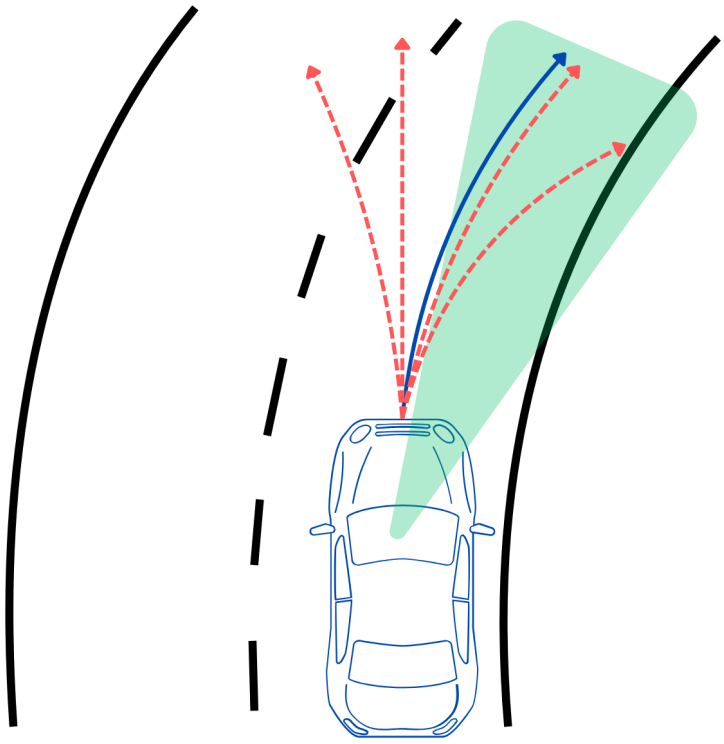
CHALMERS
UNIVERSITY OF TECHNOLOGY



Fusion of Human Context for Vehicle Trajectory Prediction

Incorporating Driver Monitoring Data for Short-Horizon ADAS Applications

Master's thesis in Complex Adaptive Systems

Samuel Armgren
Robert Kupferschmied

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's thesis 2026

Fusion of Human Context for Vehicle Trajectory Prediction

Incorporating Driver Monitoring Data for Short-Horizon ADAS Applications

Samuel Armgren
Robert Kupferschmied

CHALMERS
UNIVERSITY OF TECHNOLOGY

APTIV

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2026

Fusion of Human Context for Vehicle Trajectory Prediction,
Incorporating Driver Monitoring Data for Short-Horizon ADAS Applications.
Samuel Armgren
Robert Kupferschmied

Advisor: Erik Larsson, Aptiv
Supervisor: Aleksandar Mitrevski, Electrical Engineering
Examiner: Karinne Ramirez-Amaro, Electrical Engineering

Master's Thesis 2026
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Birds eye view of a vehicle trajectory prediction. Green area indicates driver gaze zone.

Typeset in LaTeX
Gothenburg, Sweden 2026

Fusion of Human Context for Vehicle Trajectory Prediction,
Incorporating Driver Monitoring Data for Short-Horizon ADAS Applications.
Samuel Armgren
Robert Kupferschmied
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Short-horizon ego-vehicle trajectory prediction is a key component of Advanced Driver Assistance Systems, particularly in situations where driver behavior directly influences vehicle motion. Classical kinematic models are limited in such cases, as they rely solely on observed motion and do not account for anticipatory cues related to driver awareness. This thesis investigates whether gaze information from a Driver Monitoring System can improve short-term trajectory prediction without relying on environmental perception.

A learning-based model, termed **Gaze-Aware Dual-Stream Trajectory Prediction (GA-DSTP)**, is proposed. The model predicts 3.5 seconds future ego-vehicle trajectories by jointly leveraging temporal histories of vehicle dynamics, actuator inputs, and driver gaze data. Vehicle-centric and driver-centric information are processed in separate streams and fused to capture both physical motion tendencies and driver-based indicators of upcoming maneuvers. A multi-modal prediction formulation is used to represent uncertainty in future motion.

The model is trained and evaluated on a self-collected real-world driving dataset and compared against classical kinematic baselines as well as the same model trained without DMS data. GA-DSTP achieves a 37% reduction in Average Displacement Error (ADE) compared to the Constant Turn Rate and Acceleration (CTRA) model and over 60% compared to the Constant Velocity (CV) model. It also achieves a 2.1% lower ADE than the model not using DMS data, indicating that gaze information provides complementary predictive value beyond vehicle dynamics alone. The results demonstrate that driver gaze provides complementary information to vehicle dynamics and can enhance the robustness of ego-vehicle trajectory prediction for ADAS applications.

Keywords: vehicle trajectory prediction, driver monitoring systems, dual-stream learning, advanced driver assistance systems, driver awareness modeling

Acknowledgements

We would like to thank Aptiv for the opportunity and their support throughout this project during the course of this thesis. We are especially grateful to our supervisor at Aptiv, Erik Larsson together with his team, for the valuable input and continuous guidance. We also thank our supervisor and examiner at Chalmers, Aleksandar Mitrevski and Karinne Ramirez-Amaro, for constructive feedback and insightful comments.

Samuel Armgren and Robert Kupferschmied
Gothenburg, June 2026

Declaration of Originality

We, the undersigned below, declare that this work has not previously been submitted to this or any other university and that it is, unless otherwise stated, entirely our own work. The report was, in part, written with the help of the AI assistant *Copilot*, as described in the Appendix A.1. Also, parts of our code was generated with the help of *Claude Opus*. We are aware that content generated by AI systems is no substitute for careful scientific work, which is why all AI-generated content has been critically reviewed by us, and we take full responsibility for it.

Date

Samuel Armgren

Robert Kupferschmied

Contents

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

AEB	Automatic Emergency Braking
AES	Automatic Emergency Steering
ADAS	Advanced Driver Assistance Systems
ADE	Average Displacement Error
BERT	Bidirectional Encoder Representations from Transformers
CA	Constant Acceleration
CNN	Convolutional Neural Network
CTRA	Constant Turn Rate and Acceleration
CTRV	Constant Turn Rate and Velocity
CV	Constant Velocity
DETR	Detection Transformer
DMS	Driver Monitoring System
DSTP	Dual Stream Trajectory Prediction
FDE	Final Displacement Error
GA-DSTP	Gaze Aware - Dual Stream Trajectory Prediction
GRU	Gated Recurrent Unit
IQR	Interquartile Range
LSTM	Long Short-Term Memory
ML	Machine Learning
NLP	Natural Language Processing

1
Introduction

Advanced Driver Assistance Systems (ADAS) rely on accurate short-horizon trajectory predictions to support safe and reliable operation in complex and time-critical driving scenarios [1]. As vehicles approach potentially hazardous situations, the accuracy of these predictions becomes increasingly important, since even small deviations in predicted motion may significantly affect downstream system behaviour. Recent studies show that the complexity of ADAS continues to increase as new safety and automation features are added each year [2, 3]. Given this accelerating technological evolution, it is essential that the baseline trajectory prediction models used in ADAS development are updated accordingly to ensure accurate, reliable, and future proof optimization.

Traditional motion models, such as Constant Turn Rate and Acceleration (CTRA), provide reliable baselines for trajectory prediction but operate solely on vehicle dynamics and assume simplified motion behaviour. These assumptions tend to break down during human-driven maneuvers such as sudden lane changes, sharp turns, or evasive actions [4]. As classical models lack behavioural cues, they are inherently limited in their ability to anticipate driver intent prior to observable vehicle motion. An example of this is shown in Figure 1.1

Driver Monitoring Systems (DMS) have become increasingly common in modern vehicles as automakers integrate more in-cabin safety technologies, often used to detect drowsy or distracted drivers [5]. These systems typically use an inward-facing camera to track the driver's head position, gaze direction, hand placement, and foot movements. The captured signals can be expressed as numerical feature vectors. For example, the head's (x, y, z) position, gaze orientation represented by angular components, or the foot's angle relative to the pedals. Instead of processing raw images directly, these structured feature vectors can be used as inputs to a Machine Learning (ML) model, significantly reducing computational complexity while still capturing the essential behavioral information needed for driver monitoring [5]. When combined with actuator inputs (e.g., brake, steering, and accelerator signals) and host vehicle state information (e.g., speed and yaw rate), DMS data offers the potential to enhance short-term trajectory prediction by incorporating cues about what the driver is likely to do next.

Incorporating such heterogeneous and behaviour-related signals introduces complex and potentially nonlinear relationships that are difficult to capture using conven-

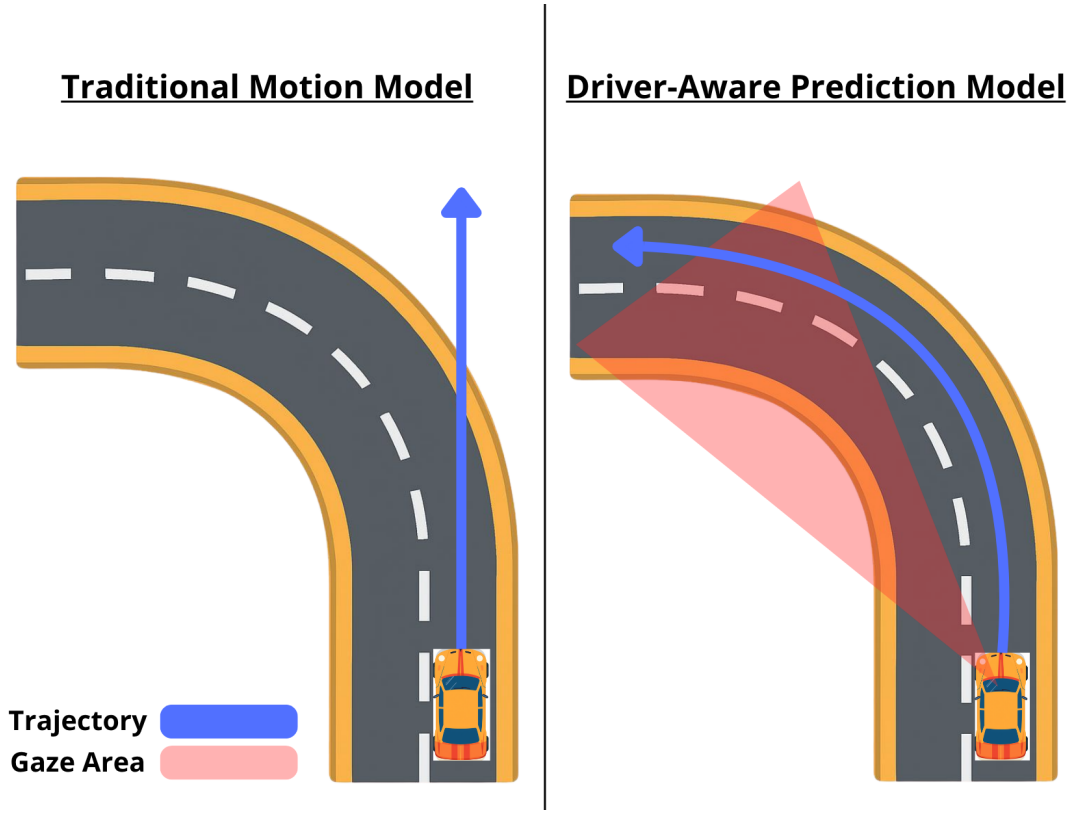


Figure 1.1: The figure shows an example of a traditional motion model prediction on the left and a proposed driver aware prediction on the right.

tional analytical motion models. Therefore, investigating data-driven methods, such as ML approaches, that fuse host vehicle state information with DMS and actuator data for short-term trajectory prediction is of interest. It has been shown in similar previous work that incorporating DMS data improves performance of ADAS functionalities [6], further motivating the exploration of such approaches in this project.

1.1 Objectives and Research Questions

The aim of this thesis is to investigate the use of a ML-based method for short-term vehicle trajectory prediction by fusing host vehicle state information with DMS data and actuator signals. The intended outcome is a model that can generate accurate short-horizon trajectory predictions and an analysis of the potential benefits of incorporating driver-related information compared to traditional motion-based approaches. The following research questions are formulated to specify the issues being investigated:

- RQ1: Can data-driven trajectory prediction models outperform traditional analytical motion models for short-term vehicle trajectory prediction?
- RQ2: Does the inclusion of DMS data improve the performance of data-driven short-term trajectory prediction models?

1.2 Summary

This thesis aims to improve short-horizon vehicle trajectory prediction for Advanced Driver Assistance Systems by incorporating information related to human driving behavior. The central objective is to better anticipate a vehicle's near-future motion in scenarios where driver intent significantly influences the outcome, such as turning or low-speed maneuvers, and where traditional motion-based approaches may be insufficient.

To achieve this, the project adopts a data-driven approach that combines vehicle state information with driver-related signals, enabling the prediction model to account for both physical motion and human intent. By learning temporal patterns from real driving data, the proposed method seeks to produce more reliable trajectory predictions than conventional kinematic models. The effectiveness of this approach is assessed through comparative evaluation against established baseline methods.

2
Related Work

Vehicle trajectory prediction is a fundamental component of ADAS and autonomous driving, where accurate short-horizon forecasts are required for safe and efficient decision-making. Existing approaches can broadly be divided into two main categories: model-based and learning-based methods.
Model-based approaches rely on analytical formulations of vehicle dynamics and are typically used for short-term prediction due to their interpretability and computational efficiency. In contrast, learning-based approaches leverage data-driven models to capture complex motion patterns and multimodal future behaviors. Recent research has increasingly focused on incorporating environmental context, driver behavior, and multimodal prediction capabilities to improve prediction accuracy.
The following sections review representative work in both categories and position this thesis within the broader research landscape.

2.1 Model-Based Trajectory Prediction

Model-based trajectory prediction methods rely on analytical motion models that describe vehicle motion using simplified kinematic assumptions. These models are widely used in vehicle tracking, state estimation, and short-term trajectory prediction within intelligent transportation systems (ITS) and ADAS applications [7].
Common formulations include constant-velocity (CV), constant-acceleration (CA), constant-turn-rate and velocity (CTRV), and constant-turn-rate and acceleration (CTRA). These models are frequently employed together with recursive filtering techniques such as Kalman filters to estimate and predict vehicle motion in real time, forming a standard baseline in many tracking and prediction systems [7].
Their popularity is largely due to their interpretability, low computational cost, and robustness in short-horizon prediction tasks. However, these models are based on simplified assumptions of vehicle dynamics and do not explicitly capture driver intent, interaction with other road users, or complex nonlinear behaviors. As a result, their performance is limited in more dynamic and uncertain driving scenarios, motivating the development of learning-based approaches.

2.2 Learning-Based Trajectory Prediction

Khakzar et al. propose a **Dual Learning Model** for vehicle trajectory prediction that explicitly incorporates interactions with surrounding vehicles through risk-aware representations [8]. Their approach combines an LSTM-based encoder–decoder

architecture with two distinct spatio-temporal inputs: an Occupancy Map, which encodes the positions and velocities of nearby vehicles, and a Risk Map, which represents interaction risk using time-to-collision measures. A ConvLSTM network is employed to learn dynamic interaction patterns from these maps, enabling the model to account for both motion history and collision risk when predicting future trajectories. The model is evaluated on large-scale real-world highway datasets (NGSIM and HighD) and demonstrates improved prediction accuracy over several state-of-the-art interaction-aware methods.

A potential limitation of this approach is its dependence on accurate perception of surrounding vehicles and interaction modeling, which may reduce robustness in environments with incomplete or noisy sensor data. Furthermore, the evaluation is restricted to highway scenarios, limiting insight into performance in more complex urban settings such as intersections or roundabouts.

Zhang et al. introduces another trajectory prediction framework **TFE-PID**, focused on the ego vehicle, emphasizing the role of driver intention rather than surrounding traffic [9]. The proposed method employs a time-feature encoding mechanism using a bidirectional GRU with attention to extract temporal driving patterns. These features are combined with a physics–intention decoding strategy, where short-term predictions rely on physical vehicle dynamics and longer-term predictions are guided by recognized driver intentions obtained through a coupled hidden Markov model. The approach is designed to operate without environmental perception data, relying solely on ego vehicle states and driver-related signals. Experiments are conducted on a human-in-the-loop dataset collected in controlled driving scenarios, demonstrating strong short-horizon prediction accuracy and stable performance.

A limitation of this approach is that it is evaluated primarily on simulated, human-in-the-loop data and focuses on a limited set of predefined driving scenarios. As a result, the method is scenario-specific and may not generalize well to the full diversity of real-world driving conditions.

Varadarajan et al. propose another framework called **MultiPath++**, a large-scale trajectory prediction framework designed to model highly multimodal future behaviors of traffic participants [10]. The method represents the driving environment using sparse, structured inputs, including agent state histories, road network polylines, and interaction features between agents. These heterogeneous inputs are encoded separately and fused using a multi-context gating mechanism that enables efficient information sharing across modalities. Future trajectories are predicted as a Gaussian Mixture Model with learned anchor embeddings, allowing the model to capture multiple plausible motion outcomes. MultiPath++ achieves state-of-the-art performance, outperforming the other compared approaches on large-scale datasets such as the Waymo Open Motion Dataset and Argoverse.

Criterion	**Dual Learning Model**	**TFE-PID**	**MultiPath++**	**This Thesis**
Dataset	NGSIM, HighD (Highway dataset)	Simulated human-in-the-loop	Waymo Open Motion Dataset	Real world collected dataset
Uses real-world data	✓	✗	✓	✓
Uses general (diverse) data	✗	✗	✓	✓
Uses driver awareness data	✗	✓	✗	✓
Uses perception features	✓	✗	✓	✗

Table 2.1: Comparison of related work and the proposed thesis approach

Despite its strong predictive performance, the method relies heavily on detailed map data and rich perception features, including road geometry and multi-agent interactions. This dependence can limit applicability in settings where such information is incomplete, unavailable, or unreliable. In addition, the model architecture is designed for large-scale prediction of multiple agents over long horizons, which increases computational complexity and may be unnecessary for short-horizon applications where driver awareness and immediate intent play a more central role.

Table 2.1 compares related work to this thesis, showing the differences and similarities. TFE-PID and this thesis create a new dataset while the others use existing ones. All of the work uses real-world data except for TFE-PID who instead collects their data from simulated driving. The Dual Learning Model is only trained on datasets driven on highways meanwhile TFE-PID is trained on specific scenarios only. This thesis and MultiPath++ uses more general datasets with all scenarios included. This thesis is the only one using DMS data but the TFE-PID also uses driver behaviour such as brake pedal and steering torque. Dual Learning Model and MultiPath++ uses perception data, such as radar and environmental data, to make their prediction meanwhile TFE-PID and this thesis only focuses on the dynamics of the car and driver behavior. This makes it harder to predict because of the uncertainty from other vehicles.

3

Theory

This chapter provides the theoretical background and literature necessary to support the methodology and analysis presented in this thesis. It begins by introducing classical vehicle motion models and their underlying kinematic assumptions, which serve as important baselines for trajectory prediction. The chapter then reviews key concepts related to driver behavior, driver intent, and DMS, highlighting how human factors influence vehicle motion. Finally, relevant machine learning techniques for trajectory prediction are discussed, with a focus on transformer-based architectures and multimodal prediction. Together, these theoretical foundations establish the motivation for integrating driver-related information into short-horizon vehicle trajectory prediction.

3.1 Vehicle Dynamics and Traditional Motion Models

Object trajectory prediction is commonly formulated as a discrete-time state estimation problem as in Equation (3.1),

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k) + \mathbf{w}_k, \qquad \mathbf{w}_k \sim \mathcal{N}(\mathbf{0},\, \mathbf{Q}), \tag{3.1}$$

where $f(\cdot)$ encodes kinematic assumptions about the object's motion. The models presented in this section follow the standard formulations widely used in ground-vehicle tracking [11].

3.1.1 Discrete-Time formulation

The state-transition function in Equation (3.1) is expressed in discrete time because both vehicle sensors and trajectory-prediction models operate on sampled data rather than continuous signals. Automotive sensors typically provide CAN, IMU, DMS, and other measurements at fixed frequencies, meaning the prediction problem is inherently discrete. In addition, discrete-time formulations allow direct integration into recursive filters and learning-based architectures that process sequences of fixed-length time steps. Although the underlying vehicle motion is continuous, the discrete-time approximation is sufficiently accurate at common automotive sampling rates (e.g., 10–100 Hz) and forms the practical basis for both analytical and machine-learning prediction methods.

