.3) extracts, for each timestep, a single ego-centric representation from the full set of K vehicle tokens.

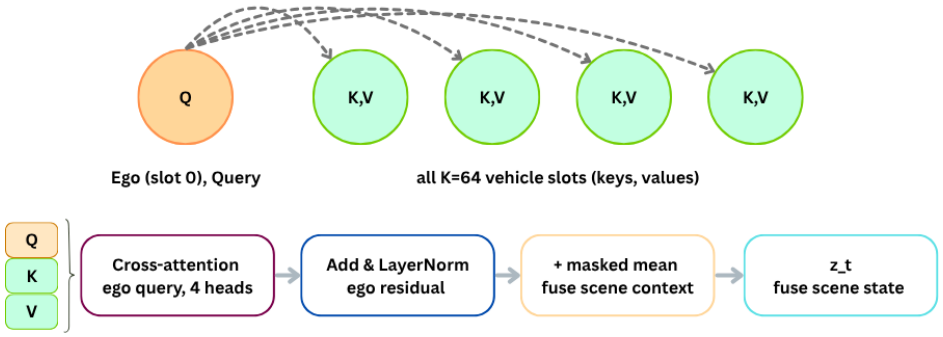

Figure 4. 4 Ego Cross-Attention module. The ego vehicle token (slot 0) acts as the query, while all K spatial tokens serve as keys and values, aggregating interaction-aware context from the full agent set into a single ego-centric vector for each timestep.

The ego vehicle's token (slot 0) is used as the query, while the entire K-token set serves as keys and values, in a standard nn.MultiheadAttention layer with 4 heads (half of nhead_spatial) and dropout 0.15. The attention output is added to the original ego token through a residual connection followed by layer normalisation and is further combined with the masked mean of all K vehicle representations at that timestep, fusing the ego-query perspective with a coarse global summary of the scene. A numerical safeguard handles the degenerate case in which every vehicle slot at a given timestep is invalid (no tracked vehicle at all, which can occur transiently at scenario boundaries): the implementation temporarily unmasks the ego slot to avoid feeding an all-True key-padding mask into nn.MultiheadAttention which would otherwise produce NaN outputs and explicitly zeroes the resulting representation for any such degenerate timestep before it propagates further.

4.2.5 Temporal Encoder

The sequence of T ego-centric vectors produced by the Ego Cross-Attention module is augmented with sinusoidal positional encoding (Section 2.2, [12]) and processed by a second stack of four Transformer encoder layers , again with 8 attention heads, feed-forward dimension 1024, pre-layer normalization, and linearly increasing Stochastic Depth (this time scaled across four layers). A timestep-level validity mask, derived as the logical OR over the vehicle-level mask at each timestep, identifies timesteps with at least one valid vehicle; the same safeguard described above (unmasking a single position to avoid NaNs, followed by explicit zeroing) is applied for recordings whose every timestep is invalid.

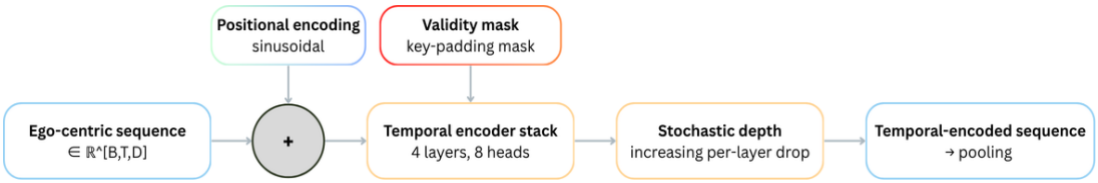

Figure 4. 5 Temporal encoder. The T-length sequence of ego-centric context vectors, augmented with sinusoidal positional encoding, is processed by stacked temporal Transformer encoder layers to model the evolution of the scenario over time.

4.2.6 Temporal Attention Pooling

The output of the temporal encoder, a per-timestep representation of dimension d_{model} is aggregated into a single fixed-size vector through a Temporal Attention Pooling layer (Section 2.2): a two-layer MLP computes a scalar relevance score per timestep, which is converted to a normalised attention distribution via a masked SoftMax, guarding against the degenerate case of an entirely invalid sequence by substituting a uniform mask before applying the SoftMax and used to compute a weighted sum of the per-timestep representations. This produces the pooled representation $z_{pool} \in R^{d_{model}}$ that feeds both task-specific heads described next.

Figure 4. 6 Temporal Attention Pooling. A learned query vector attends over the T per-timestep representations produced by the temporal encoder, yielding a single fixed-size context vector that summarises the full scenario window for downstream classification and regression.

4.2.7 Multi-Task Heads

Two heads operate on the shared z_{pool} representation, embodying the multi-task learning strategy motivated in Section 2.4. The classification head is a four-layer MLP with GELU activations, two LayerNorm operations, and dropout rates of 0.225 and 0.15 respectively, producing the 9-way scenario logits. The regression head is a lighter two-layer MLP producing the three normalised regression outputs described in Section 4.1.3.

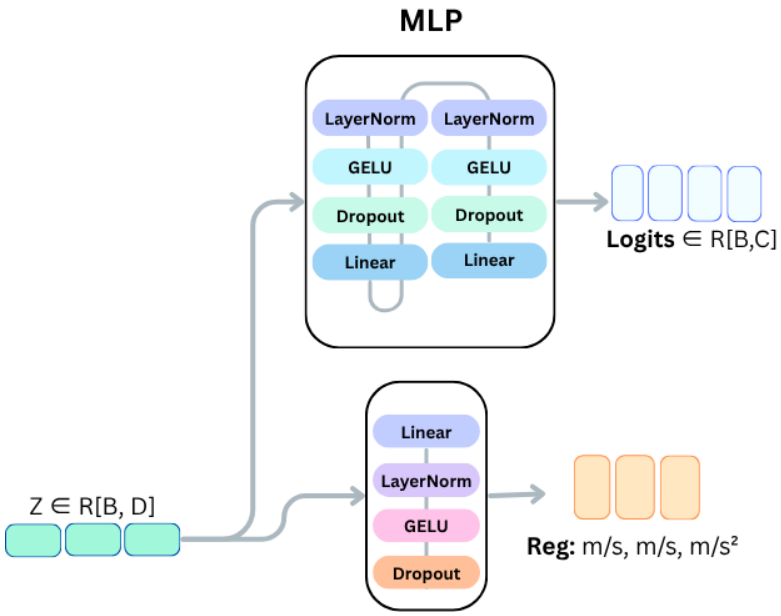

Figure 4. 7 Multi-task output heads. The shared context vector feeds a nine-class classification head (trained with Focal Loss) and a three-target regression head (trained with Huber Loss). Task weights are learned via Kendall homoscedastic uncertainty weighting.