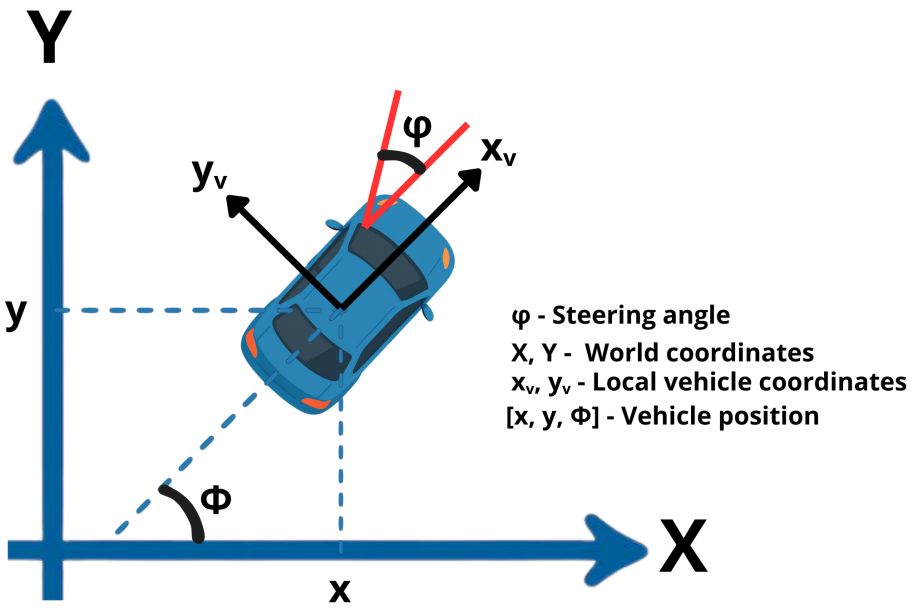


Figure 3.1: Image of a local coordinate system of the ego vehicle

3.1.2 Local Coordinate System

The local coordinate system used for vehicle motion prediction is a body-fixed frame attached to the car, as seen in Figure 3.1. Its origin is placed at the vehicle's center of gravity (CoG). The x-axis points forward in the driving direction, the y-axis points to the left side of the vehicle, and the z-axis points upward, perpendicular to the road plane. Because the frame moves and rotates with the car, velocities, accelerations, yaw rate, and short-horizon displacements are naturally expressed in this coordinate system. Most onboard sensors (e.g., radar, lidar, camera) are calibrated in this vehicle-fixed frame, which makes it the standard choice for motion prediction and tracking.

The relationship between the local frame (x_v, y_v) and the global map frame (X, Y) is defined through the vehicle's yaw angle ϕ. A point expressed in local coordinates can be mapped to global coordinates using the rotation matrix $R(\phi)$ and the vehicle's global pose (X_0, Y_0):

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} X_0 \\ Y_0 \end{bmatrix} + \underbrace{\begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}}_{R(\phi)} \begin{bmatrix} x \\ y \end{bmatrix}. \tag{3.2}$$

The inverse mapping (global to local) uses the transpose of the rotation matrix:

$$\begin{bmatrix} x \\ y \end{bmatrix} = R(\phi)^\top \left(\begin{bmatrix} X \\ Y \end{bmatrix} - \begin{bmatrix} X_0 \\ Y_0 \end{bmatrix} \right), \qquad R(\phi)^\top = R(-\phi). \tag{3.3}$$

Local velocities (v_x, v_y) relate to global velocity $(\dot{X}, \dot{Y})$ via the same rotation:

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \end{bmatrix} = R(\phi) \begin{bmatrix} v_x \\ v_y \end{bmatrix}, \qquad \begin{bmatrix} v_x \\ v_y \end{bmatrix} = R(\phi)^\top \begin{bmatrix} \dot{X} \\ \dot{Y} \end{bmatrix}. \tag{3.4}$$

When a global trajectory is required for evaluation or visualization, the predicted local points are mapped back to the global frame using Equation (3.2).

3.1.3 Classical Motion Models

Constant Velocity (CV) models assumes rectilinear motion at constant speed. With state $\mathbf{x} = [x,\, y,\, \dot{x},\, \dot{y}]^\top$, the transition is linear, shown in Equation (3.5).

$$\mathbf{x}_{k+1} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}_k + \mathbf{w}_k. \tag{3.5}$$

Any deviation from straight-line motion is absorbed by the process noise $\mathbf{w}_k$, making the model suitable only for short prediction horizons with low lateral dynamics. Because CV uses constant velocity it is not good when accelerating.

Constant Acceleration (CA) models augments CV with per-axis acceleration, $\mathbf{x} = [x,\, y,\, \dot{x},\, \dot{y},\, \ddot{x},\, \ddot{y}]^\top$, yielding Equation (3.6).

$$\mathbf{x}_{k+1} = \begin{bmatrix} 1 & 0 & \Delta t & 0 & \frac{1}{2}\Delta t^2 & 0 \\ 0 & 1 & 0 & \Delta t & 0 & \frac{1}{2}\Delta t^2 \\ 0 & 0 & 1 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 1 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}_k. \tag{3.6}$$

This discrete-time CA model uses acceleration which improves upon CV during a gas or braking maneuver, though it still assumes straight-line motion.

Constant Turn Rate and Velocity (CTRV) models, also denoted CTCV in the literature, introduces a heading angle θ and yaw rate ω to capture curvilinear motion. The state is $\mathbf{x} = [x,\, y,\, \theta,\, v,\, \omega]^\top$, and the nonlinear transition function is

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \begin{cases} \left[\frac{v}{\omega}\big(\sin(\theta+\omega\Delta t) - \sin\theta\big),\ \frac{v}{\omega}\big(\cos\theta - \cos(\theta+\omega\Delta t)\big),\ \omega\Delta t,\ 0,\ 0\right]^\top & |\omega| > \varepsilon, \\ [v\cos\theta\,\Delta t,\ v\sin\theta\,\Delta t,\ 0,\ 0,\ 0]^\top & |\omega| \le \varepsilon, \end{cases} \tag{3.7}$$

where the degenerate case $|\omega| \le \varepsilon$ recovers CV kinematics. The nonlinearity of Equation (3.7) necessitates a nonlinear filter such as the Unscented Kalman Filter (UKF).

Constant Turn Rate and Acceleration (CTRA) extends CTRV by adding a tangential acceleration a, giving $\mathbf{x} = [x,\, y,\, \theta,\, v,\, \omega,\, a]^\top$. For $|\omega| > \varepsilon$, the position updates are shown in Equation (3.8) and (3.9).

$$x_{k+1} = x_k + \frac{1}{\omega^2}\Big[(v_k\omega + a\omega\Delta t)\sin(\theta_k + \omega\Delta t) + a\cos(\theta_k + \omega\Delta t) - v_k\omega\sin\theta_k - a\cos\theta_k\Big], \tag{3.8}$$

$$y_{k+1} = y_k + \frac{1}{\omega^2}\Big[-(v_k\omega + a\omega\Delta t)\cos(\theta_k + \omega\Delta t) + a\sin(\theta_k + \omega\Delta t) + v_k\omega\cos\theta_k - a\sin\theta_k\Big], \tag{3.9}$$

The remaining states will then evolve as in Equation (3.10).

$$\theta_{k+1} = \theta_k + \omega\Delta t, \qquad v_{k+1} = v_k + a\Delta t, \qquad \omega_{k+1} = \omega_k, \qquad a_{k+1} = a_k. \tag{3.10}$$

CTRA captures both turning and longitudinal acceleration and is therefore the most expressive constant-parameter model in this family.

The CV, CA, CTRV, and CTRA models are widely used as baselines in vehicle-tracking and trajectory-prediction research because they provide simple yet physically interpretable approximations of typical on-road vehicle motion. Their state-transition equations are analytically derived from the kinematic properties of ground vehicles, making them computationally lightweight and well-suited for real-time applications. These models also form the foundation of many classical estimation frameworks, such as the Extended Kalman Filter and the Unscented Kalman Filter, which rely on explicit process models. By progressively increasing model complexity they span a spectrum of behaviour that is sufficient for comparison with more advanced data-driven approaches. For this reason, they serve as standard and reproducible baselines across most vehicle-prediction benchmarks.

3.1.4 Kinematic Bicycle Model

The Kinematic Bicycle Model (KBM) is a commonly used low-complexity vehicle model that captures the non-holonomic motion constraints of a wheeled vehicle while remaining computationally efficient [12]. It represents the vehicle as a single-track system by lumping the left and right wheels of each axle into a single front wheel and a single rear wheel. Unlike simple point-mass models, the KBM explicitly models the coupling between longitudinal and lateral motion through steering-induced curvature, making it well suited for trajectory prediction over short horizons.
The vehicle state is defined as

$$\mathbf{x} = [x,\ y,\ \theta,\ v,\ \kappa]^\top, \tag{3.11}$$

where (x, y) denotes the vehicle position in the horizontal plane, θ is the yaw angle, v is the longitudinal speed, and κ is the path curvature. Curvature is defined as the

inverse of the instantaneous turning radius and relates to the steering angle through the vehicle geometry.
Under the kinematic assumption of negligible lateral slip and pure rolling contact, the continuous-time equations of motion are

$$\dot{x} = v \cos\theta, \tag{3.12}$$

$$\dot{y} = v \sin\theta, \tag{3.13}$$

$$\dot{\theta} = v\kappa, \tag{3.14}$$

$$\dot{v} = a, \tag{3.15}$$

$$\dot{\kappa} = \delta\kappa, \tag{3.16}$$

where a is the longitudinal acceleration and $\delta\kappa$ denotes the curvature rate, which serves as a steering control input. This formulation decouples steering dynamics from explicit steering angles and vehicle geometry, enabling a compact and numerically stable representation that is well suited for learning-based control and prediction models.

For discrete-time prediction with sampling interval Δt, the KBM is commonly integrated using a semi-implicit Euler scheme. Given the state at time step t, the discrete-time update equations are

$$\kappa_{t+1} = \text{clamp}\left(\kappa_t + \delta\kappa_t \Delta t\right), \tag{3.17}$$

$$v_{t+1} = \text{clamp}\left(v_t + a_t \Delta t\right), \tag{3.18}$$

$$\theta_{t+1} = \theta_t + v_{t+1}\kappa_{t+1}\Delta t, \tag{3.19}$$

$$x_{t+1} = x_t + v_{t+1}\cos\theta_{t+1}\Delta t, \tag{3.20}$$

$$y_{t+1} = y_t + v_{t+1}\sin\theta_{t+1}\Delta t. \tag{3.21}$$

The use of semi-implicit integration improves numerical stability by updating velocity and curvature before propagating orientation and position. The clamp operator enforces physically feasible bounds on speed and curvature, preventing unrealistic predictions under aggressive control inputs.

3.1.5 Limitations of Traditional Motion Models

Traditional motion models rely on simplified kinematic assumptions that make them well-suited for smooth, low-dynamic driving scenarios but cause their accuracy to degrade under more demanding conditions. Models such as CV and CA assume rectilinear motion with either constant velocity or constant acceleration, which restricts their ability to represent curved trajectories or transient steering behaviour. CTRV and CTRA extend these formulations by introducing yaw-rate dynamics, yet they still assume that yaw rate, tangential velocity, and tangential acceleration remain

constant over the prediction interval. Such assumptions rarely hold over longer prediction horizons or during rapid driver inputs, evasive manoeuvres, and situations involving strong tyre forces, where vehicle motion becomes highly nonlinear [11]. Moreover, many classical models rely on additional simplifying assumptions, including linear tyre behaviour, small-angle approximations, and fixed vehicle parameters, all of which break down under high slip conditions, aggressive steering, or varying road surfaces [12, 13]. As a consequence, the predictive performance of these traditional motion models deteriorates rapidly as the prediction horizon increases, and they remain reliable primarily for short-term forecasting under mild and stable driving behaviour [14, 15].

Beyond the dynamical assumptions themselves, traditional motion models typically extrapolate motion based solely on the most recent host-vehicle state, making them effectively memoryless and unable to exploit longer temporal patterns that convey trends in motion or driver intent [16]. They also rely exclusively on vehicle-centric kinematic variables and do not incorporate perception data from radar, camera, or lidar. Consequently, they cannot account for interactions with surrounding vehicles, road geometry, or environmental context [16]. Likewise, they ignore behavioural cues from the driver such as gaze direction, head pose, distraction, and characteristic steering patterns, even though such cues are known to be predictive of imminent manoeuvres [17, 18].

3.2 Driver Behavior and Driver Monitoring Systems

Driving is a complex cognitive and motoric task that involves continuous interaction between the driver, the vehicle and the surrounding environment. The driver must perceive relevant information from the environment, interpret the situation, and execute appropriate control actions such as steering, acceleration and braking. This process is closely related to the concept of situational awareness, which describes the driver's ability to perceive environmental elements, comprehend their meaning and anticipate future states [19]. Situational awareness directly influences driver behaviour and vehicle motion, as drivers continuously adjust their control inputs based on perceived risks, road geometry and intended maneuvers.
Driver behaviour is naturally variable and influenced by factors such as experience, attention and intent. As a result, vehicle trajectory cannot always be predicted solely on the current vehicle state. Incorporating information about driver state and behavior may therefore improve the prediction of future vehicle motion.

3.2.1 Driver Intent and Its Influence on Vehicle Motion

Driver intent is not directly observable but can be concluded through behavioral indicators, particularly visual attention and head movements. Visual attention plays

a critical role in driving, as drivers consistently shift their gaze between different regions of interest, such as the road ahead, mirrors and intersections. These gaze patterns reflects the drivers focus of attention and provide insight into their intended future actions. Research has shown that drivers typically look toward relevant areas of the environment before initiating corresponding control action, e.g. directing their gaze towards an upcoming turn or hazard before initiating steering or braking actions [17].

Driver gaze can also be categorized into predefined gaze regions (see Figure 3.2), which reflect the drivers allocation of visual attention within the driver environment [21]. These gaze targets provide relevant information about driver attention and interaction with both the road and in-vehicle systems.

Figure 3.2: Example layout of different gaze targets used in this work, inspired by prior zone-based gaze representations [20].

These relationships between driver intent and vehicle motion has also been highlighted in the broader motion prediction literature. It has been shown that future vehicle trajectories are fundamentally influenced by driver decision-making, and that accurate motion prediction requires modeling the behavioral component of driving in addition to vehicle dynamics [22]. Since multiple future trajectories may be possible given the same immediate vehicle state, knowledge of driver intent can help reduce uncertainty in trajectory prediction.

3.2.2 Driver Monitoring Systems

Driver Monitoring Systems (DMS) are in-cabin sensing systems designed to monitor driver behaviour and state while driving. These systems typically use cameras in combination with computer vision algorithms to track various observable driving

features such as head pose, gaze direction, eye activity and hand positions [23]. These features provide measurable indicators of driver attention, behaviour and interaction with the vehicle. DMS have become increasingly important in modern vehicles and are now required by regulation in the European Union (EU). Under the EU General Safety 2019/2144, systems for monitoring driver attention and detecting driver drowsiness are required in new vehicle models and all new vehicles sold from 2024 [24]. Because of these regulations, DMS data is becoming widely available in modern vehicles, creating new opportunities for integrating driver state information into ADAS.

3.2.3 Limitations of DMS data

DMS signals are a valuable complement to conventional vehicle sensors for short-horizon motion prediction, but they also introduce practical and methodological limitations. One major limitation concerns the visibility of the driver's face. Dark sunglasses, reflective eyewear or low-light conditions can prevent the system from reliably detecting the eyes. This degrades gaze estimation and reduces the stability of gaze-related features [21, 23]. Head-pose tracking is also affected because most DMS algorithms depend on facial landmarks such as the eyes, nose, mouth and ears. If the driver wears a hood or a mask, these landmarks may be partially occluded, which increases uncertainty in the estimated head movements [23]. The camera is typically mounted near the steering wheel to obtain a clear view of the driver. This placement increases the likelihood that the camera becomes blocked by the driver's hands during normal steering, which can interrupt the measurements.

Another limitation arises from timing and sampling properties. DMS pipelines often operate at lower update rates than vehicle CAN signals. The computational steps required for gaze and head-pose estimation introduce additional latency. This creates a temporal mismatch with high-frequency vehicle state measurements at 50 Hz. As a result, behavioural cues from the DMS may not be perfectly aligned with the corresponding vehicle dynamics in real time [23].

Driver-specific differences create another challenge. Gaze and head-movement strategies vary widely across individuals. Some drivers rely almost entirely on subtle eye movements while others perform large head rotations even for small checks. In addition, not every gaze shift reflects an upcoming manoeuvre. Drivers may scan the environment, check mirrors out of habit or look toward non-driving stimuli inside or outside the cabin. These behaviours make it difficult to derive a consistent mapping from DMS signals to vehicle intent without additional contextual information. This variability reduces the ability of a single model to generalize across different drivers [17, 23].

Privacy and regulatory constraints also influence the use of DMS data. Raw in-cabin imagery is often restricted, which means that recordings may need to be deleted immediately or transformed into anonymized numerical features at the time of capture. In many jurisdictions, retaining identifiable video requires explicit consent from the

driver. These considerations limit the richness of the data that can be used for training and may affect which signals are available in production systems [23, 24].

3.3 Machine Learning for Trajectory Prediction

This section presents the machine learning concepts relevant to trajectory prediction in this thesis, with a focus on sequence modeling using transformer architectures. It first introduces the transformer and its core components, followed by key architectural variants such as DETR and BERT, and concludes with multimodal prediction methods for representing multiple possible future trajectories.

3.3.1 Transformers for Sequential Modeling

The transformer was introduced by Vaswani et al. in the paper Attention Is All You Need [25], which proposed a novel neural network architecture based entirely on self-attention mechanisms, eliminating the need for recurrence or convolution and achieving state-of-the-art results in machine translation. The network architecture is shown in Figure 3.3. The network was initially used for Natural Language Processing (NLP) where the self-attention mechanism where crucial [25]. The transformer quickly became popular for both NLP and computer vision using self-attention effectively [3]. The difference between a transformer and any other encoder-decoder structure is the positional encoding and self-attention. The positional encoding makes it possible to do future trajectory prediction of vehicles [26]. It is autoregressive, which means the model generates its output one step at a time: it produces the first token, then uses that token as input to generate the next one, continuing this process until the entire sequence has been produced.

3.3.2 Transformer Variants Relevant to this Work

Detection Transformers (DETR) are a transformer-based architecture that, instead of using autoregression to predict future timestamps, predicts all future outputs simultaneously [27]. This makes it well-suited for tasks where the input and output sequence lengths are known in advance.

A common use case for DETR is object detection in images. In this setting, the input is an image and the output consists of bounding boxes around detected objects. DETR assumes a fixed maximum number of objects and outputs both class probabilities and bounding-box coordinates for each predicted object. An example of this network is showcased in Figure 3.4.

Its architecture typically includes a convolutional neural network (CNN) backbone that extracts feature maps from the input image. These features are then passed into a transformer encoder–decoder structure. Finally, the decoder produces object probabilities and bounding-box coordinates, which are mapped back onto the original image.

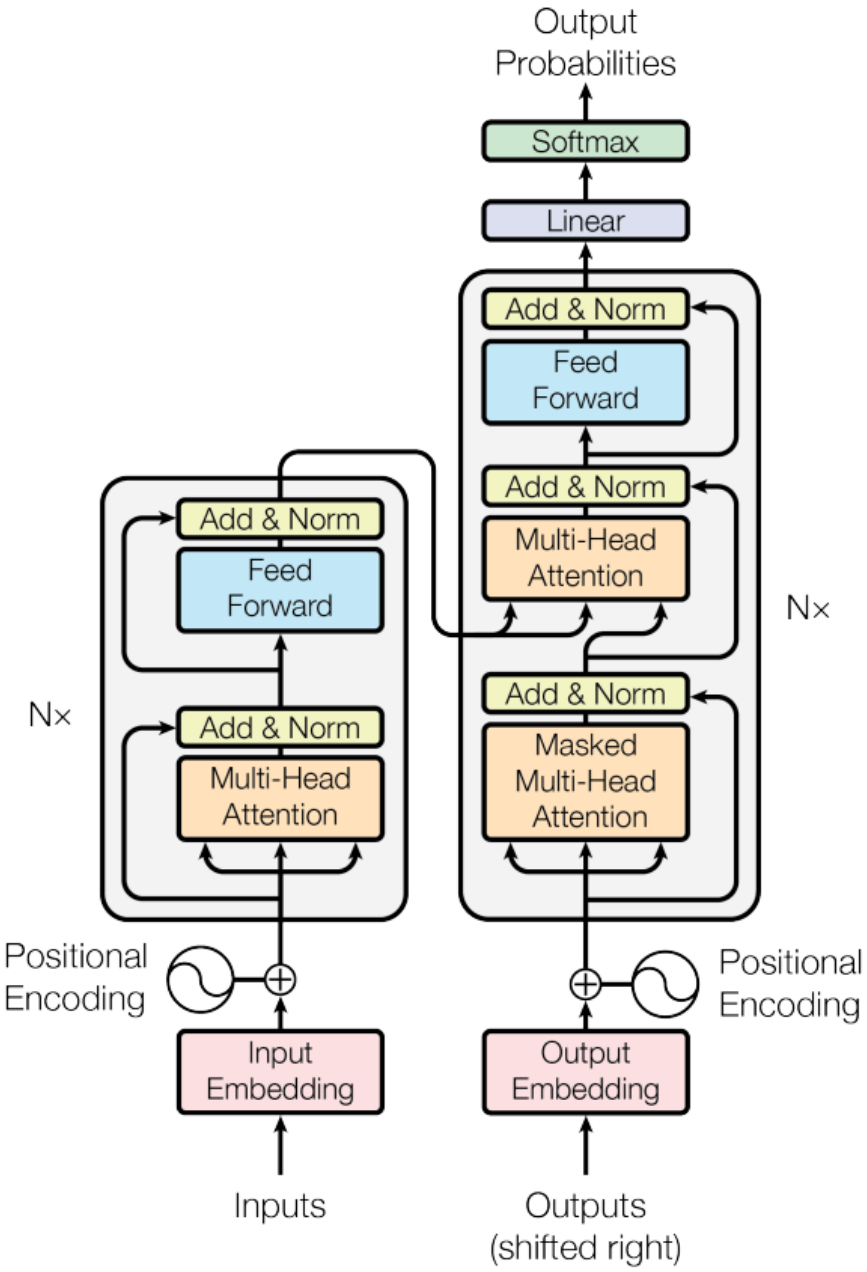


Figure 3.3: Transformer Model Architecture [25]

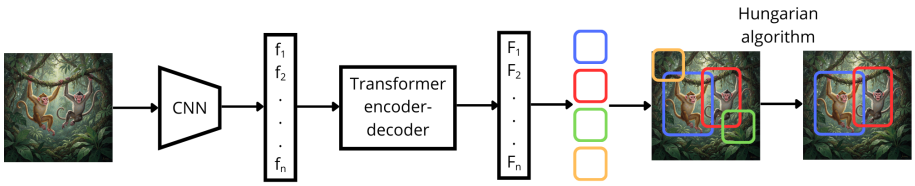


Figure 3.4: DETR flowchart example

Bidirectional Encoder Representations from Transformers (BERT) are a widely used transformer-based architecture [28]. In contrast to unidirectional models, BERT's encoder processes the input sequence in both the forward direction ($a_1 \rightarrow a_n$) and the backward direction ($a_n \rightarrow a_1$). This bidirectional processing enables any intermediate position k in the sequence to incorporate contextual information from both preceding elements ($a_1 \leftarrow a_k$) and subsequent elements ($a_k \rightarrow a_n$). This network is shown in Figure 3.5 where the arrows are pointing both left and right from each neuron.

Such a mechanism is particularly advantageous in natural language processing, where the meaning of a word often depends on both its left and right context within a sentence. By leveraging information from the entire surrounding context, BERT develops richer semantic representations.

This property is also beneficial for trajectory prediction tasks, as the model can capture how a given state arises from past behavior while simultaneously considering its implications for future states. Consequently, the bidirectional representation allows the network to learn deeper dependencies and more nuanced relationships among the features of a sequence.

A **Dual-stream Encoder** architecture is an architectural design in which two independent encoder branches process two distinct input modalities in parallel. Each stream employs a modality-specific encoder that is tailored to the temporal and statistical properties of its input features. Depending on the model design, these encoders may be implemented using transformer-based architectures with self-attention, or using CNNs that operate along the temporal dimension.

In transformer-based realizations, each stream consists of an encoder stack composed of multi-head self-attention layers, feed-forward networks, positional encodings, and normalization layers. The input modality is first embedded into a sequence of tokens compatible with transformer processing, after which positional encodings are added to preserve temporal order. Self-attention then models long-range temporal dependencies by weighting contributions from different time steps within the sequence.

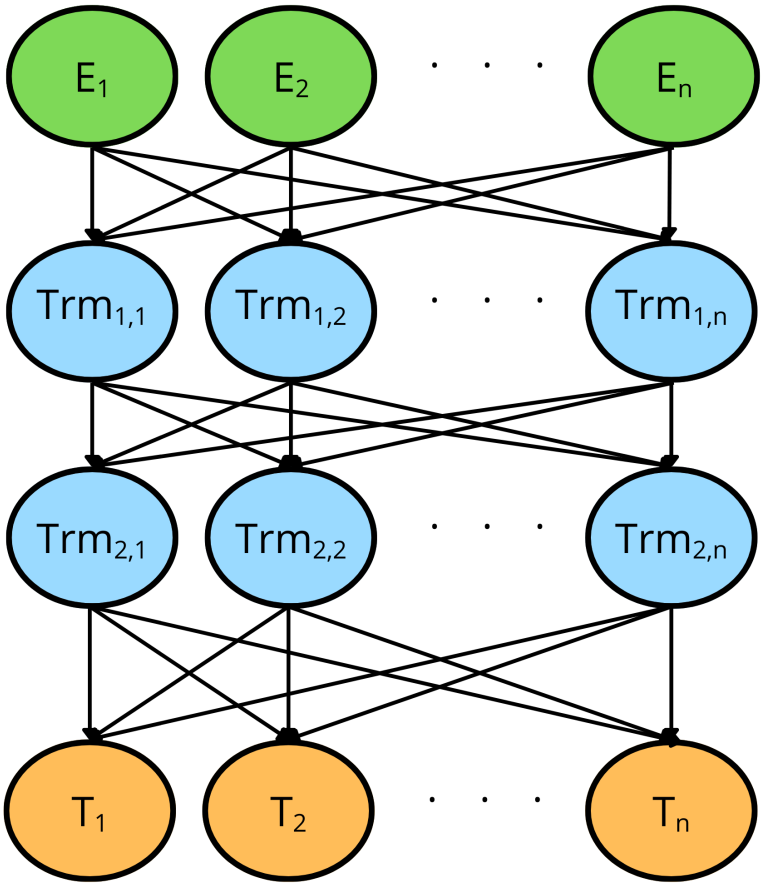


Figure 3.5: Flowchart example using bidirection

Alternatively, CNN-based encoders process each input modality using stacked temporal convolutional layers. These layers apply one-dimensional convolutions along the time axis to capture local temporal patterns and short-range dependencies. By increasing the receptive field through stacking and dilation, such encoders can model increasingly long temporal contexts while maintaining computational efficiency and strong inductive bias for regularly sampled time-series data.

Because the two streams do not interact during encoding, each learns an internal representation that is specialized to the characteristics of its respective input domain. After the independent encoding, the resulting feature sequences are merged in a downstream fusion module, where cross-modal interactions are modeled. Fusion can involve concatenation, cross-attention, or other integration techniques depending on the specific model design. The fused representation is then provided to the prediction head for the downstream task [9].

3.3.3 Multimodal Prediction

In trajectory prediction, multimodality refers to the fact that multiple distinct future outcomes may be plausible given the same past observations. This ambiguity arises naturally in dynamic systems, where different actions or behaviors can lead to qualitatively different future trajectories, even when the initial state is identical. As a result, predicting a single deterministic outcome may fail to capture the inherent uncertainty present in real-world scenarios.

Multimodal prediction methods address this issue by representing multiple possible futures instead of committing to a single prediction. Each mode typically corresponds to a different plausible evolution of the system, allowing the model to reflect uncertainty and variability in future behavior. This representation is particularly important in human-in-the-loop systems, where future motion depends not only on physical dynamics but also on latent factors such as intent, attention, and decision-making processes. By explicitly accounting for multiple outcomes, multimodal approaches provide a more expressive and realistic framework for sequence prediction under uncertainty [29].

3.4 Short-Horizon Trajectory Prediction in ADAS

Short-horizon trajectory prediction refers to forecasting the future motion of the host vehicle over time scales typically ranging from about 0.5 to a few seconds, an interval that is central to safety-critical decisions and control in ADAS [30, 31]. Although this time interval may seem brief, it is critical for a number of ADAS that rely on fast, reliable estimates of where the vehicle will be in the immediate future. Improved short-term predictive accuracy can change the inputs provided to downstream safety functions, which in turn may affect their behavior.

For ADAS systems such as Automatic Emergency Braking (AEB) and Automatic Emergency Steering (AES), short-term prediction informs whether an intervention

is necessary and which maneuver is feasible. In AEB, precise estimates of imminent longitudinal motion support time-to-collision and stopping-distance logic [32]. Inaccurate trajectory predictions can lead either to unnecessary activations or delayed braking.
AES generally imposes stricter requirements than AEB because steering-based avoidance involves faster and more complex vehicle dynamics. A steering-based avoidance maneuver is only safe if the system can predict the vehicle's combined lateral and longitudinal motion with high accuracy. Since lateral dynamics evolve quickly and non-linearly during evasive maneuvers, even small prediction errors could affect whether a steering intervention is deemed safe or even possible. Because such interventions typically unfold over fractions of a second, accurate short-term prediction become essential for determining both the feasibility and the timing of a steering response.

3.5 Evaluation Metrics for Trajectory Prediction

Two of the evaluation metrics used in trajectory prediction are Average Displacement Error (ADE) and Final Displacement Error (FDE). ADE is the average positional error for the prediction point $\hat{\mathbf{p}}_t$ compared to the ground truth for each future time t, meanwhile FDE is the positional error for the final prediction point $\hat{\mathbf{p}}_T$. For a predicted trajectory $\{\hat{\mathbf{p}}_t\}_{t=1}^{T}$ and ground truth $\{\mathbf{p}_t\}_{t=1}^{T}$,

$$\text{ADE} = \frac{1}{T}\sum_{t=1}^{T} \|\hat{\mathbf{p}}_t - \mathbf{p}_t\|_2 \,, \qquad \text{FDE} = \|\hat{\mathbf{p}}_T - \mathbf{p}_T\|_2 \,.$$

ADE captures the average deviation over the entire prediction horizon and is therefore sensitive to accumulated trajectory drift. In contrast, FDE focuses exclusively on the error at the final prediction step and emphasizes endpoint accuracy. As a result, models with similar ADE values may exhibit substantially different FDE performance, particularly in scenarios involving turning maneuvers or late trajectory corrections.

In short-horizon trajectory prediction, endpoint accuracy is particularly important because downstream ADAS functions such as AEB and AES rely on precise estimates of the vehicle's position a few seconds ahead. To further assess endpoint reliability, Miss Rate (MR) is commonly reported. In this paper, MR is defined as the fraction of predictions whose final position lies outside a predefined distance threshold δ from the ground truth final position. A prediction is therefore considered a miss if

$$\|\hat{\mathbf{p}}_T - \mathbf{p}_T\|_2 > \delta,$$

and the MR is computed as the proportion of such missed predictions over the evaluation set. MR thereby provides a complementary, threshold-based perspective to FDE by indicating whether the predicted endpoint is sufficiently accurate to be considered usable for downstream decision-making.

4

Learning-Based Multi-Modal Vehicle Trajectory Prediction

This section presents the complete methodology used in this thesis, covering every step from data acquisition to model evaluation. It begins with a formal problem definition followed by the collection and description of the dataset, preprocessing pipeline used to clean, synchronize, and construct the final feature set. The model architecture is then detailed, including the dual-stream encoder design, gaze-target embedding, and decoding process. Finally, the evaluation procedure is outlined, describing the metrics, baseline motion models, and ablation study used to assess model performance.

4.1 Formal Problem Definition

The objective of this thesis is to predict the short-horizon future trajectory of a human-driven host vehicle by leveraging both recent vehicle motion and driver-related behavioral signals. This section provides a formal, model-agnostic definition of the trajectory prediction problem addressed in this work.

We consider a discrete-time setting with a fixed sampling interval Δt, where all signals are represented as sequences indexed by time. Let t_i denote the prediction origin time. The host vehicle state is observed over a finite history window of length S_h steps, forming the sequence

$$\mathbf{x}_{1:S_h}^{(h)} = [\mathbf{x}_{t_i-S_h+1}^{(h)}, \ldots, \mathbf{x}_{t_i}^{(h)}], \tag{4.1}$$

where $\mathbf{x}_t^{(h)} \in \mathbb{R}^6$ represents the host vehicle motion state at time t, including longitudinal speed, yaw rate, and accelerations, expressed in the vehicle-fixed local coordinate system described in Section 3.1.2.
In addition to vehicle motion, driver-related signals are observed over a longer temporal window of length S_d steps,

$$\mathbf{x}_{1:S_d}^{(d)} = [\mathbf{x}_{t_i-S_d+1}^{(d)}, \ldots, \mathbf{x}_{t_i}^{(d)}], \tag{4.2}$$

where $\mathbf{x}_t^{(d)} \in \mathbb{R}^{D_d}$ contains actuator inputs (steering, brake, accelerator), discrete turn-indicator state, and driver monitoring system (DMS) features such as gaze direction, head pose, and gaze-target information.

The prediction task is to estimate the future planar trajectory of the host vehicle over a fixed horizon of S_{out} discrete timesteps,

$$\mathbf{P}_{t_i+1:t_i+S_{\text{out}}} = [\mathbf{p}_{t_i+1}, \ldots, \mathbf{p}_{t_i+S_{\text{out}}}], \tag{4.3}$$

where each $\mathbf{p}_t = (x_t, y_t) \in \mathbb{R}^2$ denotes the vehicle's future position in the local coordinate frame, relative to the vehicle pose at t_i. In this work, $S_{\text{out}} = 8$, corresponding to a prediction horizon of 3.5 seconds. The reason we use 3.5 seconds prediction horizon is explained in Section 4.4.
Formally, the trajectory prediction problem is defined as learning a function

$$f : \left(\mathbf{x}^{(h)}_{1:S_h}, \mathbf{x}^{(d)}_{1:S_d}\right) \longmapsto \mathbf{P}_{t_i+1:t_i+S_{\text{out}}}. \tag{4.4}$$

Because future vehicle motion is inherently uncertain and influenced by latent driver intent, multiple distinct future trajectories may be plausible even when conditioned on identical past observations. To account for this multimodality, the prediction function is extended to output a set of K candidate future trajectories together with associated confidence values,

$$f(\cdot) \longmapsto \left\{\left(\mathbf{P}^{(k)}_{t_i+1:t_i+S_{\text{out}}},\ p_k\right)\right\}_{k=1}^{K}, \tag{4.5}$$

where $\sum_{k=1}^{K} p_k = 1$ and each $\mathbf{P}^{(k)}$ represents a kinematically feasible future trajectory hypothesis.
This formulation is independent of the specific model architecture used to approximate f. The subsequent sections describe the dataset, preprocessing steps, and the proposed GA-DSTP architecture used to learn this mapping in practice.

4.2 Dataset

The dataset utilized in this thesis was collected using real-world driving data. Data acquisition took place in Gothenburg, Sweden, over the course of one week during the winter. Four drivers conducted approximately 17 hours of total driving during this period. The DMS camera was placed at the instrument cluster behind the steering wheel. The raw CAN and DMS data was then parsed, processed and converted into floating-point values stored in CSV format.

4.2.1 Available Signals and Feature Groups

Each CSV file contains 20 seconds of driving data, corresponding to 1,000 time steps at the 50 Hz sampling rate. The variables included in the dataset are described in Table 4.1. The vehicle position is represented in a local coordinate system, explained in Section 3.1.2, in which the initial ground-truth position is fixed at (x,y)=(0,0). The ground truth consists of nine future (x,y) positions, representing the vehicle's trajectory (relative to the local coordinate system) over the subsequent 3.5 seconds. The yaw angle is fixed at zero so that the longitudinal axis of the vehicle aligns with the x-axis (pointing forward), and the lateral axis aligns with the y-axis (pointing to the left). The generation of ground truth is described in Section 4.2.2.

Table 4.1: Dataset variables grouped by host vehicle state and driver behaviour.

Variable(s)	Description
Host Vehicle State	
`time`	Timestamp (s)
`host_speed`	Vehicle speed (m/s)
`host_yawrate`	Yaw rate (rad/s)
`host_lat_accel`	Lateral acceleration (m/s^2)
`host_long_accel`	Longitudinal acceleration (m/s^2)
Driver Behaviour (Gaze + Actuator signals)	
`steering_torque`	Steering torque (N m)
`acc_pedal_pos`	Accelerator pedal position (%)
`brake_pedal_torque`	Brake pedal torque (N m)
`turn_indicator_pos`	Turn indicator state (Enums)
`dms_pos_x,y,z`	Gaze origin position (m)
`dms_dir_x,y,z`	Gaze direction unit vector
`gaze_target1, gaze_target2`	Gaze target class IDs (Enums)
`gaze_target1_qual`, `gaze_target2_qual`	Gaze target confidence values ([0,1])
Target (Ground Truth)	
`gt_long_pos_1..9`	Longitudinal ground-truth vehicle positions (m)
`gt_lat_pos_1..9`	Lateral ground-truth vehicle positions (m)

The variables steering_torque and brake_pedal_torque represent the force applied by the driver to the steering wheel and brake pedal, respectively. The variables gaze_target1 and gaze_target2 describe the regions of the driver's field of view at which the driver is looking, together with associated gaze-quality values in the range [0,1].

4.2.2 Ground truth generation

Since no external positioning system was available during data collection, ground truth future trajectories were generated using vehicle-internal motion signals at the origin t_i, with the longitudinal axis pointing forward and the lateral axis pointing to the left, with the vehicle located at the origin (0, 0) and initial heading $\psi_0 = 0$. The planar motion of the host vehicle is obtained by the kinematic equations

$$\dot{x}(t) = v(t)\cos\psi(t), \qquad \dot{y}(t) = v(t)\sin\psi(t), \qquad \dot{\psi}(t) = r(t), \tag{4.6}$$

where $v(t)$ is the longitudinal speed and $r(t)$ is the yaw rate, both obtained directly from the CAN bus signals. No lateral tyre slip is assumed.

Ground truth is evaluated at $M = 9$ discrete horizons,

$$\mathcal{T} = \{0,\ 0.4375,\ 0.875,\ \ldots,\ 3.5\}\ \mathrm{s}, \tag{4.7}$$

spaced 0.4375 s apart and covering a total prediction horizon of 3.5 s. For each prediction origin t_i and horizon $\Delta \in \mathcal{T}$, the integration window $[t_i,\ t_i + \Delta]$ is subdivided at the native logging timestamps. Over each sub-interval $[t_k,\ t_{k+1}]$ with $\delta t_k = t_{k+1} - t_k$, the heading is updated using a trapezoidal average of the yaw rate,

$$\psi_{k+1} = \psi_k + \frac{r(t_k) + r(t_{k+1})}{2}\,\delta t_k, \tag{4.8}$$

and the position is propagated using the trapezoidal rule,

$$x_{k+1} = x_k + \frac{\delta t_k}{2}\Big[v(t_k)\cos\psi_k + v(t_{k+1})\cos\psi_{k+1}\Big], \tag{4.9}$$

$$y_{k+1} = y_k + \frac{\delta t_k}{2}\Big[v(t_k)\sin\psi_k + v(t_{k+1})\sin\psi_{k+1}\Big]. \tag{4.10}$$

Values of $v(t)$ and $r(t)$ at intermediate time points are obtained by linear interpolation between logged samples. This gives a ground truth matrix $\mathbf{P}^{\mathrm{GT}} \in \mathbb{R}^{N\times M\times 2}$ for a recording of N time steps, where each entry contains the longitudinal and lateral displacement from the prediction origin.

Since the trajectory prediction model outputs positions relative to a forward-displaced reference point, a fixed longitudinal offset of $d_{\mathrm{ref}} = 2.3614\,\mathrm{m}$ is subtracted from the computed longitudinal positions to align the ground truth with the model's coordinate origin. A visual illustration of the resulting ground truth trajectory is shown in Figure 4.1.

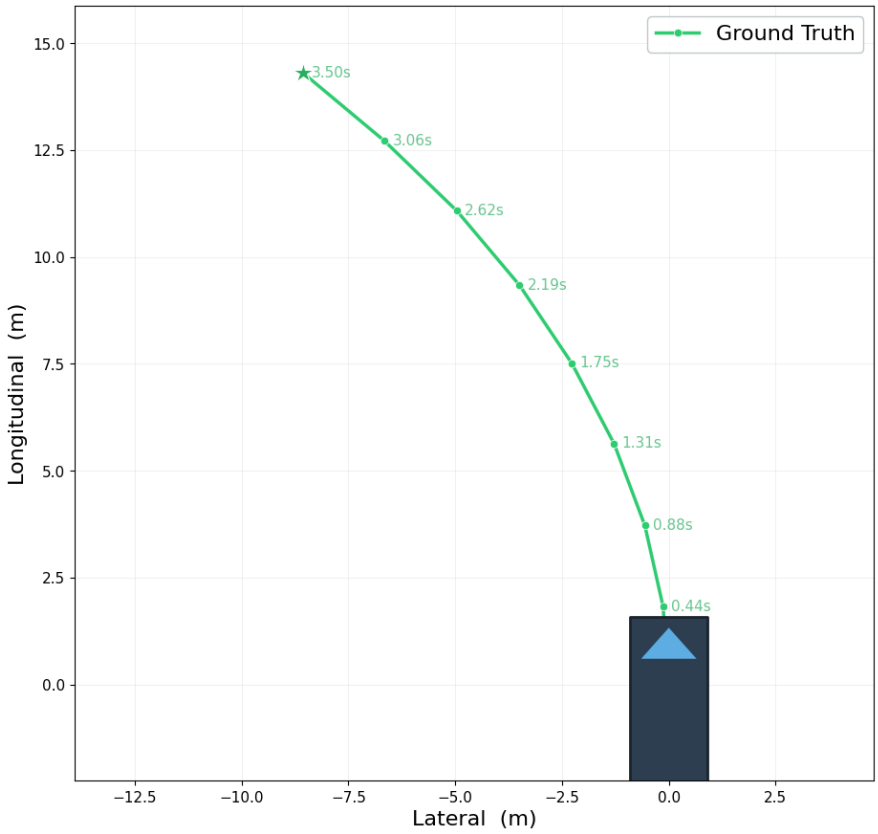


Figure 4.1: Example ground truth future trajectory relative to the prediction origin. Discrete points correspond to the evaluation horizons in $\mathcal{T}$.

4.2.3 Dataset Distribution

The dataset used for training the model was split into 80% training data and 20% validation data. A separate day of driving data was reserved as the test set to prevent data leakage. Table 4.2 below shows the distribution of maneuver categories, for each sample, within the dataset for the train, validation and test split.

The maneuver categories are based on the angular deviation of its future displacement vectors. How these are classified are shown below.

For a given sample, the longitudinal and lateral displacements $(d_{\text{long},k}, d_{\text{lat},k})$, where k is a future time from $\mathcal{T}$ except zero, over the prediction horizon are converted into angles

$$\theta_k = \arctan 2\Big(d_{\text{lat},k},\ \max(|d_{\text{long},k}|, 10^{-6})\Big)\,, \tag{4.11}$$

and the final-step angle $\theta_{\text{final}} = \theta_8$ is used for classification. A sample is labelled *standstill* if $|d_{\text{long},8}| < 0.5\,\text{m}$ and $\|(d_{\text{long},8}, d_{\text{lat},8})\| < 0.5\,\text{m}$.