Both heads are trained jointly with the shared encoder; no gradient stopping is applied between the two branches, allowing the regression objective to shape the encoder's internal representation as intended by the multi-task design.

4.2.8 Parameter Budget

With the configuration described above, the model totals approximately 9.5 million trainable parameters, a size chosen to remain trainable on a single consumer GPU within the time constraints of the experimental campaign while retaining sufficient capacity for the 9-way classification and 3-target regression tasks over sequences of up to 110 timesteps and 64 vehicle slots.

4.3 Training Setup & Optimisation

4.3.1 Loss Function and Multi-Task Weighting

The classification branch is trained with Focal Loss (Section 2.4, [33]) with focusing parameter γ = 1.5 and label smoothing 0.08, combined with per-class weights derived from the inverse class frequency in the TRAIN split (clipped to the range [0.3, 3.0] to avoid extreme weighting of the rarest class). The regression branch is trained with the Huber loss (delta = 1.0), chosen for its robustness to outliers relative to a squared-error loss, which is desirable given that the regression targets are themselves derived from peak (max/min) statistics that can be sensitive to occasional sensor or interpolation artefacts in the underlying kinematic channels.

The two task losses are combined using the homoscedastic uncertainty weighting method of Kendall et. Al (Section 2.4), enabled in the model configuration. Two learnable scalar parameters, log (σ_{cls}^2) and log (σ_{reg}^2), parametrise the precision of each task. The combined loss is

$$L_{total} = e^{-s_{cls}} \cdot L_{cls} + e^{-s_{reg}} \cdot L_{reg} + \frac{1}{2}(s_{cls} + s_{reg})$$

A fixed-weight alternative

$$L_{total} = L_{cls} + \lambda_{reg*L_{reg}} \; with \; \lambda_{reg} = 0.3$$

is also implemented and is used as a fallback should the learned weighting prove unstable, but the uncertainty-weighted formulation is the one adopted for the reported configuration, removing the need to hand-tune λ_{reg}.

4.3.2 Data Augmentation

MixUp (Section 2.4, [34]) is applied at the batch level with α = 0.3. For every training batch, a mixing coefficient λ is drawn from Beta (0.3, 0.3) and a random permutation of the batch is mixed with the original: inputs X and validity masks M are combined as a convex combination and a logical OR respectively, while the classification loss is computed as the λ-weighted combination of the loss against the original and permuted labels. As discussed in Section 2.4, the three regression targets are not linearly interpolated; instead, the target of the dominant sample (the one associated with $\lambda \geq 0.5$) is retained in full, preserving the physical validity of peak-statistic targets that do not compose linearly under interpolation.

4.3.3 Optimiser and Learning Rate Schedule

Parameters are optimised with AdamW (Section 2.4, [35]), with weight decay 2×10^{-2}. The learning rate follows a One-Cycle schedule (Section 2.4, [36]) with peak learning rate 3×10^{-4}, warm-up fraction pct_start = 0.3, initial learning rate lr_max/10, and final learning rate lr_max/1000, annealed with a cosine curve over the full training run. Gradient norms are clipped to a maximum of 0.5 before every optimiser step, mitigating the occasional gradient spikes that can occur in early training when the attention distributions are still close to uniform across the spatial and temporal encoders.

4.3.4 Regularisation Strategies

In addition to the architectural regularisation already described in Section 4.2 (Stochastic Depth, dropout, layers normalisation), training relies on the WeightedRandomSampler and on MixUp (Section 4.3.2) to smooth the decision boundary. Early stopping is applied on the validation macro-F1 score with a patience of 15 epochs out of a maximum budget of 80 and the checkpoint corresponding to the best validation macro-F1 is retained for final evaluation.

4.3.5 Evaluations Protocol and Test-Time Augmentation

Validation and test evaluation report classification accuracy, macro and weighted F1, and mean absolute error (MAE) for each of the three regression targets, the latter de-normalised back to physical units using the reg_{mean} and reg_{std} statistics described in Section 4.1.3. At test time, predictions are additionally computed with Test-Time Augmentation (Section 2.4, related to MC Dropout Interpretation of Gal & Ghahramani,[37]): five stochastic forward passes are performed with dropout kept active, and the resulting logits and regression output are averaged. This is reported alongside the single-pass (deterministic) evaluation to quantify the contribution of TTA to the final reported metrics. All experiments are logged to Weights & Biases, recording per-epoch training and validation losses, classification metrics, regression MAE, the learned task-precision values when uncertainty weighting is active, and the final confusion matrix on the test set.

5. Experimental Results

All architectures are evaluated on the same held-out test partition of 31,502 scenario windows, never exposed during training or hyperparameter selection. Windows are extracted at 10 Hz with a fixed length of up to 110 timesteps and normalised per-feature using statistics computed exclusively on the training split.

All deep models are trained with three independent random seeds to account for training stochasticity. Reported aggregate metrics are mean ± standard deviation across seeds. Three metrics are used throughout: test-set accuracy, macro-averaged F1 ($F1_m$), and weighted-average F1 ($F1^W$). Macro-F1 weights all nine classes equally and is therefore more sensitive to performance on rare or difficult classes; weighted-F1 accounts for class support, giving a more representative measure of overall recognition quality on the test distribution.

5.1 Quantitative result

Table 5.1 summarises aggregate test-set performance for all models. Two distinct performance tiers are immediately apparent.

Model	Accuracy	Macro-F1	Weighted-F1
STT (proposed)	0.783 ± 0.008	0.732 ± 0.005	0.785 ± 0.009
Flat Transformer	0.786 ± 0.017	0.733 ± 0.020	0.788 ± 0.019
STGCN	0.779 ± 0.007	0.730 ± 0.007	0.782 ± 0.008
LSTM-ego	0.602 ± 0.001	0.444 ± 0.001	0.597 ± 0.001
SVM (ego)	0.555	0.381	0.524
SVM (ego+k1)	0.597	0.462	0.588

Table 5. 1 Test-set performance comparison across all models. Deep model results are mean ± std across seeds; SVM results are deterministic single-run values.

The F1 macro result are plot in Figure 5.1 and it's followed by the confusion matrix reported in Figure 5.2.

The three interaction-aware deep models , STT, Flat Transformer, and STGCN , form a compact upper tier with macro-F1 values of 0.732±0.005, 0.733±0.020, and 0.730±0.007 respectively, and test-set accuracies between 77.9% and 78.6%. The differences among these three models lie within one standard deviation across seeds and are not statistically significant given the number of seeds used.