	Train		Validation		Test	
Category	**N**	**%**	**N**	**%**	**N**	**%**
standstill	25626	7.1%	5290	5.8%	5187	12.80%
straight	209385	57.7%	51953	57.3%	22396	55.27%
light_left_turn	56600	15.6%	15633	17.2%	5799	14.31%
heavy_left_turn	9244	2.5%	1822	2.0%	497	1.23%
light_right_turn	51634	14.2%	13538	14.9%	5720	14.12%
heavy_right_turn	9933	2.7%	2237	2.5%	887	2.19%
unclassified	734	0.2%	184	0.2%	34	0.08%
Total	363156	100%	90657	100%	40520	100%

Table 4.2: Distribution of maneuver categories, for each sample, across training, validation, and test datasets.

For non-standstill samples, the final angle determines the turning direction:

$$\text{left turn:} \qquad \theta_{\text{final}} \geq 2^\circ, \tag{4.12}$$

$$\text{right turn:} \qquad \theta_{\text{final}} \leq -2^\circ, \tag{4.13}$$

$$\text{straight:} \qquad |\theta_{\text{final}}| \leq 2^\circ. \tag{4.14}$$

Left and right turns are further divided into light and heavy based on the angle magnitude,

$$\text{heavy turn if } |\theta_{\text{final}}| \geq 15^\circ, \qquad \text{light turn otherwise.} \tag{4.15}$$

The thresholds for the maneuver category classification are selected by looking at the video of the samples and from that manually choosing a suitable threshold. Before these signals can be used for model training, several preprocessing steps are required to synchronize and clean the data.

Table 4.3 shows the DMS coverage within the driver history window for each dataset split. The dataset partitions were arranged to make the proportions of complete, partial, and missing DMS coverage as even as possible across training, validation, and test data. The training and validation sets are closely matched, while the test set shows a slightly higher proportion of samples with complete DMS coverage. Overall, the distributions are sufficiently similar to limit bias from unequal DMS availability.

Table 4.3: DMS coverage per sample within the driver history window. Here, All DMS means all timesteps in the driver-history window contain valid DMS signals, Some DMS means only a subset of timesteps contain valid DMS signals, and No DMS means no valid DMS signals are present in that window.

Split	Category	Count	Percentage
Training	All DMS	209,472	57.7%
	Some DMS	121,667	33.5%
	No DMS	32,017	8.8%
Validation	All DMS	54,429	60.0%
	Some DMS	29,151	32.2%
	No DMS	7,077	7.8%
Test	All DMS	27,784	68.6%
	Some DMS	11,494	28.4%
	No DMS	1,242	3.1%

4.3 Data Preprocessing

First the raw logs (CAN and UDP files) were parsed using processing tools provided by the company to extract the signals needed for feature construction and ground truth generation.

4.3.1 Synchronization and Resampling

Since the signals originate from different sources and channels on the CAN BUS, they are recorded with different sampling frequencies and time stamps. Therefore a resampling was necessary to make them synchronized for correct construction of time series input sequences for the trajectory prediction model. This synchronization was performed using nearest-neighbor interpolation, where for each time stamp in the master time vector, the temporally closest available sample was selected.

4.3.2 Filtering and Noise Reduction

Certain CAN signals tend to show a lot of noise which required additional filtering before being used as input features. To reduce this noise while still preserving important signal dynamics, a second order Butterworth filter was used. The Butterworth filter was chosen as it provides a smooth passband response and behaves stably when applied using zero-phase forward-backward filtering, making it suitable for noise reduction without affecting the relative timing of the signals. An nth-order Butterworth filter has the magnitude response

$$|H(j\omega)|^2 = \frac{1}{1 + \left(\frac{\omega}{\omega_c}\right)^{2n}}, \tag{4.16}$$

where n denotes the filter order and ω_c is the cutoff frequency. The filter was designed using the MATLAB `butter()` function with a sampling frequency

$$f_s = 50 \text{ Hz} \tag{4.17}$$

and an empirically selected cutoff frequency

$$f_c = 4 \text{ Hz}. \tag{4.18}$$

In MATLAB, the cutoff frequency must be normalized by the Nyquist frequency, which is half the sampling frequency. The normalized cutoff frequency is therefore given by

$$W_n = \frac{f_c}{f_s/2}. \tag{4.19}$$

The `butter()` function returns the numerator and denominator coefficients of the digital filter transfer function [33]. The filtering operation in discrete time can be expressed as

$$y[k] = \sum_{i=0}^{M} b_i x[k-i] - \sum_{i=1}^{N} a_i y[k-i], \tag{4.20}$$

where $x[k]$ represents the input signal, $y[k]$ the filtered output signal, and a_i and b_i are the filter coefficients obtained from the Butterworth filter design. To avoid phase distortion and temporal delay, the filter was applied using forward-backward filtering via the `filtfilt()` function. This method applies the filter once in the forward direction and once in the reverse direction, resulting in zero-phase distortion [34]. This makes it so the filtered signals preserve the original timing of the signal dynamics while removing high-frequency noise.

4.3.3 Feature Cleaning and Validity Handling

After synchronization and filtering, additional handling is required to ensure that the remaining signals form a consistent and usable input sequence. Missing values introduced during resampling are filled using linear interpolation so that all input variables remain time-aligned across the observation window. For categorical gaze-target readings, entries marked as invalid based on their confidence values are replaced with the designated padding index used throughout the preprocessing pipeline.

When only one of the gaze-target entries at a time step is valid, the invalid one is replaced with its padding representation, allowing feature construction to proceed without discarding the entire time step. Continuous DMS signals that cannot be interpolated reliably, such as directional vectors with clear discontinuities, are instead set to zero for the affected samples to avoid introducing artificial trends. These steps ensure that the feature set maintains consistent dimensionality and avoids propagating incomplete or unreliable measurements into the model.

4.3.4 Final Feature Construction

Once cleaning is completed, all processed variables are assembled into the two input streams used by the model. The host-vehicle state stream contains the six continuous signals related to vehicle motion, while the second stream groups actuator inputs, DMS positional and directional features, turn-indicator encoding, and the fused gaze-target representation. Each time step is represented by one feature vector per stream, and these sequences are mapped into their respective embedding dimensions through linear projection layers. The outcome of this stage is a pair of synchronized, fixed-length embedded sequences that constitute the final model input. They provide a compact representation of both vehicle dynamics and driver-related behaviour, prepared for processing in the dual-stream encoder.

4.4 Model Architecture

This thesis propose the **Gaze Aware - Dual Stream Trajectory Prediction (GA-DSTP)** model which employs a transformer-style decoder with bidirectional attention and non-autoregressive prediction, similar to a DETR-model, combined with CNN encoders for temporal feature extraction. Bidirectional attention is applied in the decoder stage, while temporal feature extraction in the encoder is handled by the CNN layers. The flowchart of the model architecture is shown in Figure 4.2. The GA-DSTP model is implemented in the PyTorch deep-learning framework, which provides efficient tensor operations and flexible modules for transformer-based architectures.

We use 1 second of host vehicle history because prior work has shown that vehicle trajectory prediction performance saturates when using approximately 0.5 to 1 seconds of past data [6, 10]. Shorter windows fail to capture meaningful steering and lateral motion patterns, while longer histories provide little additional benefit and may introduce noise and overfitting. Dahl et al. demonstrate empirically that a historic depth in this range is sufficient for accurate lane-departure prediction and argue that recent driver behaviour dominates future motion, motivating the use of a limited, recent time window rather than long continuous histories [35]. A longer temporal window (3 s) is used for driver behaviour signals to account for human perception, decision and action latency and to capture behavioural trends rather than instantaneous responses. Human-factors research shows that driver reactions, particularly steering responses, unfold over a finite time interval that typically exceeds 1 s under realistic driving conditions [36]. In addition, recent driver intention and steering prediction studies explicitly model driver behaviour as a temporally evolving process and rely on multi-second input windows to capture consistent trends in steering actions, rather than isolated instantaneous measurements [37].

The prediction horizon is set to 3.5 seconds, which lies at the upper end of what is commonly considered short-horizon trajectory prediction. While many ADAS functions, such as AEB and AES, primarily rely on predictions within approximately two

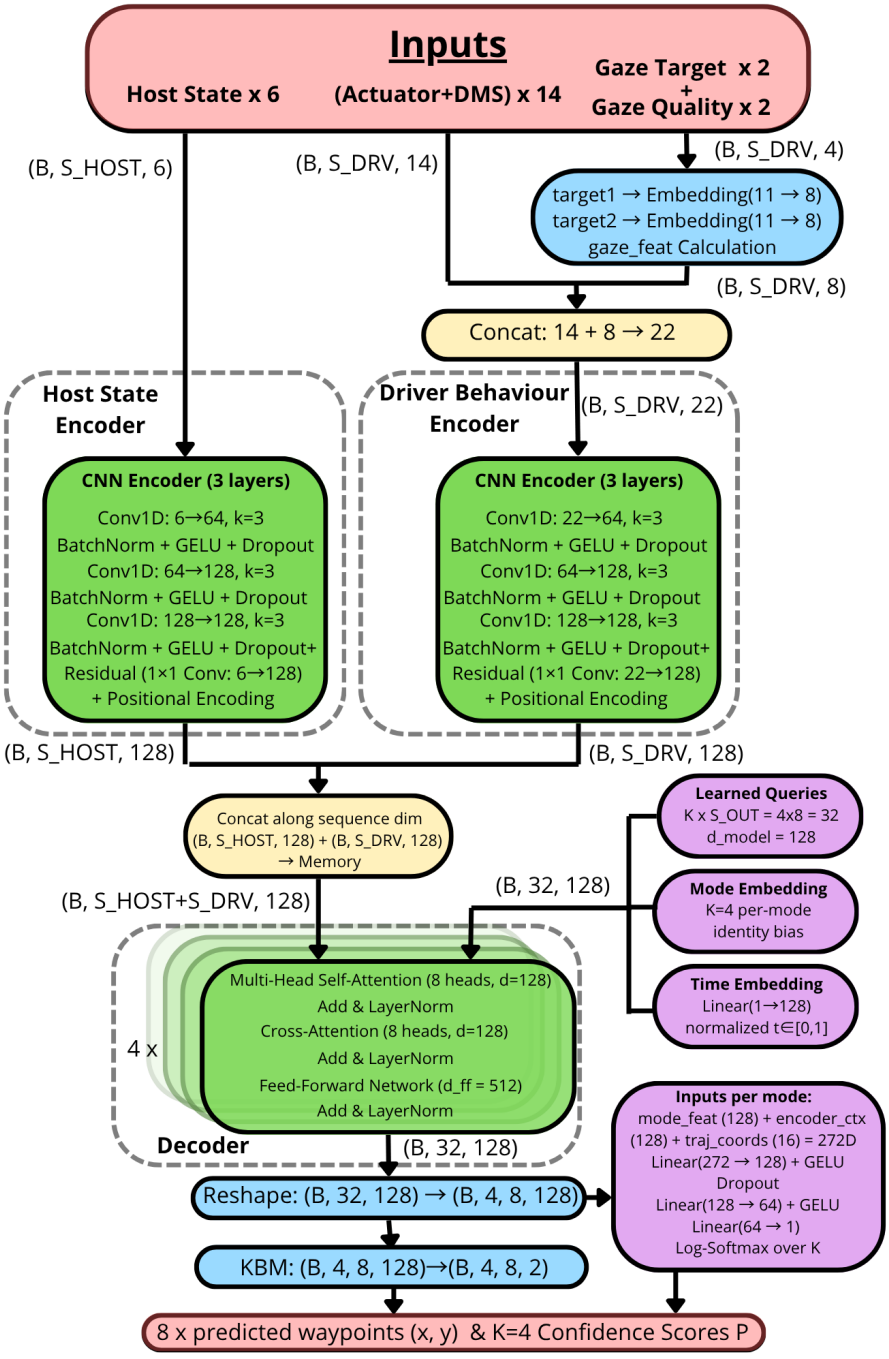


Figure 4.2: Architecture of the GA-DSTP model

seconds, extending the horizon provides a more challenging prediction setting. This design choice allows the evaluation of model robustness under increased uncertainty, where small deviations can accumulate and classical motion assumptions begin to degrade. By selecting a 3.5-second horizon, the objective is not to mirror conservative ADAS operating limits, but rather to assess whether driver monitoring data can contribute to improved prediction quality when future vehicle motion becomes less certain. This horizon therefore serves as a stress test for the proposed model, enabling a clearer analysis of the potential benefits of incorporating driver-related information beyond near-term reactive behavior.

To summarize, the network operates on a 3-second sequence of input signals from driver behavior data and 1-second sequence of input signals from vehicle host-state data. The output trajectory is 3.5-seconds in the future.

4.4.1 Dual-Stream Encoder

The GA-DSTP model processes two separate input modalities through a dual-stream encoder architecture, where each stream uses a dedicated CNN-based temporal encoder. Each CNN encoder operates independently on its respective feature sequence. This separation allows each modality to be contextualized without mixing information prematurely. This also enables an ablation study comparing the GA-DSTP model with the Dual Stream Trajectory Prediction (DSTP) model, yielding the architecture shown in Figure 4.3, which excludes DMS inputs. The outputs of the two encoder streams are later combined in the fusion step described in the following section.

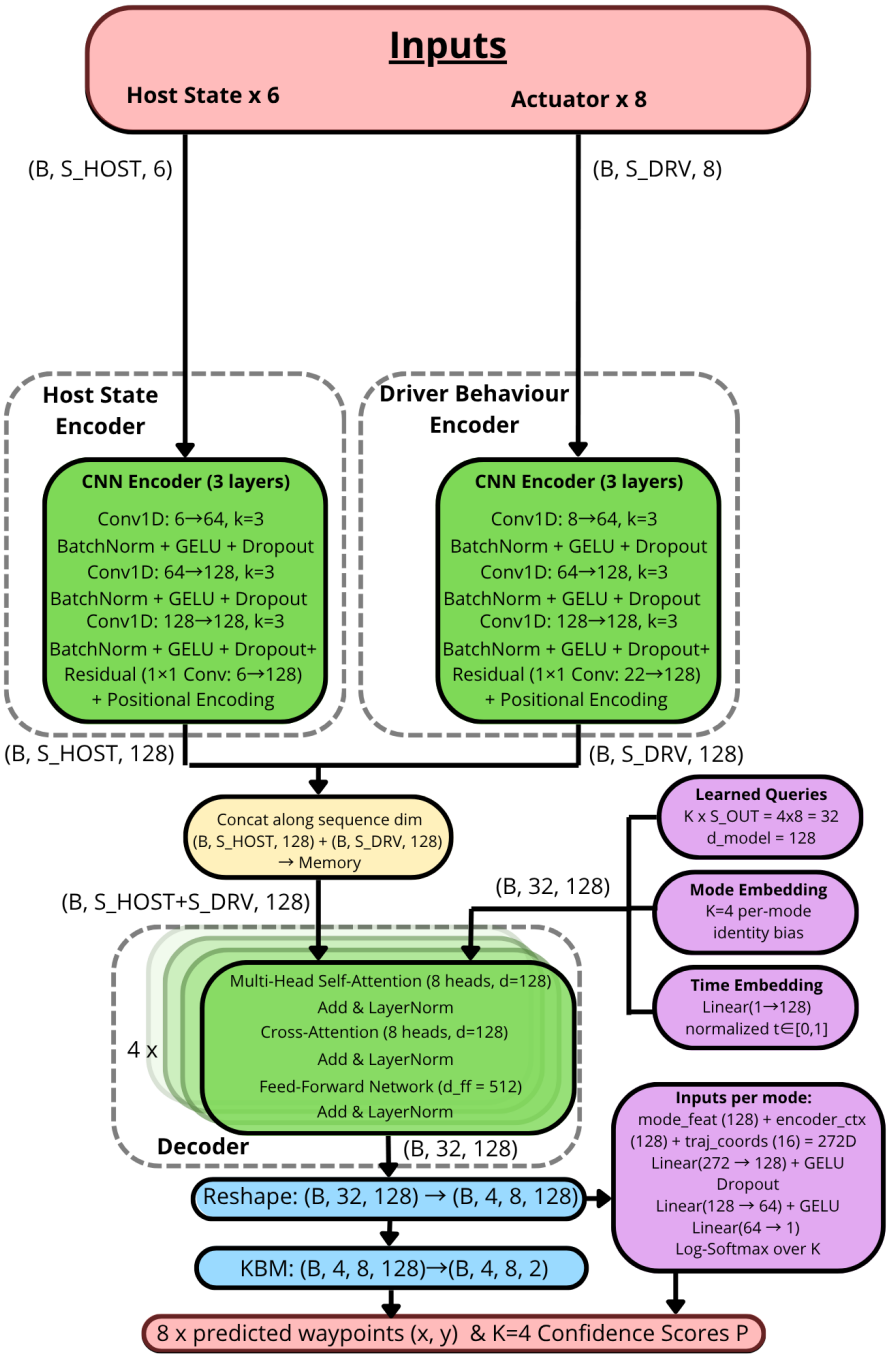


Figure 4.3: Architecture of the DSTP model, which excludes DMS data

Although transformer encoders were discussed in theory, CNN-based temporal encoders where chosen due to the short, fixed-length input windows, the modest dataset size, and the noisy, driver-specific nature of DMS signals. In such settings, transformer encoders can introduce excessive capacity and overfit, a phenomenon observed in small-data regimes where transformers underperform CNNs [38]. CNNs offer a stronger inductive bias for smooth time-series and serve as an implicit regularizer, while attention is instead reserved for the decoder where multimodal interaction and trajectory hypothesis selection are critical.

4.4.2 Fusion Layer and Decoder

After the two encoder streams have produced their modality-specific hidden representations, they are combined into a single memory sequence for decoding. This is done by concatenating the encoded host-state and driver-behaviour representations along the temporal dimension, forming a unified sequence that preserves all contextual information from both modalities. The resulting memory is passed directly to the decoder without additional temporal processing. The architecture of the GA-DSTP decoder is shown in Figure 4.4.

The decoder operates on this shared memory together with a fixed set of learned queries. These queries are designed to represent multiple candidate future motion hypotheses, enabling multi-modal trajectory prediction within a single forward pass. Through self-attention among queries and cross-attention over the fused memory, the decoder jointly reasons about alternative future behaviors conditioned on the same observed history.

The decoder outputs a set of latent vectors which are mapped by K separate per-mode control heads to 2-D control signals, curvature rate ($\delta\kappa$) and longitudinal acceleration (a) at each future timestep. These controls are bounded via tanh activation to physically feasible ranges and then integrated forward through a differentiable KBM to produce (x, y) waypoint trajectories. All future positions are generated simultaneously, enabling efficient and stable non-autoregressive trajectory prediction, while the KBM integration guarantees that every predicted trajectory satisfies non-holonomic vehicle dynamics.

4.4.3 Formal Model Specification

To formalize the GA-DSTP model structure, we denote the host-state sequence as

$$x^{(h)}_{1:S_h} \in \mathbb{R}^6,$$

and the driver-related inputs as

$$x^{(d)}_{1:S_d} \in \mathbb{R}^{14},$$

which comprise actuator signals and DMS features. The input features are further explained in Table A.1. In addition, gaze information is provided in the form of gaze targets together with associated quality measures. Each gaze target is embedded into an 8-dimensional representation and combined with gaze-derived features, yielding an 8-dimensional gaze feature vector per time step. This representation is concatenated with the driver inputs, resulting in a 22-dimensional driver-behaviour feature sequence.