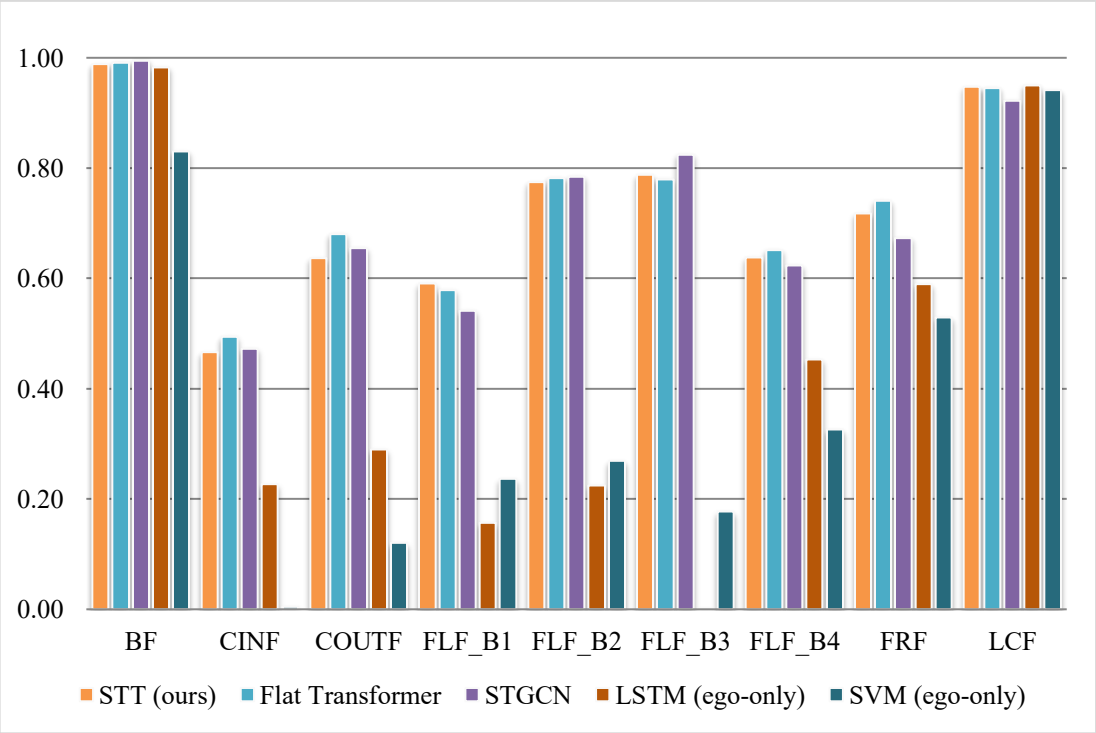

Figure 5. 1 Comparison per Class of F1 weight score

The LSTM-ego forms the second tier with a macro-F1 of 0.444±0.001 a gap of approximately 0.29 points below the interaction-aware models. This drop is entirely consistent across all metrics and reproducible across seeds (variance < 0.001), providing clear evidence that multi-agent context is the dominant driver of scenario recognition performance.

The SVM baselines occupy a third tier. SVM(ego) reaches a macro-F1 of 0.381, and SVM(ego+k1) raises this to 0.462 by including the nearest-neighbour vehicle features.

Even with neighbour information, both SVM variants remain well below the deep interaction-aware models (gap of at least 0.27 points in macro-F1).

Table 5.2 shows the per-class F1 score for each model, averaged across seeds where applicable. The breakdown reveals substantial heterogeneity across the nine scenario classes.

Class	STT	Flat	STGCN	LSTM-ego	SVM(ego)	SVM(ego+k1)
BF	0.989	0.988	**0.995**	0.982	0.830	0.877
CINF	**0.487**	0.478	0.481	0.213	0.004	0.386
COUTF	0.629	0.646	**0.655**	0.293	0.121	0.101
FLF_B1	**0.569**	0.564	0.548	0.193	0.236	0.264
FLF_B2	**0.795**	0.787	0.783	0.254	0.269	0.363
FLF_B3	0.811	0.789	**0.825**	0.000	0.177	0.199
FLF_B4	0.623	0.623	**0.632**	0.473	0.326	0.376
FRF	0.728	**0.763**	0.720	0.626	0.529	0.648
LCF	0.953	0.958	0.932	**0.959**	0.941	0.945

Table 5. 2 Per-class F1 scores (averaged across seeds). Bold values mark the best score per class.

The nine classes fall into three groups according to difficulty. BF (Braking Leading Vehicle) and LCF (Lane Change of Ego) are reliably recognised by all models, including the SVM and the ego-only LSTM. BF achieves F1 scores of 0.830 (SVM ego), 0.982 (LSTM-ego), and 0.989 (STT); LCF reaches 0.941, 0.959, and 0.953 respectively. Both classes are largely definable from ego-centric signals: braking manifests as ego deceleration and headway reduction, while a lane change is captured in lateral velocity and heading error. This explains why even the most limited baselines handle them reliably.

CINF (Leading Vehicle Cut-In) and FLF_B3 (Following at Close Range) are the clearest examples of interaction-dependent classes. For CINF, SVM(ego) achieves an F1 of only 0.004 because the defining event, a neighbouring vehicle merging into the ego lane, is invisible from ego signals at its onset. Adding the nearest-neighbour vehicle to the SVM raises CINF F1 to 0.386, confirming that even a single neighbour

provides discriminative information. The LSTM-ego reaches 0.213 on CINF, still far below the interaction-aware deep models which achieve approximately 0.481–0.487. FLF_B3 is the most striking case: the LSTM-ego assigns exactly F1 = 0.000 across both seeds, the model cannot learn to distinguish this class from other following scenarios because the close-range criterion is defined entirely by the relative distance to the leading vehicle, a signal the LSTM never receives. The interaction-aware models achieve F1 between 0.789 and 0.825 on FLF_B3.

The remaining classes occupy an intermediate difficulty range. FLF_B2 (Approaching Leading Vehicle) is the best-recognised of this group for the deep models (F1 ≈ 0.783–0.795), while FLF_B1 (Ego Pulling Away) is the weakest (F1 ≈ 0.548–0.569). COUTF (Leading Vehicle Cut-Out) exhibits a notable precision-recall asymmetry: precision is high across deep models, but recall is markedly lower, suggesting the models correctly identify COUTF when they commit to the prediction, but miss many instances, most likely classifying them as the preceding steady-state scenario.

5.2 Ablation studies

Two complementary ablations quantify the contribution of multi-agent context, the key design choice shared by the STT, Flat Transformer, and STGCN.

Effect of neighbour information in the SVM baseline. The transition from SVM(ego) to SVM(ego+k1) constitutes a controlled ablation of the neighbour information contribution within the classical feature-engineering framework. Adding the nearest-neighbour vehicle features raises the overall macro-F1 by 0.081 (from 0.381 to 0.462) and accuracy by 0.042. The gain is highly class-specific: CINF improves by +0.382 (from near-zero to 0.386), FRF improves by +0.119, and FLF_B2 by +0.094. By contrast, COUTF slightly worsens (−0.020), which may reflect interference between ego-vehicle features and the kinematics of the departing vehicle.

This ablation confirms two things: (i) multi-agent context provides genuine discriminative signal even in the simplest linear model; and (ii) the information from

a single nearest neighbour is sufficient to dramatically improve some classes while being neutral or mildly detrimental for others. The fact that COUTF worsens suggests that the SVM cannot reliably generalise the cut-out cue from only the ego-neighbour pair, whereas the full attention-based models, which attend to all slots, can resolve the ambiguity.

Effect of multi-agent interaction in the deep models. Comparing the LSTM-ego against the interaction-aware deep models provides an implicit ablation of the spatial attention mechanism. The LSTM-ego uses the identical temporal encoder structure, training configuration, loss function, and classification head as the STT, but operates exclusively on the ego vehicle's feature sequence. The macro-F1 gap of 0.288 points (0.444 vs. 0.732) therefore isolates the contribution of multi-agent context modelling from all other architectural choices.

The completeness of this gap is underscored by the FLF_B3 result: an entire class that the LSTM-ego cannot learn at all (F1 = 0.000) becomes readily identifiable once surrounding vehicles are visible (F1 ≈ 0.811 for STT). This is arguably the most interpretable single finding of the experimental evaluation: multi-agent context is not merely helpful; it is necessary for the recognition of certain scenario classes.

5.3 Auxiliary Regression Evaluation

The three deep interaction-aware models STT, Flat Transformer, and STGCN were trained with an auxiliary regression objective over three ego-vehicle kinematic targets: peak lateral velocity, mean speed, and peak deceleration. Table 5.3 reports the test-set mean absolute error (MAE) for each target, both in normalised units and in physical units, alongside the LSTM-ego for comparison.

Model	Peak lat. vel. [m/s]	Mean speed [m/s]	Peak decel. [m/s²]
STT (proposed)	6.6e–5 ± 0.6e–5	0.310 ± 0.042	0.078 ± 0.015
Flat Transformer	6.2e–5 ± 0.1e–5	0.308 ± 0.039	0.074 ± 0.005
STGCN	5.8e–5	0.278	0.075

LSTM-ego	6.0e–5 ± 0.2e–5	0.340 ± 0.037	0.077 ± 0.001

Table 5. 3 Test-set MAE for the three auxiliary regression targets (physical units). Values are mean ± std across seeds.

The MAE for Peak lat. Velocity is approximately 6×10^{-5} m/s across all models. Eight of the nine scenario classes involve no lateral ego manoeuvre, making the target nearly always zero in the test set. The model learns to predict near-zero for all samples and achieves trivially low MAE as a result. This target did not provide a meaningful discriminative signal at inference time, although the lateral-velocity gradient may still have contributed a mild regularisation effect during training for the LCF class.

For the Mean speed and peak deceleration, two of non-degenerate targets, MAE is 0.278–0.340 m/s for mean speed and 0.074–0.078 m/s² for peak deceleration across all models. Both values represent a small fraction of the typical range of each target, indicating that the regression head learned meaningful predictions. Notably, the LSTM-ego achieves regression MAE that is comparable to or within the variance of the interaction-aware models: 0.340 m/s versus 0.308–0.310 m/s for mean speed, and 0.077 m/s² versus 0.074–0.078 m/s² for peak deceleration.

This near-parity is expected and interpretable: mean speed and peak deceleration are properties of the ego vehicle and are directly observable from its own feature sequence. Multi-agent context provides little additional information for predicting these quantities. The result stands in sharp contrast to the classification task, where the same LSTM-ego falls 0.288 macro-F1 points below the interaction-aware models. The divergence between the two tasks confirms the design rationale of the multi-task training strategy: the regression objective acts as a task-specific regulariser that encourages the shared encoder to represent kinematically meaningful ego features, while the classification objective drives the spatial attention mechanism to exploit multi-agent context. Neither objective interferes with the other.

5.4 Qualitative results

This section discusses the key findings of the experimental evaluation, analysing why the results deviate from initial expectations and what practical conclusions can be drawn.

The original hypothesis motivating the factorised design was that decomposing spatial and temporal attention would improve accuracy by reducing complexity while preserving the capacity to model long-range dependencies. On this dataset, the hypothesis is not supported: the STT and Flat Transformer reach statistically indistinguishable macro-F1 scores.

Several factors may explain this outcome. First, the temporal sequences are relatively short (up to 110 timesteps) and the number of populated agent slots is bounded, so the quadratic cost of flat attention is not prohibitive at this scale. The flat model may be capturing cross-agent-timestep dependencies that the factorised model explicitly decouples. Second, the dataset is synthetic and generated by CARLA with scripted traffic, which may impose strong temporal regularity that makes the factorised approximation sufficient. In natural traffic with more complex and less predictable behaviour patterns, the factorised design may prove more beneficial.

It is worth noting that the Flat Transformer exhibits considerably higher variance across seeds (std = 0.020 on $F1_m$ versus 0.005 for the STT), suggesting that while its average performance is comparable, the factorised design produces more stable training dynamics. In a deployment context where consistency across training runs matters, this stability advantage is meaningful.

The 0.288-point gap in macro-F1 between the ego-only LSTM-ego baseline and the interaction-aware models is not uniformly distributed across classes. Ego-only models perform near-chance on CINF and COUTF, where the defining event (lateral movement of a neighbour) produces no diagnostic signal in the ego kinematic stream until the ego vehicle begins to react, which may occur at the very end of the window or not at all within the fixed observation horizon. Similarly, the FLF class, which

requires representing the relative distance to the leading vehicle, is not recoverable from ego speed alone. By contrast, ego-only models retain reasonable accuracy on BF (characterised by strong ego deceleration) and FF (characterised by constant low speed with no proximity cue required). This class-conditional pattern is consistent with the theoretical prediction: adding multi-agent context should benefit precisely those classes whose label is defined by an inter-agent relationship rather than by an absolute ego-vehicle state.