Both host-state and driver-behaviour input sequences are processed by separate temporal convolutional encoders. Each encoder consists of three one-dimensional convolutional layers with kernel size 3 and channel widths of 64, 128, and 128, respectively. Each convolution is followed by batch normalization, a GELU activation,

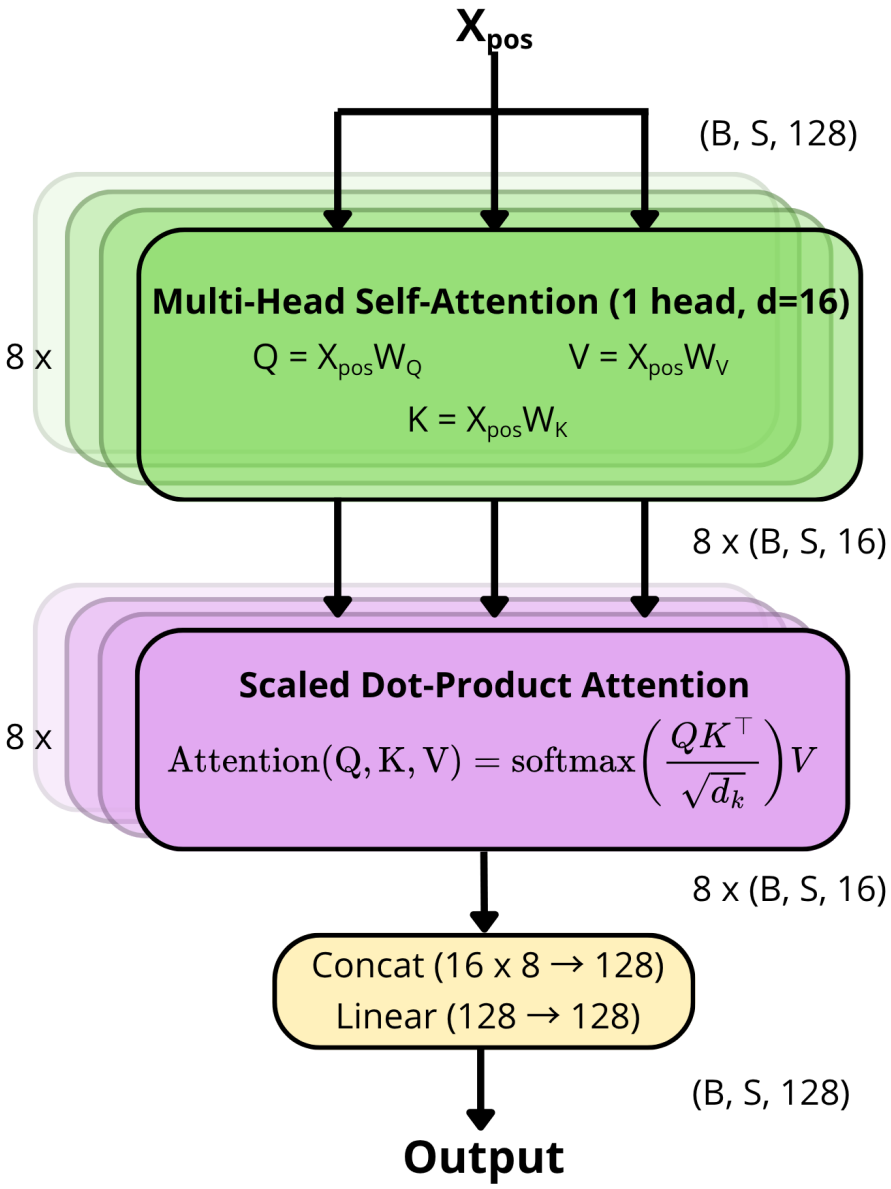


Figure 4.4: Architecture of the GA-DSTP decoder self-attention

and dropout. GELU activations are used throughout the encoders for consistency with the Transformer decoder, where GELU is the standard activation in feed-forward layers [28]. A residual projection is applied at the encoder output to match the final embedding dimensionality, and positional encodings are added to preserve temporal ordering. The two encoders produce modality-specific representations

$$H^{(h)} \in \mathbb{R}^{S_h \times 128}, \qquad H^{(d)} \in \mathbb{R}^{S_d \times 128}.$$

Rather than performing channel-wise fusion, the encoded host and driver representations are concatenated along the temporal dimension to form a unified memory sequence,

$$M = [H^{(h)}; H^{(d)}] \in \mathbb{R}^{(S_h+S_d)\times 128},$$

which is used directly as input to the decoder.
Trajectory prediction is performed using a transformer-style decoder operating on a fixed set of learned queries. Each query is augmented with a mode embedding encoding the trajectory mode identity, as well as a continuous time embedding normalized over the prediction horizon. The decoder comprises four stacked layers, each consisting of multi-head self-attention, cross-attention over the memory sequence M, and a feed-forward network. The decoder outputs a latent tensor of shape

$$(K \times S_{\text{out}}, 128),$$

where K denotes the number of trajectory modes and S_{out} the prediction horizon.

The decoder outputs are reshaped to

$$(B, K, S_{\text{out}}, 128). \tag{4.21}$$

Each mode k has a dedicated control head—a two-layer MLP

$$128 \to 64 \to 2, \tag{4.22}$$

that maps the latent vectors to raw control signals. Tanh bounding is applied to produce physically feasible controls:

$$\delta\kappa_t^{(k)} = \tanh(\cdot)\, \delta\kappa_{\max}, \tag{4.23}$$

$$a_t^{(k)} = \tanh(\cdot)\, a_{\max}. \tag{4.24}$$

These control sequences are integrated through a differentiable Kinematic Bicycle Model initialized from the current vehicle state

$$(v_0, \kappa_0). \tag{4.25}$$

The KBM performs semi-implicit Euler integration per timestep Δt:

$$\kappa_{t+1} = \text{clamp}(\kappa_t + \delta\kappa_t\, \Delta t)\,, \tag{4.26}$$

$$v_{t+1} = \text{clamp}(v_t + a_t\, \Delta t)\,, \tag{4.27}$$

$$\theta_{t+1} = \theta_t + v_{t+1}\kappa_{t+1}\Delta t, \tag{4.28}$$

$$x_{t+1} = x_t + v_{t+1}\cos\theta_{t+1}\Delta t, \tag{4.29}$$

$$y_{t+1} = y_t + v_{t+1}\sin\theta_{t+1}\Delta t. \tag{4.30}$$

The entire rollout is fully differentiable, allowing gradients to flow from the trajectory loss back through the integration and into the control heads and transformer decoder. Using per-mode control heads prevents the Winner-Takes-All training loss from collapsing all modes to identical outputs, since each mode's MLP parameters receive gradients only from that mode's trajectory.

In parallel, a confidence head predicts a categorical distribution over the K modes, producing normalized confidence scores. The model therefore outputs K candidate future trajectories, each consisting of S_{out} waypoints, together with an associated confidence value.

By predicting the full trajectory horizon for all modes in a single, non-autoregressive forward pass, the model avoids temporal error accumulation and is well suited to short-term, multi-modal trajectory forecasting.

4.4.4 Training Objective

The GA-DSTP model is trained in two successive stages, designed to first establish high-quality, diverse trajectory modes and then learn to predict which mode best matches the current driving context. This decoupled strategy avoids the conflicting gradient dynamics that can arise when trajectory regression and confidence estimation are jointly optimised from scratch, an issue previously noted in multimodal trajectory prediction models such as MultiPath++ [10]. All hyperparameter values referenced below are listed in Appendix A.1.

Stage 1: Trajectory Learning with aWTA and Diversity Regularisation

In the first stage, the full model is trained using an *Annealed Winner-Takes-All* (aWTA) loss [39]. Unlike standard WTA, which assigns gradient exclusively to the single best mode and thereby starves all other modes of learning signal, aWTA distributes gradients across all K modes through a soft-weighting scheme:

$$\mathcal{L}_{\text{aWTA}} = \sum_{k=1}^{K} w_k \cdot s_k, \qquad w_k = \frac{\exp(-s_k/\tau_t)}{\sum_{j=1}^{K}\exp(-s_j/\tau_t)}, \tag{4.31}$$

where $s_k = \text{ADE}_k + \lambda_{\text{FDE}} \cdot \text{FDE}_k$ is the per-mode trajectory score and w_k is the soft assignment weight for mode k. The weights are treated as constants during backpropagation so that only the scores s_k receive gradient updates, pulling each mode towards the ground truth in proportion to how close it already is.
The temperature τ_t decays exponentially over training epoch t:

$$\tau_t = \tau_0 \cdot \alpha^t, \tag{4.32}$$

At high temperature the weights are nearly uniform, ensuring all modes learn realistic trajectories early in training. As $\tau_t \to 0$ the distribution sharpens towards standard hard WTA, allowing modes to specialise.
To further prevent modes from collapsing to identical trajectories, a margin-based diversity regularisation term penalises each mode whose nearest-neighbour trajectory distance falls below a minimum separation m:

$$\mathcal{L}_{\text{div}} = \frac{1}{K} \sum_{k=1}^{K} \max\Big(0,\ m - \min_{j \neq k} \bar{d}_{kj}\Big) \tag{4.33}$$

where $\bar{d}_{kj}$ is the mean ℓ_2 distance along the trajectory between modes k and j. The full Stage 1 objective is therefore:

$$\mathcal{L}_{\text{S1}} = \mathcal{L}_{\text{aWTA}} + \lambda_{\text{div}}\, \mathcal{L}_{\text{div}}. \tag{4.34}$$

Stage 2: Joint Fine-Tuning

After Stage 1, the backbone produces high-quality, diverse trajectory modes but has no systematic relationship to which mode best matches the current input. Stage 2 therefore fine-tunes the entire model jointly using differential learning rates, with the confidence head trained at a rate 100× larger than the backbone. This allows the backbone to subtly adapt its representations to become mode-discriminative while preserving the trajectory quality established in Stage 1.
The Stage 2 objective combines a hard WTA trajectory loss with a confidence loss based on Kullback–Leibler (KL) divergence against soft labels:

$$\mathcal{L}_{\text{S2}} = \mathcal{L}_{\text{WTA}} + 2 \cdot \mathcal{L}_{\text{KL}}. \tag{4.35}$$

Rather than assigning a hard label to the single best mode—which is noisy when two modes are similarly close to the ground truth—soft probability targets are constructed as

$$\mathbf{p}^* = \text{softmax}\Big(-\frac{\mathbf{s}}{\tau_{\text{cls}}}\Big), \qquad s_k = \text{ADE}_k + \lambda_{\text{FDE}} \cdot \text{FDE}_k, \tag{4.36}$$

where the low temperature τ_{cls} concentrates probability mass on the best mode while distributing a small residual to close competitors. The confidence loss is then

$$\mathcal{L}_{\text{KL}} = \text{KL}(\mathbf{p}^* \| \hat{\mathbf{p}}) = \sum_{k=1}^{K} p_k^* \log \frac{p_k^*}{\hat{p}_k} \tag{4.37}$$

where $\hat{\mathbf{p}} = \text{softmax}(\mathbf{c})$ are the predicted confidence probabilities from the confidence head. Stage 2 is checkpointed based on the validation ADE of the trajectory selected by the highest confidence prediction. This criterion is preferred over the total loss, as this more directly reflects deployment performance.

4.4.5 Gaze-Target Embedding with Quality-Weighted Fusion

An additional component of the GA-DSTP architecture is a learned gaze-target embedding. The raw gaze-targets, presented in Table 4.1, are replaced with a compact, learned representation. At each time step, the dataset provides up to two gaze-target readings, each consisting of a discrete target_idx $\in \{0, \ldots, 9\}$ and a confidence score $q \in [0, 1]$. Invalid measurements (confidence below 0.20 or missing) are mapped to a padding index 10 with $q = 0$. This is done with an embedding table $E \in \mathbb{R}^{11\times 8}$, where the first ten entries are trainable 8-dimensional vectors corresponding to the gaze targets, and the final entry is a fixed zero vector for padding. These embeddings are learned jointly with the rest of the model.

If two gaze targets are present at a time step, we fuse their embeddings using a confidence-weighted average:

$$\tilde{e}_t = \frac{q_t^{(1)} e_t^{(1)} + q_t^{(2)} e_t^{(2)}}{q_t^{(1)} + q_t^{(2)} + \varepsilon}.$$

If only one reading is valid then $\tilde{e}_t$ reduces to that embedding but if both are invalid, the result is the zero vector. The fused vector replaces the raw gaze-related columns and is concatenated with the continuous actuator features before entering the encoder. This design explicitly addresses known limitations of DMS data discussed in Section 3.2.3, such as unreliable gaze estimates due to occlusion and driver variability.

This representation avoids imposing an artificial ordering on gaze targets (as integer encoding would) and allows the model to learn meaningful relationships between targets (unlike one-hot encoding). Targets with similar behavioral roles may naturally cluster in the embedding space, providing a more informative input representation without manual feature design.

4.4.6 Turn-Indicator One-Hot Encoding

One-hot encoding of turn signal is another feature added to the GA-DSTP architecture to help the model understand them better. The turn-indicator signal is represented using a three-dimensional one-hot encoding. The raw categorical values {“no indicator”, “left”, “right”} are mapped to the vectors

$$\text{no indicator} = [1, 0, 0], \qquad \text{left indicator} = [0, 1, 0], \qquad \text{right indicator} = [0, 0, 1].$$

This encoding avoids introducing an artificial ordinal relationship between indicator states (e.g., left > right) and allows the model to treat each state as an independent categorical outcome. The resulting one-hot vector is concatenated with the other continuous and embedded features before being passed into the actuator encoder.

4.5 Model Evaluation

To ensure that RQ1 and RQ2 are addressed in a fair and meaningful manner, this section outlines the evaluation methodology used for model comparison. A clear and consistent evaluation framework is essential for drawing valid conclusions, and is therefore presented here to avoid confusion.

4.5.1 Classical motion models with team comparison

For each test instance, ADE and FDE are computed for the classical motion models (CA, CV, CTRV, CTRA) and for the GA-DSTP model. A "beat-all" indicator is recorded if the GA-DSTP model strictly improves upon *all* classical baselines simultaneously:

$$\mathbb{I}_{\text{beat-all}} = \mathbb{I}\Big(\text{ADE}_{\text{GA-DSTP}} < \min_{m\in\{\text{CA,CV,CTRV,CTRA}\}} \text{ADE}_m\Big) \wedge \mathbb{I}(\text{FDE}_{\text{GA-DSTP}} < \min_m \text{FDE}_m)\,.$$

The primary summary statistics include the mean of ADE/FDE for each method, and the proportion of test instances with $\mathbb{I}_{\text{beat-all}} = 1$. Per-sample win rate is used as the primary indicator of beat-all performance.

4.5.2 Ablation: With vs. without DMS

The contribution of driver-monitoring signals is isolated by evaluating two otherwise identical models:

1. **GA-DSTP model (with DMS):** Input features comprise host-vehicle state, actuator signals (including one-hot turn indicators), DMS features, and a learned gaze-target embedding that replaces the four raw gaze columns.
2. **DSTP model (without DMS):** Identical architecture and training protocol; DMS features are removed from the input. The actuator-stream input dimensionality is adjusted accordingly. All remaining components (host-state stream, fusion module, decoder, loss, and output heads) are unchanged.

Performance differences are summarized as

$$\Delta\text{ADE} = \text{ADE}_{\text{GA-DSTP}} - \text{ADE}_{\text{DSTP}}, \qquad \Delta\text{FDE} = \text{FDE}_{\text{GA-DSTP}} - \text{FDE}_{\text{DSTP}},$$

along with percentage changes relative to the DSTP model. Positive values indicate improvement attributable to DMS information.

4.5.3 Experimental Protocol

As mentioned in Section 4.2.3 the dataset was divided into an 80/20 split, where the larger portion was used for training and the remaining portion was used for validation. A separate test set collected on a different day was kept isolated and only used for the final performance assessment. Before splitting, the data was shuffled to avoid grouping consecutive segments from the same driving sequence.

A fixed random seed was applied to all data handling steps, including shuffling,

batching, and model initialization. Both model variants, GA-DSTP and DSTP, are trained under identical conditions: same train/validation/test splits, batch size, optimizer and learning rate, number of epochs, early-stopping criterion, random seeds, prediction horizon, and query schedule. Hyperparameters are not re-tuned for the DSTP model.

Each model was trained using the input window and evaluated on the same eight future prediction horizons. Validation was performed after each epoch, and the checkpoint with the lowest validation loss was selected for testing.

This protocol ensured that any observed performance differences between the models were the result of the input feature sets rather than differences in training or evaluation conditions, while accounting for the inherently multimodal nature of vehicle trajectory prediction.

5

Evaluation and Results

This chapter presents the experimental results and evaluation of the proposed GA-DSTP model. The performance of the model is first analyzed in terms of training dynamics and overall prediction error on the test set. It is then compared against classical motion models using standard trajectory prediction metrics, including ADE, FDE, and MR. Further analysis examines performance across different maneuver categories to highlight scenario-dependent behavior. The chapter also includes an ablation study to assess the contribution of DMS features, followed by qualitative trajectory examples that illustrate representative successes and failure cases.

5.1 Training and Evaluation Summary

The training behavior shown in Figures 5.1 and 5.2 indicates stable convergence during both the initial trajectory learning stage and the subsequent joint fine-tuning stage. Table 5.1 reports the ADE and FDE of the GA-DSTP model across the training, validation, and test datasets. The relatively small gap between training and validation performance indicates that the model generalizes well and does not suffer from severe overfitting. While a modest degradation is observed on the held-out test set, the increase in both ADE and FDE remains limited, suggesting that the learned representations capture consistent patterns of vehicle motion and driver behavior that transfer across driving sessions and data splits.

Table 5.1: ADE and FDE of the GA-DSTP model across training, validation and test dataset

Dataset	ADE (m)	FDE (m)
Train	0.364	0.954
Val	0.440	1.185
Test	0.375	0.992

Figures 5.3 and 5.4 show the distributions of ADE and FDE for the GA-DSTP model on the test set. Both distributions are right-skewed, with a high concentration of predictions at low error values and a smaller number of larger-error outliers. The

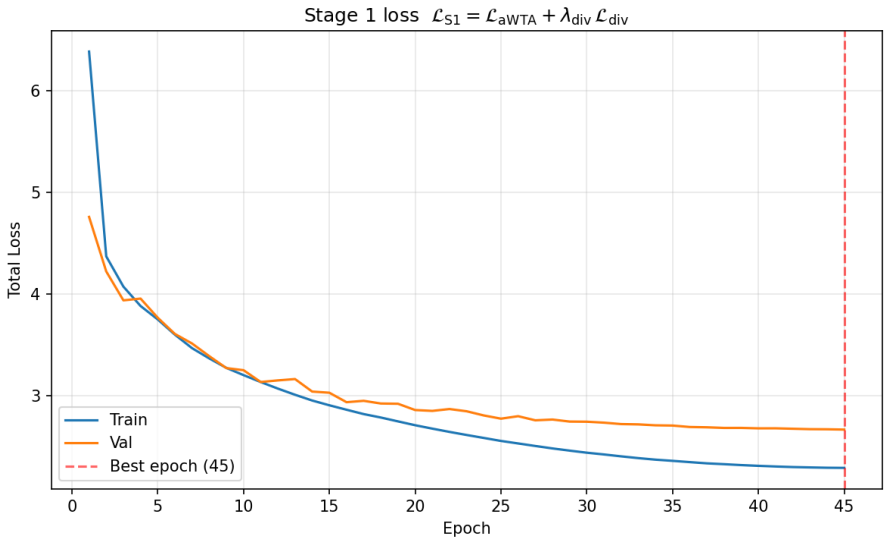


Figure 5.1: Stage 1 (Trajectory learning) training and validation loss up to the epoch with the best validation performance.

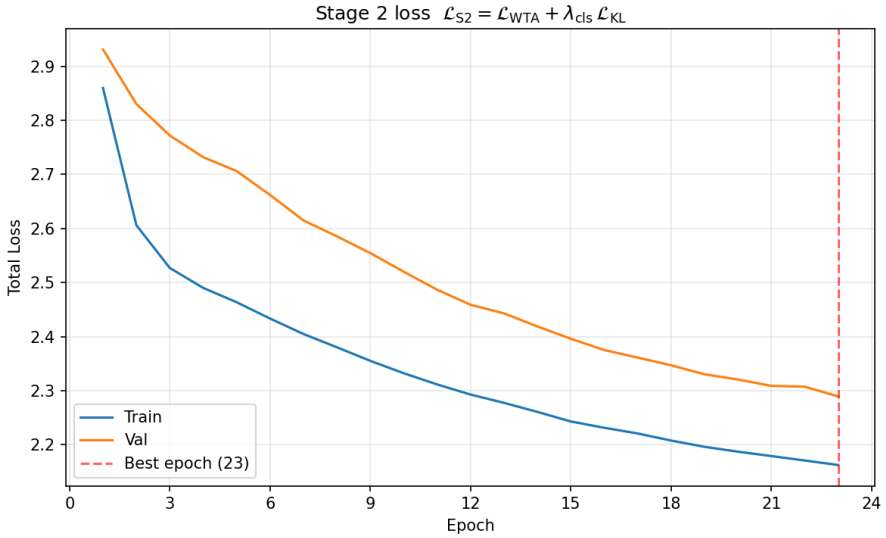


Figure 5.2: Stage 2 (Joint Fine-tuning) training and validation loss up to the epoch with the best validation performance

median error is notably lower than the mean, indicating that most predictions are accurate and that the mean is influenced by a limited number of difficult samples.