STGCN requires a pre-specified adjacency structure, which is typically seen as a limitation in ego-relative representations where the spatial topology changes continuously. In this implementation, the graph was constructed with a fixed spatial neighbourhood based on relative proximity at each timestep. The comparable performance with the graph-free Transformer models suggests that proximity-based connectivity provides sufficient relational information for this dataset, and that the more flexible attention-based models do not gain a measurable advantage from their topology-free design.

A secondary observation is that the STGCN variant evaluated here uses a proximity-based dynamic graph rather than a fully static one: the adjacency matrix is recomputed at each timestep based on current inter-vehicle distances, rather than being fixed to the initial configuration of the episode. This dynamic graph construction partially closes the gap between graph-based and attention-based approaches, because it allows the spatial topology to track agent movements over time. The remaining gap relative to unconstrained self-attention lies in the binary nature of the adjacency entries: two vehicles are either connected or not, whereas self-attention computes a continuous relevance weight for every pair. Whether this binary-versus-continuous distinction is practically significant on larger or more diverse datasets remains an open question.

Both classes correspond to transient events (specifically, a cut-in or cut-out) that occupy only a fraction of the window duration. During the majority of a CINF or COUTF window, the ego vehicle is simply following traffic in a state that resembles an FRF or FLF scenario; the defining event (the lateral movement of a neighbour into

or out of the ego lane) occurs late in the sequence and may not dominate the temporal context. This makes the class temporally ambiguous and may explain why recall is low even for models with full multi-agent context. Future work could address this through label-aware temporal weighting or event-detection preprocessing that up-weights the transition region.

Beyond the aggregate macro-F1 score, the per-class confusion structure is informative. The three highest-performing individual classes across all interaction-aware models are Braking Front (BF), Free-Ride (FRF) all of which are characterised by sustained and kinematically unambiguous ego states. BF is identified primarily through high ego deceleration; FR and FRF by low relative velocity and large headway. These features are salient over most of the window duration, which makes them robust to the temporal pooling mechanism. By contrast, the lane-change classes (LCF, CINF, COUTF) show intermediate difficulty: the defining lateral displacement is sustained long enough to dominate temporal attention, but the confusion between LCF and the ego-following portion of a CINF window inflates false negatives in both directions. The FLF (Follow Leading Vehicle Far) class, which requires precisely characterising the distance gap to the leading vehicle, is consistently confused with FRF across all models, suggesting that the feature resolution of the input representation near the headway boundary is insufficient at the current frame rate and quantisation precision.

The key takeaway from the experimental evaluation is not which architecture performs best, given that the differences among interaction-aware models are negligible in absolute terms, but rather the qualitative shift that occurs when multi-agent information is introduced. The transition from ego-only to multi-agent context accounts for most of the performance gap observed across all metrics and all classes.

Within the interaction-aware tier, the practical differentiators are computational cost (the factorised STT has lower asymptotic complexity than the Flat Transformer at scale), topology requirements (STGCN requires a fixed graph, the Transformer-based models do not), and training stability (the STT shows notably lower seed-to-seed variance). These properties, rather than peak accuracy on this synthetic benchmark,

should guide architecture selection in deployment-oriented scenarios where robustness and efficiency are first-order concerns.

A final caveat concerns the synthetic nature of the evaluation dataset. CARLA simulates traffic with scripted agents following deterministic rules, which means the feature distributions are cleaner and less ambiguous than those encountered in real-world recordings. Performance on real-world data from platforms such as the Next Generation Simulation (NGSIM) dataset or naturalistic driving studies may differ substantially, and the relative ordering of models could shift under more complex and stochastic traffic patterns.

6. Conclusions and Future Work

This thesis addressed the problem of automatic driving scenario recognition from multi-agent kinematic time series data. The work produced two main contributions.

The first contribution is a synthetic labelled benchmark of 31,502 scenario windows extracted from CARLA-simulated multi-agent recordings. Nine scenario classes are defined in accordance with the ISO 34502 taxonomy, covering braking, lane-change, cut-in, cut-out, and following-distance scenarios, and each window is encoded in the ASAM Open Simulation Interface (OSI) format, enabling compatibility with standardised simulation toolchains. The benchmark provides reproducible train/validation/test splits and serves as a controlled evaluation platform for scenario recognition methods.

The second contribution is the Spatio-Temporal Transformer (STT), a factorised Transformer architecture that alternates spatial multi-head attention across agent slots with temporal multi-head attention along the time axis. The model is trained under a multi-task learning framework combining focal loss with class-balanced sampling for the primary classification objective and a Huber-loss regression objective for three ego-vehicle kinematic targets, with task weights learned via homoscedastic uncertainty weighting. On the test partition, the STT achieves a macro-F1 of 0.732 ± 0.005, matching the computationally more expensive Flat Transformer (0.733 ± 0.020) and the graph-based STGCN (0.730 ± 0.007), while exhibiting lower seed-to-seed variance than either alternative.

The experimental evaluation established a clear hierarchy of performance tied to the availability of multi-agent context. The ego-only LSTM-ego baseline falls 0.288 macro-F1 points below the interaction-aware models (0.444 vs. 0.732), and the FLF_B3 class whose defining criterion is the relative distance to the leading vehicle achieves exactly F1 = 0.000 for the LSTM-ego, confirming that certain scenario classes are fundamentally inaccessible without inter-agent information. The auxiliary regression evaluation further showed that ego-centric kinematic targets (mean speed,

peak deceleration) are predicted with comparable MAE by both ego-only and interaction-aware models, validating the multi-task design: the classification and regression objectives exploit complementary information and do not interfere with each other.

6.1 Future work

Several directions for future research are identified below.

Real-world validation. The most immediate priority is evaluating the STT and the benchmark pipeline on real-world recordings. Candidate datasets include the highD dataset for highway scenarios, the inD/rounD/exiD family for intersection and roundabout settings, and publicly available naturalistic driving study corpora. Such evaluation would quantify the synthetic-to-real performance gap and identify which scenario classes suffer most from domain shift.

A particularly valuable evaluation protocol would be a zero-shot transfer experiment: training exclusively on the synthetic CARLA benchmark and evaluating directly on a real-world dataset such as highD without any fine-tuning. The resulting performance gap would provide a direct measure of the synthetic-to-real shift attributable to the absence of sensor noise, human behavioural variability, and occlusion effects. If the zero-shot gap is small for certain classes (e.g., BF, FRF) but large for others (e.g., CINF, COUTF), this would inform a selective fine-tuning strategy that targets only the most domain-sensitive classes, reducing the annotation burden on real-world data.

Event-aware temporal modelling. Recognition of CINF and COUTF would likely benefit from a temporal attention mechanism that learns to up-weight the onset region of the defining event rather than integrating over the full window uniformly. Candidate approaches include temporal event detection as a preprocessing step that identifies the transition region within each window, or causal attention masking that constrains the temporal receptive field to emphasise recent context.

A complementary approach is to reformulate the classification problem as a change-point detection task. Rather than assigning a single label to the full window, the model would predict at each timestep the probability that a scenario transition has occurred, jointly localising the event onset and classifying the post-transition state. This reformulation is more demanding in terms of temporal supervision (requiring event-level timestamps rather than window-level labels) but directly addresses the fundamental difficulty of CINF and COUTF: the label is dominated by a brief transition that the uniform pooling mechanism cannot prioritise without explicit temporal supervision.

Patch-based temporal tokenisation. Inspired by PatchTST, partitioning each agent's temporal sequence into non-overlapping patches before applying temporal attention could improve representational efficiency and reduce sensitivity to window length, particularly for longer recording horizons beyond the 110 timesteps used in this work.

Online and streaming recognition. The current approach assumes a fixed-length historical window and produces a single label per window in batch mode. An online variant that updates the scenario prediction at each new timestep, using a causal Transformer or an attention-augmented recurrent state, would be more suitable for real-time ADAS deployment, where low-latency scenario awareness is a first-order requirement.

Transfer learning from synthetic to real. The synthetic benchmark could serve as a pretraining source for a transfer learning pipeline, with fine-tuning on small amounts of annotated real-world data. Domain adaptation techniques such as feature-level distribution alignment or adversarial training could reduce the synthetic-to-real gap and allow the benchmark to act as a low-cost substitute for expensive real-world annotation.

Self-supervised pre-training offers an alternative that does not require any real-world labels. A masked autoencoding objective, as used in TimeMAE (Cheng et al., 2023), would train the STT encoder to reconstruct randomly masked timesteps and agent slots

from unlabelled CARLA recordings, learning rich kinematic representations before any scenario label is introduced. The pre-trained encoder could then be fine-tuned on the labelled subset with standard cross-entropy loss, potentially achieving higher accuracy on CINF and COUTF by starting from a stronger feature initialisation than random weights. The large volume of unlabelled CARLA data available from the same simulator configuration makes this direction practically feasible without additional data collection.

Extended scenario taxonomy. The current nine-class taxonomy covers common highway scenarios but omits simultaneous or compound events (e.g., a cut-in combined with emergency braking) and intersection-specific behaviours. Extending the CARLA data generation pipeline to cover a richer taxonomy, including multi-label scenarios and rare edge cases, would increase the ecological validity of the benchmark and challenge future recognition models more severely.

Systematic STT architectural ablation. As noted in Section 6.2, the experimental evaluation does not ablate the individual components of the STT: the number of spatial encoder layers, the number of temporal encoder layers, the embedding dimension, the presence or absence of the Ego Cross-Attention module, and the temporal pooling mechanism (attention pooling versus mean pooling versus last-hidden-state). A systematic ablation along each axis, holding all others fixed, would identify which components are essential for the observed performance and which could be simplified without loss of accuracy. The Ego Cross-Attention module is of particular interest: its theoretical motivation is strong, but its empirical contribution relative to a model that uses only the masked mean of all agent tokens has not been quantified. If the module proves non-essential, the architecture could be simplified while retaining interpretability by substituting a standard mean aggregation, reducing the parameter count and inference cost.

Bibliography

[1] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A Survey of Autonomous Driving: Common Practices and Emerging Technologies," *IEEE Access*, vol. 8, pp. 58443–58469, 2020, doi: 10.1109/ACCESS.2020.2983149.

[2] ASAM e.V., "ASAM OSI: Open Simulation Interface Specification," Association for Standardisation of Automation and Measuring Systems, Graefelfing, Germany, Version 3.7.0, 2023. [Online]. Available: https://www.asam.net/standards/detail/osi/

[3] S. Mozaffari, O. Y. Al-Jarrah, M. Dianati, P. Jennings, and A. Mouzakitis, "Deep Learning-Based Vehicle Behavior Prediction for Autonomous Driving Applications: A Review," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 1, pp. 33–47, 2022, doi: 10.1109/TITS.2020.3012034.

[4] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator," in *Proceedings of the 1st Annual Conference on Robot Learning (CoRL 2017)*, in Proceedings of Machine Learning Research, vol. 78. PMLR, 2017, pp. 1–16.

[5] H. Caesar *et al.*, "nuScenes: A Multimodal Dataset for Autonomous Driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11621–11631. doi: 10.1109/CVPR42600.2020.01164.

[6] J. Jansson and F. Gustafsson, "A Framework and Automotive Application of Collision Avoidance Decision Making," in *IFAC Proceedings Volumes*, 2002.

[7] C. Laugier *et al.*, "Probabilistic Analysis of Dynamic Scenes and Collision Risk Assessment to Improve Driving Safety," *IEEE Intell. Transp. Syst. Mag.*, vol. 3, no. 4, pp. 4–19, 2011.

[8] S. Klingelschmitt, M. Platho, H.-M. Groß, V. Willert, and J. Eggert, "Recognition of Cooperative and Non-Cooperative Behavior of Vehicle Drivers," in *IEEE Intelligent Vehicles Symposium (IV)*, 2014.

[9] Y. Xing, C. Lv, and D. Cao, "Driver Lane Change Intention Inference for Intelligent Vehicles: Framework, Survey, and Challenges," *IEEE Trans. Veh. Technol.*, 2020.

[10] N. Khairdoost, M. Shirpour, M. A. Bauer, and S. S. Beauchemin, "Real-Time Driver Maneuver Prediction Using LSTM," *IEEE Trans. Intell. Veh.*, 2020.