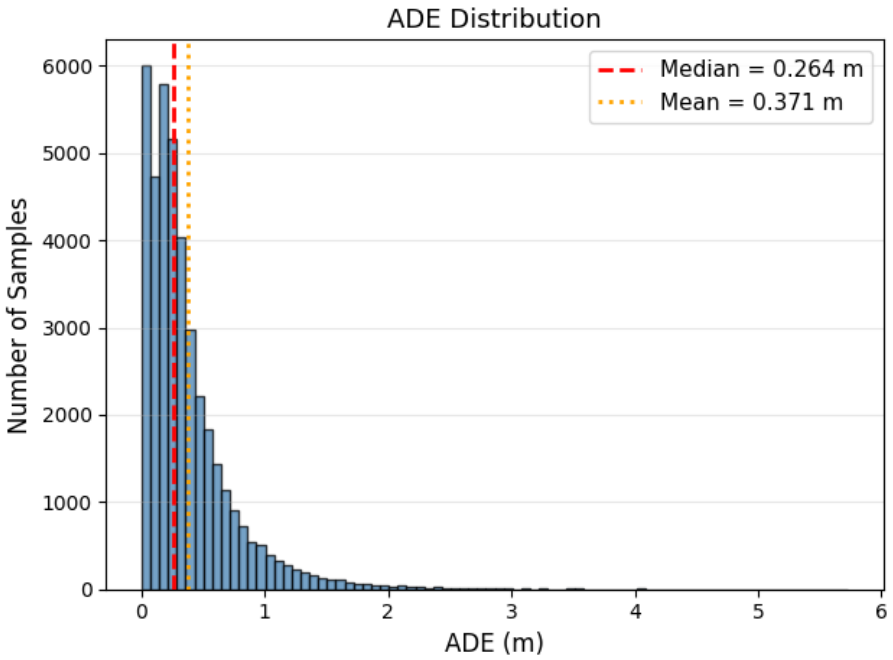


Figure 5.3: Histogram of ADE density of the GA-DSTP model

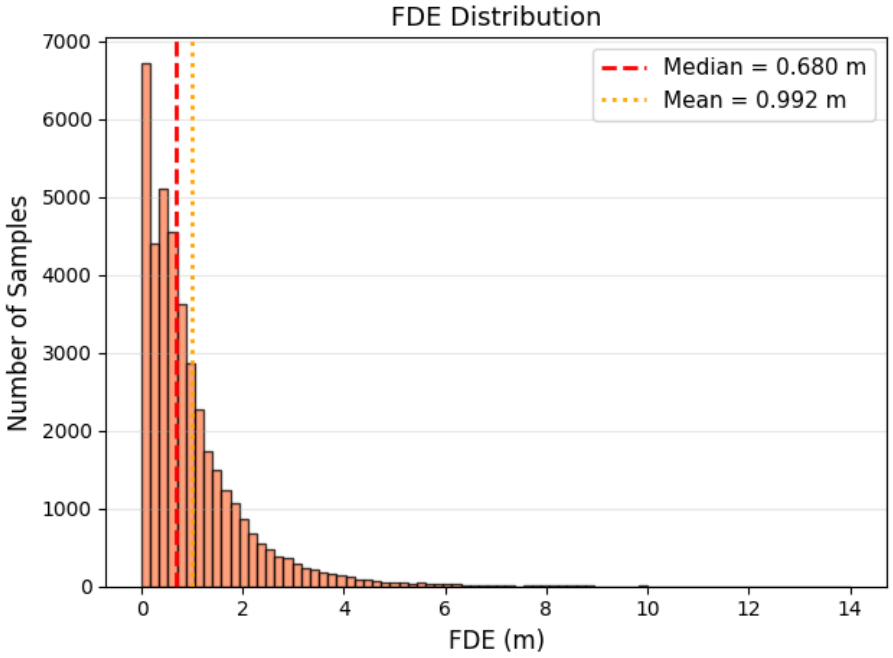


Figure 5.4: Histogram of FDE density of the GA-DSTP model

Figure 5.5 further illustrates how prediction error evolves over the prediction horizon.

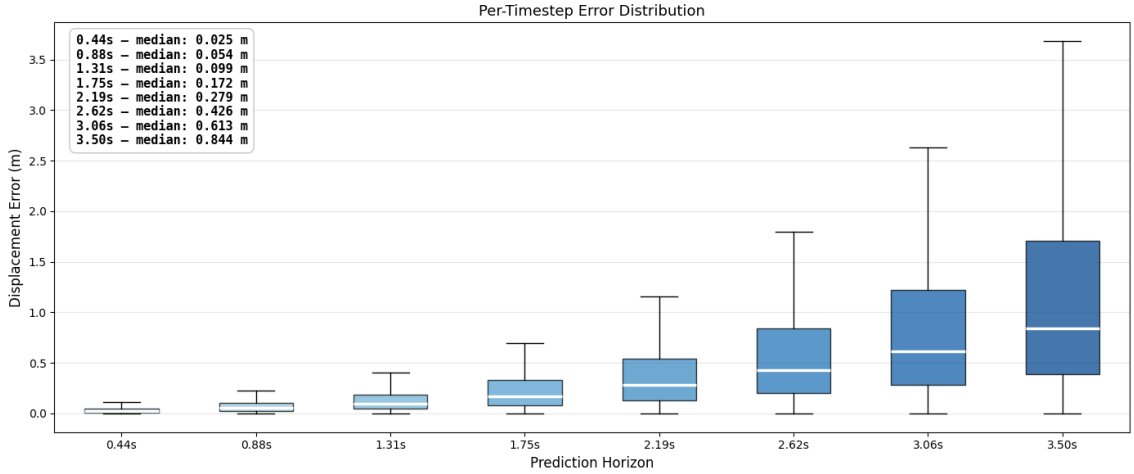


Figure 5.5: Box plot of the displacement error each timestep of the GA-DSTP model. The white line in the box is the median, which are also stated in the legend.

The interquartile range increases steadily with time, reflecting growing uncertainty as the prediction extends further into the future. Nevertheless, the median error remains relatively low throughout the horizon, indicating stable short-term prediction behavior.

5.2 Baseline Comparison with Classical Motion Models

Table 5.2 compares the GA-DSTP model against classical motion models (CV, CA, CTRV, and CTRA) on the test set. The GA-DSTP model achieves substantially lower ADE and FDE than all classical baselines, reducing ADE by approximately 37% compared to CTRA and by more than 60% compared to CV.

Table 5.2: The GA-DSTP model compared to traditional motion models on the test set.

Model	ADE (m)	FDE (m)	MR@0.5 m	MR@1.0 m	MR@2.0 m
CV	0.991	2.438	77.7%	63.4%	44.5%
CA	0.884	2.235	77.7%	63.5%	43.0%
CTRV	0.743	1.892	75.5%	55.9%	33.1%
CTRA	0.642	1.714	76.1%	57.2%	32.4%
GA-DSTP	**0.371**	**0.992**	**61.9%**	**34.7%**	**12.5%**

MR analysis further highlights the improvement. At a 1.0 m threshold, the GA-

DSTP model reduces miss rate from 66.4% (CTRA) to 43.6%, indicating a markedly higher proportion of predictions that are sufficiently accurate for downstream decision-making.

Figure 5.6 and Table 5.3 show that while classical motion models perform competitively at very short horizons, their error grows rapidly after approximately 1.5 seconds. In contrast, GA-DSTP maintains a consistently lower error growth, highlighting its ability to better capture nonlinear vehicle behavior and driver inputs. In Figure 5.6, the upper edge of the interquartile range of GA-DSTP is close to the error levels of the CTRA and CTRV models. This indicates that approximately 75% of GA-DSTP predictions achieve errors comparable to or lower than those classical models.

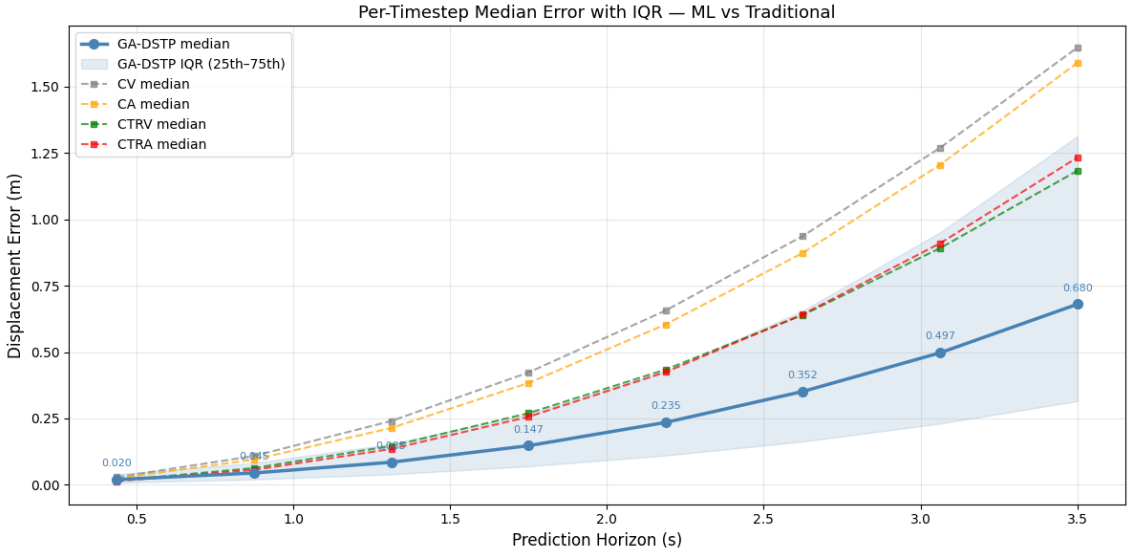


Figure 5.6: Median displacement error and IQR for every timestep of the GA-DSTP model compared to classical motion models.

Table 5.3: Mean displacement error (m) at each prediction timestep.

Model	0.44 s	0.88 s	1.31 s	1.75 s	2.19 s	2.62 s	3.06 s	3.50 s
CV	0.042	0.164	0.364	0.639	0.987	1.405	1.889	2.438
CA	0.035	0.136	0.307	0.548	0.860	1.245	1.704	2.235
CTRV	0.028	0.111	0.253	0.456	0.720	1.048	1.438	1.892
CTRA	**0.020**	0.082	0.194	0.363	0.596	0.896	1.268	1.714
GA-DSTP	0.029	**0.062**	**0.118**	**0.207**	**0.335**	**0.505**	**0.723**	**0.992**

Table 5.4 reports the per-sample win rate against the best-performing classical model. GA-DSTP outperforms the classical team in over 56% of test samples in

Table 5.4: Per-sample win rate: best-of classical team (CV, CA, CTRV, CTRA) versus the GA-DSTP model.

Metric	Classical Team	GA-DSTP
Win rate by ADE	40.5%	59.5%
Win rate by FDE	41.8%	58.2%

terms of both ADE and FDE, demonstrating that its advantage holds on a per-instance basis rather than being driven by a subset of scenarios.

5.3 Performance Across Maneuver Categories

Figure 5.7 and Tables 5.5–5.7 present performance broken down by maneuver category. The GA-DSTP model consistently outperforms classical motion models across most categories, with particularly large improvements observed for turning maneuvers. In Figure 5.7 specifically it is shown that the GA-DSTP model is, on average, better at predicting a right turn than a left turn.

For heavy left and right turns, GA-DSTP achieves approximately 35–40% lower ADE and FDE compared to the best classical baseline, reflecting its ability to model nonlinear steering dynamics that violate constant-turn assumptions.

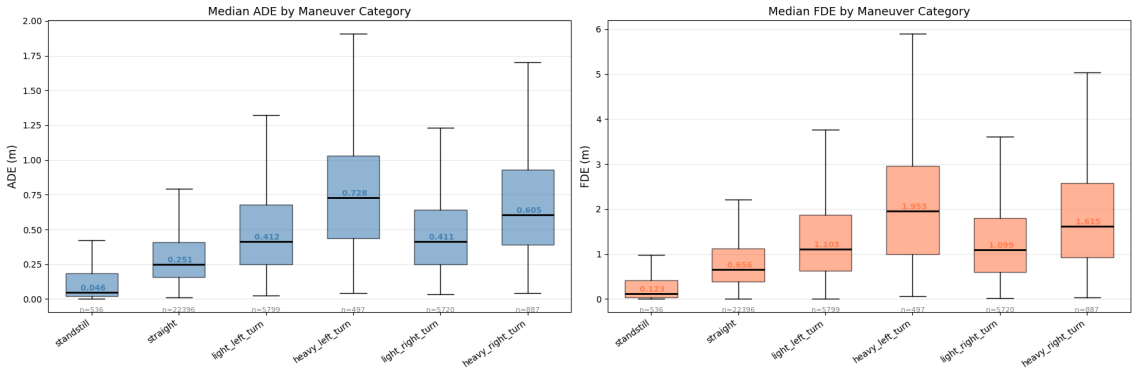


Figure 5.7: Box plot of the displacement error for each maneuver category of the GA-DSTP model. The black line in the boxes is the median.

Table 5.5: ADE in metres by scenario category. Best result per row in **bold**.

Category	N	GA-DSTP	CV	CA	CTRV	CTRA
Standstill	5 187	**0.025**	0.031	0.223	0.031	0.223
Straight	22 396	**0.352**	0.758	0.600	0.745	0.590
Light left turn	5 799	**0.533**	1.659	1.458	1.040	0.850
Heavy left turn	497	**0.824**	3.153	2.964	1.476	1.356
Light right turn	5 720	**0.497**	1.582	1.494	0.950	0.882
Heavy right turn	887	**0.760**	3.085	3.045	1.120	1.013

Table 5.6: FDE in metres by scenario category. Best result per row in **bold**.

Category	N	GA-DSTP	CV	CA	CTRV	CTRA
Standstill	5 187	**0.054**	0.066	0.556	0.066	0.556
Straight	22 396	**0.941**	1.859	1.539	1.877	1.569
Light left turn	5 799	**1.425**	4.100	3.671	2.678	2.304
Heavy left turn	497	**2.197**	7.872	7.433	3.954	3.749
Light right turn	5 720	**1.327**	3.893	3.746	2.430	2.361
Heavy right turn	887	**2.047**	7.637	7.586	2.999	2.788

Table 5.7: Miss Rate at 1.0 m threshold (%) by scenario category. Lower is better; best per row in **bold**.

Category	N	GA-DSTP	CV	CA	CTRV	CTRA
Standstill	5 187	**1.7**%	2.5%	15.4%	2.5%	15.4%
Straight	22 396	**29.6**%	57.9%	55.5%	56.6%	54.7%
Light left turn	5 799	**54.9**%	97.2%	96.2%	74.3%	76.3%
Heavy left turn	497	**74.8**%	100.0%	100.0%	92.0%	89.1%
Light right turn	5 720	**54.4**%	97.0%	96.3%	74.8%	78.5%
Heavy right turn	887	**72.2**%	96.8%	99.3%	86.1%	80.3%

As seen in Table 5.8 the performance gains are especially pronounced in standstill scenarios, where classical motion models frequently fail due to their reliance on constant velocity or acceleration assumptions. Straight-driving scenarios constitute approximately 52% of the test set, which means that the performance in this category is particularly important for overall metrics. In these scenarios, the GA-DSTP model achieves a win rate above 50% in terms of ADE, indicating that it provides more accurate short-horizon predictions than the classical baselines for the majority of straight-driving samples.

However, the corresponding win rate in terms of FDE is lower, indicating that while GA-DSTP is effective at capturing the immediate future motion in straight trajectories, long-horizon prediction differences between models are less pronounced.

Table 5.8: Per-sample win rate: best-of classical team (CV, CA, CTRV, CTRA) versus the GA-DSTP model for each maneuver category.

Metric	Classical Team	GA-DSTP
Win rate by ADE		
Straight	48.9%	**51.1%**
Left turn	40.0%	**60.0%**
Right turn	39.9%	**60.1%**
Standstill	5.3%	**94.7%**
Win rate by FDE		
Straight	**51.7%**	48.3%
Left turn	39.2%	**60.8%**
Right turn	39.2%	**60.8%**
Standstill	5.5%	**94.5%**

5.4 Ablation Study: Effect of DMS Features

Figure 5.8 shows an inference-time feature masking experiment conducted on the validation set to assess whether GA-DSTP actively uses DMS and gaze information. The model was evaluated under three conditions: with all features available, with DMS and gaze features masked to zero, and with the entire driver behavior stream masked.

The full model achieved an ADE of 0.44 m and FDE of 1.18 m. Masking DMS and gaze features increased the error to 0.47 m ADE (+6.8%) and 1.27 m FDE (+7.6%). Masking the entire driver stream resulted in substantially larger degradation, with ADE increasing to 0.63 m (+43%) and FDE to 1.70 m (+44%).

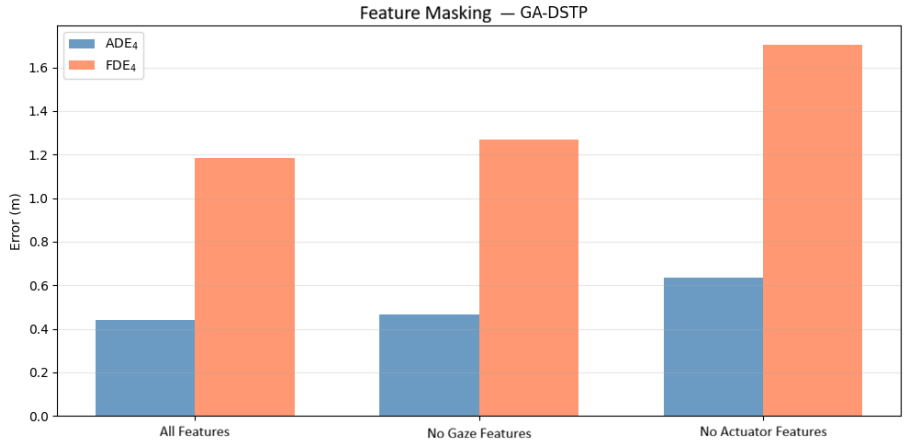


Figure 5.8: This shows the effect of error when certain features are removed from the GA-DSTP model on the validation set.

Tables 5.9–5.12 compare the GA-DSTP model with the DSTP variant that excludes DMS signals. Incorporating DMS features results in consistent improvements across all global metrics, reducing ADE by 1.7% and FDE by 2.1% on the test set.

While the absolute improvements are modest, they are systematic and most pronounced in scenarios involving braking and turning maneuvers. In heavy right turns, the inclusion of DMS features decreases MR by 2.4 percentage points.

Table 5.9: Ablation study: effect of DMS data by comparing GA-DSTP model vs DSTP model.

Model	**ADE (m)**	**FDE (m)**	**MR@0.5 m**	**MR@1.0 m**	**MR@2.0 m**
GA-DSTP	0.375	0.992	61.9%	34.7%	12.5%
DSTP	0.383	1.003	62.8%	34.9%	12.7%
Improvement	−2.1%	−1.1%	−0.9 pp	−0.2 pp	−0.2 pp

Table 5.10: ADE (m) by scenario category for GA-DSTP model and DSTP model. Best result per row in **bold**.

Category	N	GA-DSTP	DSTP	Δ
Standstill	5 187	0.025	**0.013**	+0.012
Straight	22 359	**0.352**	0.371	-0.019
Light left turn	5 799	**0.533**	0.545	-0.012
Heavy left turn	497	**0.824**	0.887	-0.063
Light right turn	5 720	**0.497**	0.498	-0.001
Heavy right turn	887	0.760	**0.720**	+0.040

Table 5.11: FDE (m) by scenario category for GA-DSTP model and DSTP model. Best result per row in **bold**.

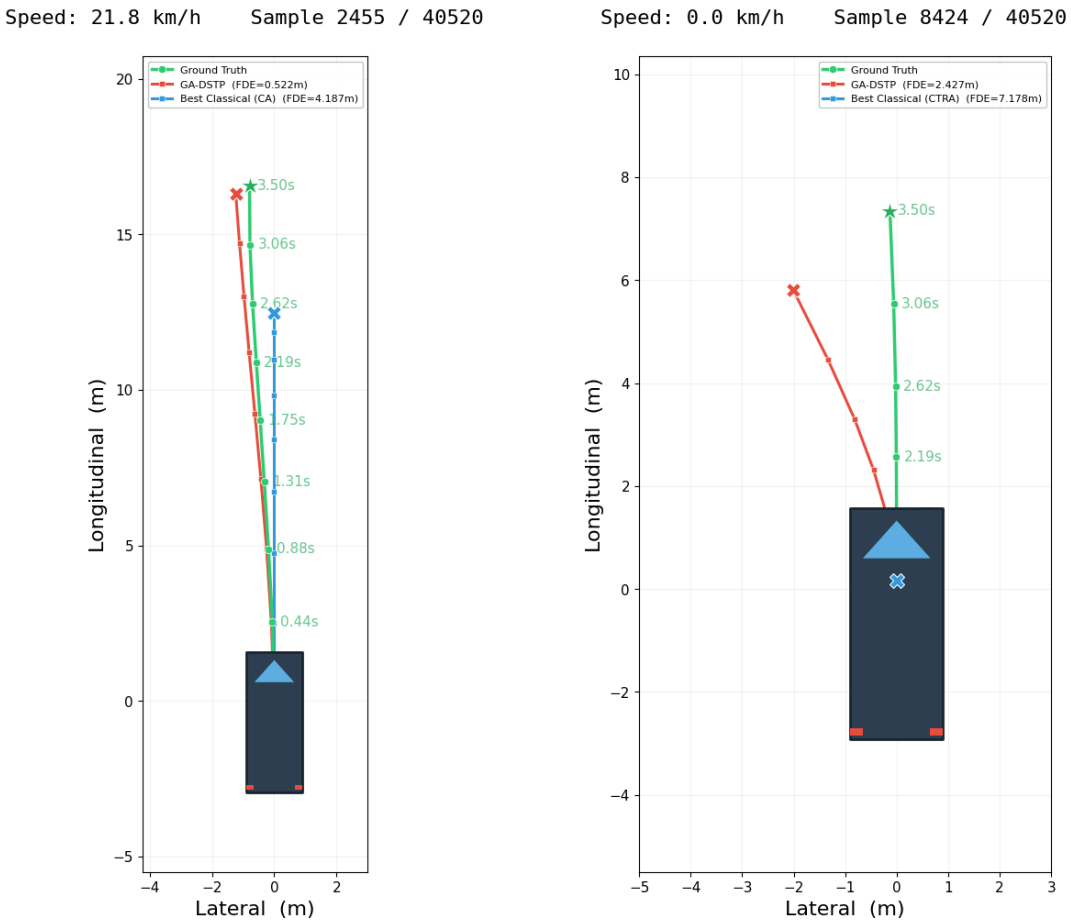
Category	N	GA-DSTP	DSTP	Δ
Standstill	5 187	0.054	**0.027**	+0.027
Straight	22 396	**0.941**	0.963	-0.022
Light left turn	5 799	**1.425**	1.446	-0.021
Heavy left turn	497	**2.197**	2.311	-0.114
Light right turn	5 720	1.327	**1.320**	+0.007
Heavy right turn	887	2.047	**1.963**	+0.084

Table 5.12: Miss Rate at 1.0 m (%) by scenario category for GA-DSTP model and DSTP model. Lower is better; best per row in **bold**.