[11] D. Dörr and et al, "Driving Maneuver Classification from Time Series Data: A Rule Based Machine Learning Approach," *Appl. Intell.*, 2022, [Online]. Available: https://link.springer.com/article/10.1007/s10489-022-03328-3

[12] A. Vaswani *et al.*, “Attention Is All You Need,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. [Online]. Available: https://arxiv.org/abs/1706.03762

[13] K. W. Church and et al, “Positional Encoding in Transformer-Based Time Series Models: A Survey,” *ArXiv Prepr. ArXiv250212370*, 2025, [Online]. Available: https://arxiv.org/abs/2502.12370

[14] Q. Wen *et al.*, “Transformers in Time Series: A Survey,” *ArXiv Prepr. ArXiv220207125*, 2022, [Online]. Available: https://arxiv.org/abs/2202.07125

[15] N. M. Foumani, L. Miller, C. W. Tan, G. I. Webb, G. Forestier, and M. Salehi, “Deep Learning for Time Series Classification and Extrinsic Regression: A Current Survey,” *ACM Comput. Surv.*, 2024, [Online]. Available: https://arxiv.org/abs/2302.02515

[16] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, “A Time Series is Worth 64 Words: Long-term Forecasting with Transformers,” in *International Conference on Learning Representations (ICLR)*, 2023. [Online]. Available: https://arxiv.org/abs/2211.14730

[17] S. Yang and et al, “Superb: Speech Processing Universal Performance Benchmark,” *ArXiv Prepr. ArXiv210501051*, 2021.

[18] H. Wang *et al.*, “STGformer: Efficient Spatiotemporal Graph Transformer for Traffic Forecasting,” *ArXiv Prepr. ArXiv241000385*, 2024, [Online]. Available: https://arxiv.org/abs/2410.00385

[19] K. Choromanski *et al.*, “Rethinking Attention with Performers,” Nov. 19, 2022, *arXiv*: arXiv:2009.14794. doi: 10.48550/arXiv.2009.14794.

[20] A. Das, W. Kong, R. Sen, and Y. Zhou, “A decoder-only foundation model for time-series forecasting,” Apr. 17, 2024, *arXiv*: arXiv:2310.10688. doi: 10.48550/arXiv.2310.10688.

[21] M. Goswami, K. Szafer, A. Choudhry, Y. Cai, S. Li, and A. Dubrawski, “MOMENT: A Family of Open Time-series Foundation Models,” Oct. 10, 2024, *arXiv*: arXiv:2402.03885. doi: 10.48550/arXiv.2402.03885.

[22] G. Bertasius, H. Wang, and L. Torresani, “Is Space-Time Attention All You Need for Video Understanding?,” in *International Conference on Machine Learning (ICML)*, 2021. [Online]. Available: https://arxiv.org/abs/2102.05095

[23] H. Wang *et al.*, “STGformer: Efficient Spatiotemporal Graph Transformer for Traffic Forecasting,” *ArXiv Prepr. ArXiv241000385*, 2024, [Online]. Available: https://arxiv.org/abs/2410.00385

[24] J. Zhou, E. Liu, W. Chen, S. Zhong, and Y. Liang, “Navigating Spatio-Temporal Heterogeneity: A Graph Transformer Approach for Traffic Forecasting,” *ArXiv Prepr. ArXiv240810822*, 2024, [Online]. Available: https://arxiv.org/abs/2408.10822

[25] J. Wu *et al.*, “Goal-Guided Graph Attention Network with Interactive State Refinement for Multi-Agent Trajectory Prediction,” *Sensors*, vol. 24, no. 7, p. 2065, 2024.

[26] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, “Deep Networks with Stochastic Depth,” in *European Conference on Computer Vision (ECCV)*, 2016. [Online]. Available: https://arxiv.org/abs/1603.09382

[27] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, “Deep Networks with Stochastic Depth,” in *European Conference on Computer Vision (ECCV)*, 2016. [Online]. Available: https://arxiv.org/abs/1603.09382

[28] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long, “TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis,” in *International Conference on Learning Representations (ICLR)*, 2023. [Online]. Available: https://arxiv.org/abs/2210.02186

[29] B. Yu, H. Yin, and Z. Zhu, “Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting,” in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2018. [Online]. Available: https://arxiv.org/abs/1709.04875

[30] R. Caruana, “Multitask Learning,” *Mach. Learn.*, vol. 28, pp. 41–75, 1997.

[31] A. Kendall, Y. Gal, and R. Cipolla, “Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7482–7491.

[32] A. Kendall, Y. Gal, and R. Cipolla, “Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7482–7491.

[33] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal Loss for Dense Object Detection,” in *IEEE International Conference on Computer Vision (ICCV)*, 2017. [Online]. Available: https://arxiv.org/abs/1708.02002

[34] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, “mixup: Beyond Empirical Risk Minimization,” in *International Conference on Learning Representations (ICLR)*, 2018. [Online]. Available: https://arxiv.org/abs/1710.09412

[35] I. Loshchilov and F. Hutter, “Decoupled Weight Decay Regularization,” in *International Conference on Learning Representations (ICLR)*, 2019. [Online]. Available: https://arxiv.org/abs/1711.05101

[36] L. N. Smith, “A Disciplined Approach to Neural Network Hyper-Parameters: Part 1 – Learning Rate, Batch Size, Momentum, and Weight Decay,” *ArXiv Prepr. ArXiv180309820*, 2018.

[37] Y. Gal and Z. Ghahramani, “Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning,” in *International Conference on Machine Learning (ICML)*, 2016.

[38] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “CARLA: An Open Urban Driving Simulator,” in *Proceedings of the 1st Annual Conference on Robot Learning (CoRL)*, 2017. [Online]. Available: https://arxiv.org/abs/1711.03938

[39] Institute for Automotive Engineering (ika), RWTH Aachen University, “Omega-Prime: Data Model, Data Format and Python Library for Handling Ground Truth Traffic Data.” 2025. [Online]. Available: https://github.com/ika-rwth-aachen/omega-prime

[40] Foxglove, “MCAP: A Modular, Performant, and Serialization-Agnostic Container File Format.” 2022. [Online]. Available: https://github.com/foxglove/mcap

[41] ASAM e.V., “ASAM OSI (Open Simulation Interface) Specification, v3.7.0 ‘Jolly Jones.’” 2023. [Online]. Available: https://github.com/OpenSimulationInterface/open-simulation-interface/releases/tag/v3.7.0

[42] ASAM e.V., “ASAM OpenDRIVE Specification, V1.7.0.” 2021. [Online]. Available: https://www.asam.net/standards/detail/opendrive/older/

[43] R. Xiong *et al.*, “On Layer Normalization in the Transformer Architecture,” in *International Conference on Machine Learning (ICML)*, 2020.

Appendix

This appendix presents high-level pseudocode for the four main algorithmic components of the thesis: the CARLA-to-OSI data conversion pipeline, the STT forward pass, the multi-task training loop, and the test-time augmentation evaluation procedure.

Algorithm 1: CARLA to Omega-Prime Conversion Pipeline

Transforms a raw CARLA binary recorder log into a standardised OSI-encoded MCAP file ready for model training.

```
Input:  L  , CARLA binary recorder log file
Output: M  , OSI-encoded MCAP file

procedure CONVERT(L):
          // Phase 1 , lightweight single-pass metadata scan
          header  <- SCAN_HEADER(L)
          actors  <- EXTRACT_ACTOR_LIST(L)

          // Phase 2 , frame-by-frame reconstruction
          for each frame f in L do
            for each actor a in actors do
              state[a,f] <- RECONSTRUCT_KINEMATIC_STATE(a, f, L)
            end for

            // Coordinate system conversion
            // CARLA: left-handed (Unreal Engine)  ->  OSI: right-handed
            for each actor a do
              state[a,f].y       <- -state[a,f].y
              state[a,f].heading <- -state[a,f].heading
            end for

            // Temporal resampling , fill gaps to enforce 10 Hz
            if TIMESTAMP_GAP(f) > MAX_ALLOWED_GAP then
              state <- LINEAR_INTERPOLATE(state[f-1], state[f])
            end if

            // Build and validate OSI GroundTruth message
            gt_msg <- BUILD_OSI_GROUNDTRUTH(state, header, f)
            if not VALIDATE_PLAUSIBILITY(gt_msg) then
              MARK_INVALID(gt_msg)   // flagged, not discarded
            end if

            WRITE_MCAP_MESSAGE(M, gt_msg)
          end for
return M
end procedure
```

Algorithm 2: Spatio-Temporal Transformer (STT) Forward Pass

Maps a batch of multi-agent kinematic tensors to classification logits and kinematic regression predictions.

```
Input:  X in R^{B x T x K x F}  , batch of agent features
        M in {0,1}^{B x K}       , valid-agent mask
Output: logits   in R^{B x C}    , class scores (C = 9)
        reg_pred in R^{B x R}    , regression targets (R = 3)

procedure STT_FORWARD(X, M):
    // Input projection: F -> d_model for every (agent, timestep)
    E <- LINEAR_PROJ(X)               // B x T x K x d_model

    // Spatial encoder , operates per timestep, across K agent slots
    for t = 1 to T do
      E[:,t,:,:] <- TRANSFORMER_ENC(E[:,t,:,:], mask=M)
                                      // B x K x d_model per t
    end for

    // Ego cross-attention , query=ego token (slot 0), key/value=all K
    for t = 1 to T do
      q      <- E[:,t,0,:]             // ego query: B x d_model
      k, v   <- E[:,t,:,:]             // all tokens: B x K x d_model
      H[:,t] <- CROSS_ATTN(q, k, v, mask=M)  // B x d_model per t
    end for
    // H: B x T x d_model , ego-centric context sequence

    // Add sinusoidal positional encoding along time axis
    H <- H + SINUSOIDAL_PE(T, d_model)

    // Temporal encoder , models evolution across T timesteps
    H <- TEMPORAL_TRANSFORMER_ENC(H)   // B x T x d_model

    // Temporal attention pooling -> fixed-size scenario representation
    z <- ATTN_POOL(H)                  // B x d_model

    // Multi-task output heads
    logits   <- LINEAR(z)              // B x C  (classification)
    reg_pred <- LINEAR(z)              // B x R  (regression)

return logits, reg_pred
end procedure
```

Algorithm 3: Multi-Task Training Loop (MTL with Kendall Weighting)

Trains the STT with Focal Loss (classification) and Huber Loss (regression), with task weights learned via homoscedastic uncertainty weighting [32].

```
Input:  D , training dataset, theta , model parameters
        sigma_c, sigma_r , learnable log-uncertainty parameters
        E , number of epochs
Output: theta* , trained model
```

```
procedure TRAIN(D, theta, sigma_c, sigma_r, E):
          optimizer <- AdamW(theta + {sigma_c, sigma_r},
                             lr=1e-4, weight_decay=1e-2)
          scheduler <- ONE_CYCLE_LR(optimizer, max_lr=1e-3)

          for epoch = 1 to E do
            for each mini-batch (X, M, y_cls, y_reg) in D do

                // Mixup data augmentation (alpha=0.3)
                lam       <- BETA_SAMPLE(alpha=0.4)
                X_mix     <- lam*X + (1-lam)*SHUFFLE(X)
                y_cls_mix <- (lam, y_cls, (1-lam), SHUFFLE(y_cls))

                // Forward pass
                logits, reg_pred <- STT_FORWARD(X_mix, M)

                // Task losses
                L_cls <- FOCAL_LOSS(logits, y_cls_mix, gamma=1.5)
                L_reg <- HUBER_LOSS(reg_pred, y_reg, delta=1)

                // Kendall uncertainty weighting
                // L = L_cls/(2*exp(2*s_c)) + s_c  +  L_reg/(2*exp(2*s_r))
              + s_r
                L <- L_cls / (2 * exp(sigma_c)) + sigma_c
                   + L_reg / (2 * exp(sigma_r)) + sigma_r

                // Backprop + gradient clipping
                BACKWARD(L)
                CLIP_GRAD_NORM(theta, max_norm=0.5)
                optimizer.STEP()
                scheduler.STEP()
            end for
          end for
return theta
end procedure
```

Algorithm 4: Test-Time Augmentation (TTA) Evaluation

Produces final test-set predictions by averaging logits across P stochastic forward passes (dropout active), then computes macro-F1.

```
Input:  D_test , test dataset, theta , trained model
        P = 5  , number of TTA passes
Output: y_hat , predicted class labels, F1_macro , macro-F1 score

procedure EVALUATE_TTA(D_test, theta, P):
          all_logits <- []   // will hold P tensors of shape [N, C]
          all_reg    <- []   // will hold P tensors of shape [N, R]

          theta.ENABLE_DROPOUT()   // model.train() , activates stochasticity

          for p = 1 to P do
              logits_p, reg_p <- [], []
              for each mini-batch (X, M) in D_test do
                l, r <- STT_FORWARD(X, M)   // no gradient tracking
```

```
            APPEND(logits_p, l)
            APPEND(reg_p,    r)
          end for
          APPEND(all_logits, CONCAT(logits_p))  // [N, C]
          APPEND(all_reg,    CONCAT(reg_p))     // [N, R]
        end for

        // Average across P TTA passes
        avg_logits <- MEAN(all_logits, axis=0)   // [N, C]
        avg_reg    <- MEAN(all_reg,    axis=0)   // [N, R]

        // Hard-label classification
        y_hat <- ARGMAX(avg_logits, axis=1)      // [N]

        // Macro-F1: unweighted mean of per-class F1 scores
        F1_macro <- MEAN([ F1_SCORE(y_hat, y_true, class=c)
                           for c in 0..C-1 ])

    return y_hat, F1_macro
    end procedure
```