Category	N	GA-DSTP	DSTP	Δ (pp)
Standstill	5 187	1.7	**0.6**	+1.1
Straight	22 396	**29.6**	29.9	−0.3
Light left turn	5 799	**54.9**	55.3	−0.4
Heavy left turn	497	**74.8**	77.5	−0.7
Light right turn	5 720	**54.4**	54.9	−0.5
Heavy right turn	887	**72.2**	74.6	−2.4

5.5 Qualitative Trajectory Examples

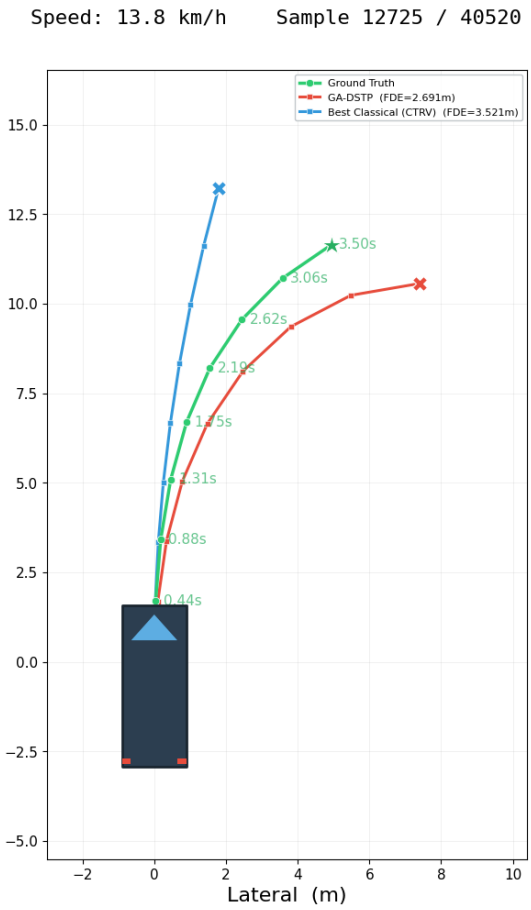
The following plots illustrate representative success and failure cases. The green line corresponds to the ground truth trajectory, the red line to the GA-DSTP prediction, and the blue line to the best-performing classical motion model for the given sample.



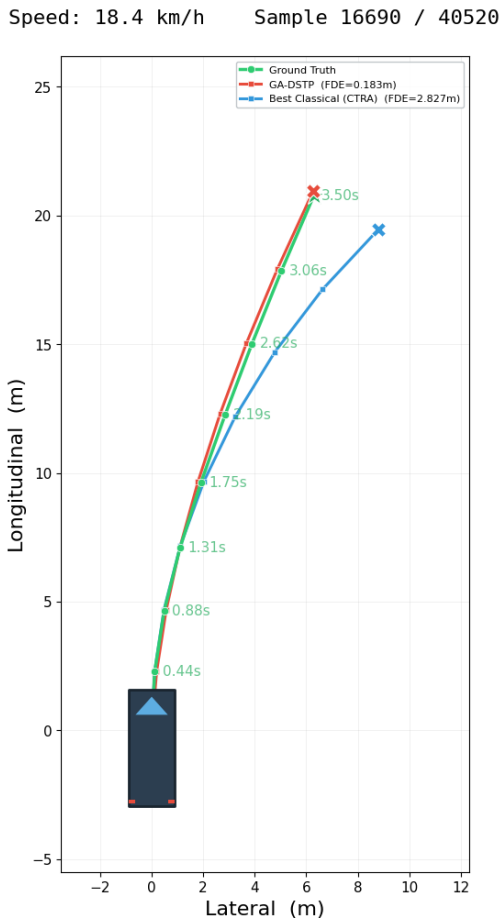
(a) Straight driving failure of the classical model, where GA-DSTP maintains accurate longitudinal tracking.

(b) One common issue observed in the GA-DSTP model occurs during the initial phase of acceleration from standstill, where the predicted trajectory becomes unstable.

Figure 5.9: Examples of longitudinal motion prediction in straight-driving scenarios.

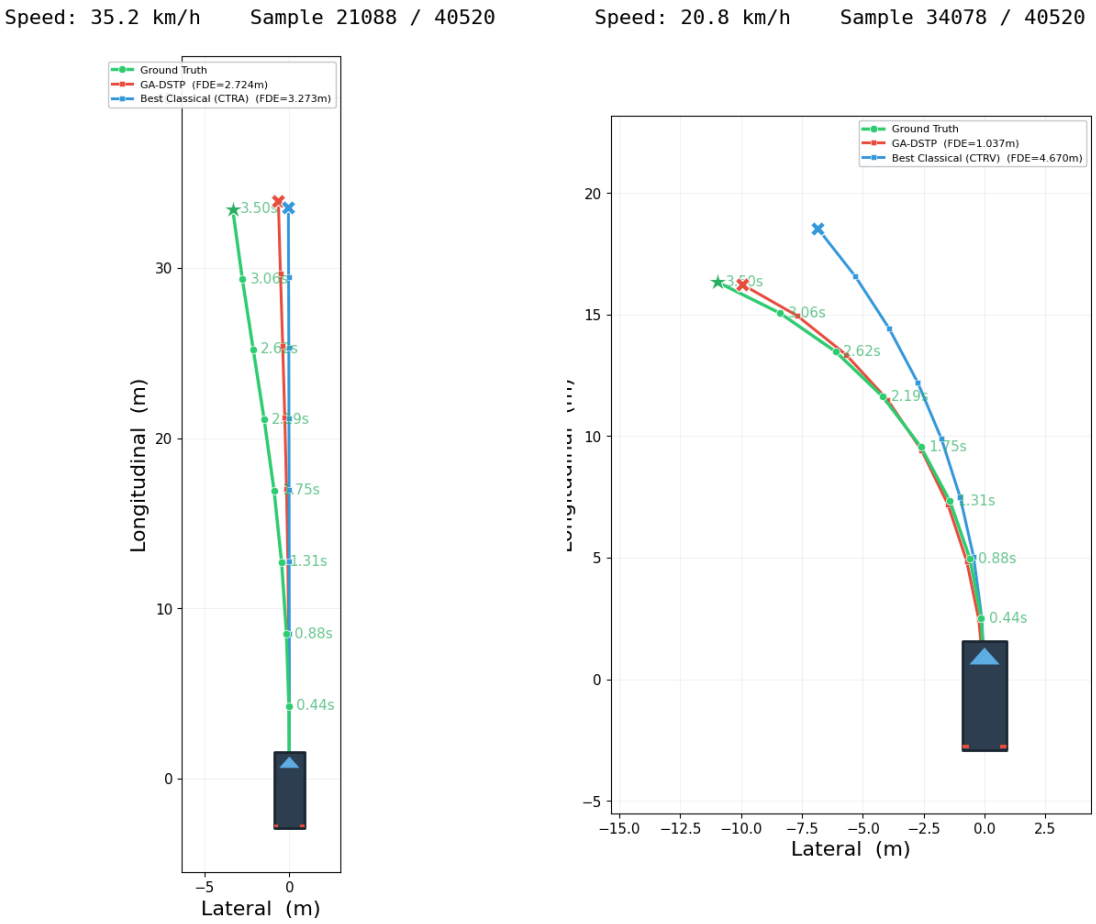


(a) Right turn where GA-DSTP overestimates steering, leading to accumulated lateral error.



(b) Right turn where GA-DSTP closely follows the ground truth trajectory.

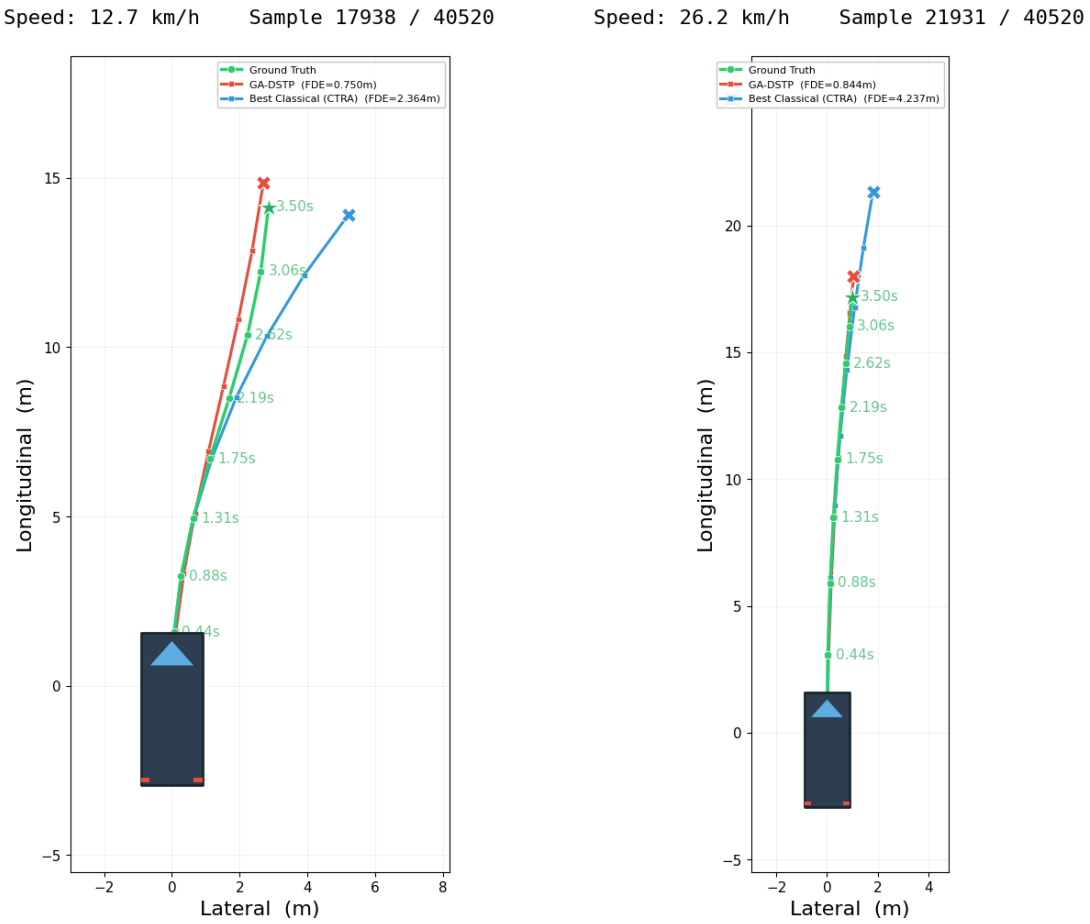
Figure 5.10: Contrasting GA-DSTP behavior in right-turn scenarios.



(a) Left turn where neither GA-DSTP nor the classical model accurately predicts the maneuver.

(b) Left turn where both models perform well, with a slight advantage for GA-DSTP.

Figure 5.11: Examples of left-turn prediction performance.



(a) Failure of the constant turn-rate assumption, resulting in a fixed-curvature trajectory that deviates from the ground truth.

(b) Braking scenario where the GA-DSTP model anticipates deceleration earlier than the classical motion model.

Figure 5.12: Examples illustrating limitations of classical motion assumptions and anticipatory behavior of GA-DSTP.

5.6 Summary of Key Findings

The experimental evaluation demonstrates that the proposed GA-DSTP model achieves substantially lower prediction errors than classical motion models across all global trajectory prediction metrics. In particular, GA-DSTP consistently reduces both ADE and FDE on the test set and achieves a lower MR at all evaluated thresholds.

Analysis of prediction error over time shows that GA-DSTP maintains slower error growth across the entire prediction horizon compared to classical baselines, indicating improved robustness at longer horizons. While classical motion models perform competitively at very short prediction times, their error increases rapidly after ap-

proximately 1.5 seconds.

Scenario-specific evaluation reveals that GA-DSTP provides the largest performance gains in turning maneuvers, where constant-velocity and constant-acceleration assumptions are frequently violated. For both left and right turns, GA-DSTP achieves significantly lower ADE and FDE than all classical models and outperforms them on a per-sample basis. Straight-driving scenarios show more moderate improvements, with GA-DSTP achieving higher short-horizon accuracy while long-horizon differences are less pronounced.

The ablation study shows that incorporating driver monitoring system features leads to consistent improvements across global metrics and most maneuver categories. While the absolute gains from DMS data are modest, they are most pronounced in turning scenarios, indicating that driver-related signals provide complementary information beyond vehicle-state history alone.

Finally, qualitative trajectory examples illustrate both the strengths and limitations of the proposed approach. GA-DSTP demonstrates improved anticipation of braking and nonlinear steering behavior compared to classical motion models, while failure cases highlight challenges in scenarios involving delayed driver intent or sudden motion changes such as acceleration from standstill.

6
Conclusion

This chapter interprets the experimental results presented in Chapter 5 and discusses their implications for short-horizon vehicle trajectory prediction and the use of driver monitoring data. The discussion focuses on understanding the observed performance trends, the scenarios in which the proposed model provides the greatest benefit, and the limitations revealed by the results.

6.1 Overall Model Behavior and Training Stability

The relatively small gap between training, validation, and test performance reported in Table 5.1 suggests that the GA-DSTP model generalizes well and does not suffer from severe overfitting. The layer sizes of the architecture were made smaller to prevent the model memorizing the training data. That was a problem because the dataset collected is rather small with less datatypes compared to public datasets such as Waymos open dataset [40]. This is particularly important in trajectory prediction tasks, where temporal correlations in driving data can easily lead to overly optimistic training performance.

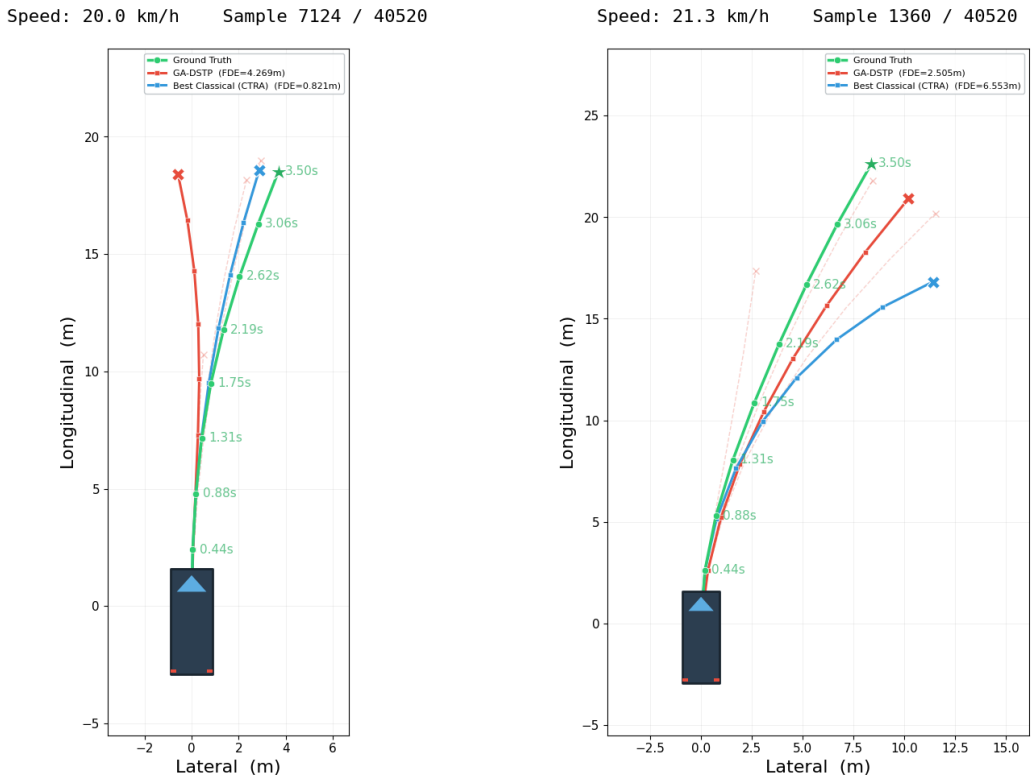
Beyond convergence stability, the two-stage training strategy plays an important role in shaping model behavior. By first focusing on learning geometrically plausible and diverse trajectory modes, and only later learning to select among them, the model avoids optimization instabilities commonly associated with jointly training trajectory regression and confidence estimation from scratch. This decoupling contributes to reliable learning dynamics and helps prevent mode collapse, which is especially relevant for multi-modal trajectory prediction over longer horizons.

Table 6.1 presents the minADE and minFDE, which defines the error of the closest predicted trajectory to the ground truth across all four modes, alongside the ADE and FDE of the confidence-selected trajectory. The gap between the minimum metrics and the confidence-selected metrics indicates how much performance is lost due to imperfect mode selection. This shows the potential of the model if it would choose the best trajectory more frequently. A plausible method would be to include radar data such that we can exclude a modes trajectory if there are something in the way. This is ofcourse with the assumption that the driver is alert.

Table 6.1: GA-DSTP performance across data splits.

Split	**minADE**	**minFDE**	**ADE**	**FDE**
Train	0.2909	0.7791	0.3643	0.9539
Validation	0.3420	0.9392	0.4394	1.1858
Test	0.2918	0.7981	0.3714	0.9922

Figure 6.1 presents qualitative examples that help explain the behavior summarized in Table 6.1. Figure 6.1a show a prediction where some modes are very close to the ground truth but the model choose one that turns the complete opposite way. This behavior could occur if the driver is distracted and not looking same way as they are turning for a while. Then the model believes the driver will turn the other direction and chooses the incorrect mode to proceed with. Figure 6.1b shows a case where the selected trajectory is reasonable but a more accurate mode is available among the predictions. Overall, this indicate that the stage 1 training produces accurate and diverse predictions, while the gap to the confidence-selected metrics suggests that the stage 2 training has room for improvement.



(a) This figure shows an example of having good predictions but choosing the wrong one during stage 2.

(b) The prediction shown in this figure is good but could be even better if it would choose the more accurate trajectory instead.

Figure 6.1: Examples illustrating limitations of the stage 2 learning.

The error distributions shown in Figures 5.3 and 5.4 reveal that most predictions achieve relatively low displacement error, while a smaller subset of difficult samples contributes disproportionately to the error tail. Importantly, the median is left of the mean in both figures which indicates that there are a large quantity of samples with a low error. Such long-tailed distributions are common in trajectory prediction and reflect the presence of ambiguous or highly dynamic maneuvers. Figure 5.5 further shows that uncertainty increases steadily over the 3.5-second prediction horizon, as expected. The error is even more expected to rise over time because of our uncertainty of the surroundings. Nevertheless, the median error remains comparatively low throughout the horizon, indicating that GA-DSTP maintains stable short-term prediction performance even as uncertainty accumulates.

6.2 Comparison with Classical Models

To address **RQ1**, whether data-driven trajectory prediction models can outperform classical analytical motion models, we compare the proposed GA-DSTP model against established kinematic baselines. As shown in Table 5.2, GA-DSTP substantially outperforms all classical baselines across ADE, FDE, and MR metrics.

Importantly, this comparison is conservative in favor of the classical approaches, as the reported baseline performance reflects a best-of selection across CV, CA, CTRV, and CTRA. In other words, the classical models are evaluated as a team, while GA-DSTP produces a single prediction conditioned only on past observations. A key distinction lies in the information available to each approach. Classical motion models rely on a single vehicle state and a fixed motion assumption, while GA-DSTP leverages a longer temporal history of both vehicle dynamics and driver behavior. Figure 5.6 and Table 5.3 show that classical models perform competitively at very short horizons but degrade rapidly beyond approximately 1.5 seconds. In contrast, GA-DSTP exhibits significantly slower error growth across the entire prediction horizon. The proximity of the upper interquartile range of GA-DSTP to the median error of the best classical models in Figure 5.6 indicates that a substantial majority of GA-DSTP predictions achieve accuracy comparable to or better than classical motion models, even at longer horizons.

The per-sample win-rate analysis in Table 5.4 further confirms that these improvements are not driven by a small subset of scenarios. GA-DSTP outperforms the classical team in over half of the test samples for both ADE and FDE, demonstrating a consistent advantage on an instance-by-instance basis.

An important observation is that the relative advantage of GA-DSTP depends on the driving regime. At higher vehicle speeds, motion tends to be smoother and more constrained by road geometry, making it closer to the assumptions underlying kinematic motion models. As speed increases, the gap between GA-DSTP and classical models therefore narrows. Conversely, at lower speeds, such as in urban driving, maneuvering, standstill, and stop-and-go traffic, uncertainty increases and driver influence becomes more pronounced. In these regimes, GA-DSTP shows clearer performance gains, as it is better able to capture deviations from idealized motion assumptions.

Turning scenarios further illustrate this behavior. As shown in Tables 5.5–5.7, GA-DSTP provides particularly large improvements in both left- and right-turn maneuvers. These scenarios violate the constant-turn-rate assumptions used by classical models and often involve early driver decisions that are not immediately observable in vehicle state. An interesting result is that the GA-DSTP model is better at predicting right turns than left turns, in both the heavy and light category. The reason may have to do with the dataset where right turns often occur during lower speeds and more standardized in contrast to left turns which often has a larger turning radius and interaction with opposing traffic. It could also be that left turns often involve delayed or ambiguous driver intent, which makes early trajectory prediction inherently more challenging compared to right turns that are typically executed in a more decisive manner.

Overall, these results provide clear evidence that the proposed approach effectively addresses RQ1, demonstrating that data-driven models can outperform classical motion models for short-horizon trajectory prediction.

6.3 Impact of DMS Features

To address **RQ2**, whether the inclusion of DMS data improves the performance of data-driven trajectory prediction models, we analyze the contribution of driver monitoring features through ablation studies. As shown in Figure 5.8, the degradation in performance when gaze features are masked demonstrates that GA-DSTP has learned to actively exploit this information, rather than treating it as noise or routing around it. This validates that the performance improvement of GA-DSTP over DSTP (which lacks DMS and gaze features) is due to genuine utilization of driver monitoring signals rather than architectural differences alone. The larger degradation when masking all driver behavior features confirms that actuator signals (steering, pedals, turn indicators) remain the most informative driver inputs, with DMS and gaze providing supplementary predictive value.

The ablation study presented in Tables 5.9–5.12 shows that incorporating DMS data yields consistent, though modest, performance improvements across most metrics. These gains are most pronounced in braking and turning scenarios, where driver intent typically precedes observable changes in vehicle motion. The GA-DSTP model in Table 5.12 has the lowest MR in all categories except standstill which indicates that it is better at keeping the trajectories more reasonable than the DSTP model. This suggests that DMS features provide complementary information that enables earlier anticipation of upcoming maneuvers.

At the same time, the limited magnitude of the improvements highlights practical challenges associated with driver monitoring data. Camera occlusions, unfavorable lighting conditions, or temporary loss of visibility can degrade signal quality, reducing the reliability of gaze- and head-based features. The DMS camera was located behind the steering wheel which implies that during sharp turns the camera was blocked by the steering wheel or hands. Therefore, during difficult maneuvers, most of the DMS data is unavailable. In addition, driver behavior is highly individual: different drivers attend to different visual cues and exhibit distinct head-movement and gaze patterns. This inter-driver variability introduces noise and limits the extent to which a single model can fully exploit DMS signals without personalization or adaptation.

These findings are in line with Doshi et al. [17], who found that head-movement dynamics provide a stronger indication of driver intent than eye gaze alone. A likely reason is that gaze behavior during driving is often ambiguous, since drivers frequently shift their eyes between the road, mirrors, dashboard, and other points of interest without necessarily intending to perform a maneuver. Head movements, by contrast, are often more closely aligned with an upcoming action and may therefore provide a clearer indication of future vehicle motion. This may help explain why the DMS features in this thesis yield only modest improvements overall, suggesting that gaze-related cues alone are not always sufficient to robustly distinguish future driving behavior.

Overall, these findings support RQ2, showing that the inclusion of DMS features provides complementary information that improves trajectory prediction performance, although the impact remains modest due to practical sensing limitations.

6.4 Model Limitations and Future Work

Several limitations are revealed by the quantitative results presented in Chapter 5 and by the qualitative trajectory examples in Section 5.5. Addressing these limitations highlights clear directions for future research.

A primary limitation is that the GA-DSTP model does not explicitly incorporate information about the surrounding environment, such as neighboring vehicles, lane geometry, intersections, or traffic control elements. As a result, the model cannot reason about interaction-driven behavior. This limitation is particularly apparent in turning maneuvers, which often depend on external factors such as oncoming traffic, road curvature, and intersection layout. As shown in Tables 5.5 and 5.6, although GA-DSTP substantially outperforms classical motion models in both left and right turns, prediction errors remain significantly higher than in straight-driving scenarios. Qualitative examples in Figures 5.11a and 5.10a further illustrate that delayed steering decisions or external constraints can lead to large prediction errors even when temporal vehicle and driver data are available. Incorporating explicit environmental context would allow the model to better distinguish between different turning outcomes and could further improve performance in these scenarios.

A second limitation concerns maneuver categorization. In this work, maneuver labels are derived using rule-based thresholds on kinematic displacements, as described in Section 4.2.3. While this approach is sufficient for large-scale evaluation and enables systematic comparison across models, it may misclassify transitional behaviors, particularly during the onset and completion of turns. This effect may contribute to the elevated error variance observed in turning categories, as reflected in Figure 5.7. Future work could benefit from manual maneuver labeling using synchronized video data, which would provide more accurate and semantically meaningful turn annotations and enable deeper analysis of driver intent during maneuver transitions. Further insight could be gained by analyzing samples with the highest prediction errors. The ADE and FDE distributions shown in Figures 5.3 and 5.4 indicate that a relatively small fraction of samples dominates the error tail. Qualitative inspection of these high-error cases reveals failure modes such as sudden acceleration from standstill and delayed steering inputs. For example, the acceleration scenario in Figure 5.9b shows that GA-DSTP fails to anticipate motion onset, while the left-turn example in Figure 5.11a demonstrates that both GA-DSTP and classical models struggle in cases of ambiguous or late maneuver execution. These examples highlight the limitations of deterministic trajectory prediction in situations where multiple future motions are equally plausible. This observation motivates future extensions toward probabilistic or explicitly multi-modal prediction frameworks that can better capture uncertainty in real-world driving.

Finally, while driver monitoring data contributes positively to prediction performance, its effectiveness is constrained by sensing limitations and inter-driver variability. The ablation results in Tables 5.9–5.12 show consistent but modest improvements when DMS features are included, particularly in turning and braking scenarios. Eye-gaze information provides useful cues about driver attention and workload, but gaze signals are sensitive to noise and occlusions. As discussed in Section 3.2.3, large steering inputs can partially occlude the DMS camera, which coincides with situations where intent cues are most valuable. Prior work has shown that the dynamics of head movement provide more robust indicators of driver intent than gaze alone, particularly for anticipating turning maneuvers [17]. The scenario-dependent improvements observed in Tables 5.10 and 5.11 are consistent with this finding and suggest that future systems could benefit from placing greater emphasis on head-motion dynamics, potentially combined with gaze information and driver-specific adaptation.

6.5 Summary

This thesis investigated short-horizon vehicle trajectory prediction with a focus on incorporating human-context information through driver monitoring data. The overall objective was to assess whether fusing driver-related signals with traditional vehicle motion data could improve predictive performance in safety-critical ADAS scenarios where driver intent plays a significant role.

A learning-based trajectory prediction model was developed that combines host vehicle dynamics, actuator inputs, and DMS features within a dual-stream architecture. By separating vehicle state information from driver behavior during encoding and fusing them at a later stage, the model was designed to capture both physical motion tendencies and anticipatory cues related to driver awareness. The use of a multi-modal prediction framework enabled the model to represent multiple plausible future trajectories, which is particularly important in situations characterized by behavioral uncertainty. Experimental results demonstrate that the proposed approach consistently outperforms classical kinematic motion models across standard trajectory prediction metrics. The performance gains are most pronounced in scenarios involving turning, braking, and standstill maneuvers, where classical assumptions of constant motion are frequently violated.

An ablation study further shows that incorporating DMS features leads to systematic improvements over an otherwise identical model without driver monitoring inputs, confirming that driver awareness information provides complementary value beyond vehicle dynamics alone. While the absolute improvements contributed by DMS data are moderate, they are stable across scenarios and align with the intended application of short-horizon prediction in ADAS. The findings suggest that driver-related signals are particularly beneficial in situations where vehicle motion alone is insufficient to anticipate imminent maneuvers. At the same time, the results highlight remaining challenges related to uncertainty, data limitations, and mode selection, indicating opportunities for future improvements.

In summary, this work demonstrates that integrating driver monitoring data into learning-based trajectory prediction models can enhance short-term prediction accuracy and robustness. The results support the inclusion of human-context information as a meaningful complement to traditional motion-based approaches in the development of future ADAS trajectory prediction systems.

Bibliography

[1] Alsanwy, S., Chalak Qazani, M. R., Al-Ashwal, W., Shajari, A., Nahvandi, S., & Asadi, H. (2024). *Vehicle Trajectory Prediction Using Deep Learning for Advanced Driver Assistance Systems*. IEEE. Retrieved from https://ieeexplore.ieee.org/abstract/document/10553601

[2] MITRE (2024). *Advanced Driver Assistance Systems Continue to Expand, New Report Shows*. Retrieved from https://www.mitre.org.

[3] Shah, V., & Patel, H. (2025). *Integrating Machine Learning and Computer Vision in Advanced Driver Assistance Systems: A Comprehensive Review*. Springer Nature.

[4] Raskoti, C., Islam, I., Wang, X., & Li, W. (2025). *MIAT: Maneuver-Intention-Aware Transformer for Spatio-Temporal Trajectory Prediction*. arXiv preprint arXiv:2504.05059. Retrieved from https://arxiv.org/html/2504.05059v1

[5] Li, P., Liu, Z., Shan, H., & Chen, C. (2025). *Enhanced Broad-Learning-Based Dangerous Driving Action Recognition on Skeletal Data for Driver Monitoring Systems*. Sensors, 25(6). Retrieved from https://www.mdpi.com/1424-8220/25/6/1769

[6] Dahl, J., de Campos, G. R., & Fredriksson, J. (2023). *Intention-Aware Lane Keeping Assist Using Driver-Gaze Information*. IEEE Intelligent Vehicles Symposium (IV). Retrieved from https://research.chalmers.se/publication/535461/file/535461_Fulltext.pdf

[7] R. Schubert, E. Richter, and G. Wanielik, "Comparison and Evaluation of Advanced Motion Models for Vehicle Tracking," in *Proceedings of the 11th International Conference on Information Fusion (FUSION)*, Cologne, Germany, 2008. Retrieved from https://ieeexplore.ieee.org/document/4632283

[8] Khakzar, A., Rezvani, M., and Shahbazi, Z. (2022). *A Dual Learning Model for Vehicle Trajectory Prediction*. IEEE Transactions on Intelligent Vehicles. Retrieved from https://arxiv.org/abs/2111.14973

[9] Zhang, Z., Wang, C., Zhao, W., Cao, M., and Liu, J. (2024). *Ego Vehicle Trajectory Prediction Based on Time-Feature Encoding and Physics-Intention Decoding*. IEEE Transactions on Intelligent Transportation Systems, vol. 25, no. 7, pp. 6527–6541. doi:10.1109/TITS.2023.3344718.

[10] Varadarajan, B., Hefny, A., Srivastava, A., Refaat, K. S., Nayakanti, N., Cornman, A., Chen, K., Douillard, B., Lam, C. P., Anguelov, D., & Sapp, B. (2021). *MultiPath++: Efficient Information Fusion and Trajectory Aggregation for Behavior Prediction*. arXiv preprint arXiv:2111.14973. Retrieved from https://arxiv.org/abs/2111.14973

[11] Schubert, R., Richter, E., & Wanielik, G. (2008). *Comparison and Evaluation of Advanced Motion Models for Vehicle Tracking*. Proceedings of the 11th International Conference on Information Fusion. Retrieved from http://fusion.isif.org/proceedings/fusion08CD/papers/1569107835.pdf

[12] Rajamani, R. (2012). *Vehicle Dynamics and Control* (2nd ed.). Springer. https://doi.org/10.1007/978-1-4614-1433-9

[13] Pacejka, H. B. (2012). *Tire and Vehicle Dynamics* (3rd ed.). Butterworth–Heinemann. ISBN: 978-0-08-097016-5. Retrieved from https://www.sciencedirect.com/book/9780080970165/tire-and-vehicle-dynamics

[14] Kong, J., Pfeiffer, M., Schildbach, G., & Borrelli, F. (2015). *Kinematic and Dynamic Vehicle Models for Autonomous Driving Control Design*. In *Proceedings of the 2015 IEEE Intelligent Vehicles Symposium (IV)* (pp. 1094–1099). doi:10.1109/IVS.2015.7225830. Retrieved from https://ieeexplore.ieee.org/document/7225830

[15] Deo, N., & Trivedi, M. M. (2018). *Convolutional Social Pooling for Vehicle Trajectory Prediction*. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (pp. 1468–1476). doi:10.1109/CVPRW.2018.00196. Retrieved from https://ieeexplore.ieee.org/document/8575507

[16] Lefèvre, S., Laugier, C., & Ibanez-Guzman, J. (2014). *A Survey on Motion Prediction and Risk Assessment for Intelligent Vehicles*. Robomech Journal, 1(1), 1–14. doi:10.1186/s40648-014-0001-z. Retrieved from https://robomechjournal.springeropen.com/articles/10.1186/s40648-014-0001-z

[17] Doshi, A., & Trivedi, M. M. (2009). *On the Roles of Eye Gaze and Head Dynamics in Predicting Driver's Intent to Change Lanes*. IEEE Transactions on Intelligent Transportation Systems, 10(3). doi:10.1109/TITS.2009.2026675. Retrieved from https://ieeexplore.ieee.org/abstract/document/5173535

[18] Braunagel, C., Rosenstiel, W., & Kasneci, E. (2017). *Ready for Take-Over? A New Driver Assistance System for an Automated Class of Vehicles*. In *Proceedings of the 2017 IEEE Intelligent Vehicles Symposium (IV)* (pp. 1579–1586). doi:10.1109/IVS.2017.7995924. Retrieved from https://ieeexplore.ieee.org/document/7995924

[19] Endsley, M. R. (1995). *Toward a theory of situation awareness in dynamic systems*. Human Factors, 37(1), 32–64. Retrieved from https://www.researchgate.net/publication/210198492

[20] S. Vora, A. Rangesh, and M. M. Trivedi, "Driver Gaze Zone Estimation using Convolutional Neural Networks: A General Framework and Ablative Analysis," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 3, pp. 254–265, Sept. 2018. Retrieved from https://ieeexplore.ieee.org/document/8370646

[21] Tawari, A., & Trivedi, M. M. (2014). *Robust and continuous estimation of driver gaze zone by dynamic analysis of multiple face videos*. In *2014 IEEE Intelligent Vehicles Symposium (IVS)*, pp. 344–349. doi:10.1109/IVS.2014.6856607. Retrieved from https://ieeexplore.ieee.org/document/6856607

[22] Lefèvre, S., Vasquez, D., & Laugier, C. (2014). *A survey on motion prediction and risk assessment for intelligent vehicles*. ROBOMECH Journal, 1(1), 1. doi:10.1186/s40648-014-0001-z. Retrieved from https://inria.hal.science/hal-01053736

[23] Qu, F., Dang, N., Furht, B., & Nojoumian, M. (2024). *Comprehensive study of driver behavior monitoring systems using computer vision and machine learning techniques*. Journal of Big Data, 11(32). doi:10.1186/s40537-024-00890-0. Retrieved from https://doi.org/10.1186/s40537-024-00890-0

[24] European Parliament and Council. (2019). *Regulation (EU) 2019/2144 of 27 November 2019 on type-approval requirements for motor vehicles and their trailers, and systems, components and separate technical units intended for such vehicles, as regards their general safety and the protection of vehicle occupants and vulnerable road users*. Official Journal of the European Union, L 325, 1–40. Retrieved from https://eur-lex.europa.eu/eli/reg/2019/2144/oj/eng

[25] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). *Attention Is All You Need*. arXiv. Retrieved from https://arxiv.org/abs/1706.03762

[26] Postnikov, A., Gamayunov, A., & Ferrer, G. (2021). *Transformer-Based Trajectory Prediction*. arXiv preprint arXiv:2112.04350. Retrieved from https://arxiv.org/abs/2112.04350

[27] Lee, Y., Kim, H., & Oh, K. (2020). *The Importance of Prediction Horizon in Trajectory Prediction for Autonomous Driving*. arXiv preprint arXiv:2005.12872. Retrieved from https://arxiv.org/abs/2005.12872

[28] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv:1810.04805v2. Retrieved from https://arxiv.org/abs/1810.04805

[29] Baltrušaitis, T., Ahuja, C., & Morency, L.-P. (2017). *Multimodal Machine Learning: A Survey and Taxonomy*. arXiv preprint arXiv:1704.04394. Retrieved from https://arxiv.org/abs/1704.04394

[30] Muñoz Sánchez, M., van der Ploeg, C., Smit, R., Elfring, J., Silvas, E., & van de Molengraft, R. (2024). *Prediction Horizon Requirements for Automated Driving: Optimizing Safety, Comfort, and Efficiency*. arXiv. Retrieved from https://arxiv.org/abs/2402.03893

[31] Yang, Y., Negash, N., & Yang, J. (2025). *Influence of Prediction Horizon on Trajectory Optimization for Autonomous Vehicle Maneuvers*. SAE Technical Paper 2025-01-8309. Retrieved from https://saemobilus.sae.org/papers/influence-prediction-horizon-trajectory-optimization-autonomous-vehicle-maneuvers-2025-01-8309

[32] Lai, F., Liu, J., & Hu, Y. (2024). *An Automatic Emergency Braking Control Method for Improving Ride Comfort*. World Electric Vehicle Journal, 15(6), 259. Retrieved from https://www.mdpi.com/2032-6653/15/6/259

[33] MathWorks. (2026). *butter — Butterworth filter design*. Retrieved from https://se.mathworks.com/help/signal/ref/butter.html

[34] MathWorks. (2026). *filtfilt — Zero-phase digital filtering*. Retrieved from https://www.mathworks.com/help/signal/ref/filtfilt.html

[35] Dahl, J., Rodrigues de Campos, G., & Fredriksson, J. (2020). *A Path Prediction Model based on Multiple Time Series Analysis Tools used to Detect Unintended Lane Departures*. IEEE.

[36] Koppa, R. J. (1995). *Traffic Flow Theory: Human Factors*. Federal Highway Administration (FHWA). Retrieved from https://www.fhwa.dot.gov/publications/research/operations/tft/chap3.pdf

[37] Zhou, C., Zhang, F., Nacpil, E. J. C., Wang, Z., & Xu, F.-X. (2025). *Driver Steering Intention Prediction for Human-Machine Shared Systems of Intelligent Vehicles Based on CNN-GRU Network*. Sensors, 25(10), 3224. https://doi.org/10.3390/s25103224

[38] Raghu, M., Unterthiner, T., Kornblith, S., Zhang, H., & Dosovitskiy, A. (2021). *Do Vision Transformers See Like Convolutional Neural Networks?* Advances in Neural Information Processing Systems (NeurIPS). Retrieved from https://arxiv.org/abs/2108.08810

[39] Y. Xu, V. Letzelter, M. Chen, É. Zablocki, and M. Cord, "Annealed Winner-Takes-All for Motion Forecasting," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

[40] Waymo LLC. (2020). *Waymo Open Dataset*. GitHub repository. https://github.com/waymo-research/waymo-open-dataset

[41] Gustaver, M. (2020) A Chalmers University of Technology Master's thesis template for LaTeX. Unpublished.

A

Appendix 1

A.1 Description of AI-Generated Content

Artificial intelligence–based tools were used to support specific aspects of this thesis, primarily as writing and coding assistance, while all core ideas, methodology, analysis, and conclusions remain the authors' own. Copilot was used throughout the writing process to assist in improving clarity, structure, and linguistic quality of the text. In particular, Copilot was employed to revise and refine sentences and paragraphs to ensure grammatical correctness, readability, and overall coherence of the report. The generated suggestions were carefully reviewed by the authors, and only those aligned with the intended meaning and academic standards were incorporated.

In addition, Claude Opus was used as a coding assistant during the implementation phase of the project. Claude Opus contributed to generating and refining parts of the source code based on high-level instructions and comments provided by the authors. All generated code was reviewed, tested, and, where necessary, modified before being integrated into the final codebase to ensure correctness, relevance, and consistency with the project objectives.

No AI tool was used to autonomously generate scientific content, experimental results, analyses, or conclusions. The responsibility for the final content of the thesis, including all technical decisions and interpretations, lies solely with the authors.

A.2 Hyperparameters

Below in Table A.1 the hyperparameters used when training the GA-DSTP model are listed.

Table A.1: GA-DSTP model configuration and hyperparameters.

Hyperparameter	**Value**
Input Features	
Host input dimension	6 (4 raw + Δspeed, Δyawrate)
Driver input dimension	22 (actuator 3, turn indicator 3, DMS , Δsteering 1, gaze embed 8)
Gaze embedding	Embedding($11 \to 8$), quality-weighted
CNN Encoders (per stream)	
CNN channels / kernel	[64, 128, 128] / k=3 with residual skip
Positional encoding	Additive sinusoidal
Transformer Decoder	
d_{model} / heads / layers	128 / 8 / 4
Feed-forward dim (d_{ff})	512 (GELU)
Learned queries	$K \times S_{\text{out}} = 4 \times 8 = 32$
Mode embedding	4 per-mode identity biases
Time embedding	Linear($1 \to 128$)
Dropout	0.25
Output Heads	
Trajectory head	Linear($128 \to 2$) $\to (B, 6, 8, 2)$
Confidence head	$272 \to 128 \to 64 \to 1$, Log-Softmax over K
Trajectory Setup	
Trajectory modes (K)	4
Prediction horizon	3.5 s (8 waypoints)
Host / Driver history	1.0 s (10 steps) / 3.0 s (30 steps)
Sampling rate	10 Hz (downsampled from 50 Hz)
Stage 1 — Trajectory Training	
Epochs (warmup)	50 (5)
Optimizer / LR	AdamW / 2×10^{-4}
Loss	$\mathcal{L}_{\text{S1}} = \mathcal{L}_{\text{aWTA}} + \lambda_{\text{div}} \mathcal{L}_{\text{div}}$
τ_0 / α / λ_{FDE} / λ_{div}	2.0 / 0.95 / 2.0 / 0.5
Weight decay	1×10^{-4}
Stage 2 — Joint Fine-Tuning	
Epochs (warmup)	30 (3)
Backbone / Conf head LR	1×10^{-6} / 1×10^{-3}
Loss	WTA $+\lambda_{cls} \times$ KL(soft)
τ_{cls} / λ_{cls}	0.1 / 2.0
Weight decay	1×10^{-5}
General	
Batch size / Gradient clip	256 / max_norm = 1.0
Scheduler	Linear warmup + Cosine annealing

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY