

Article

Physics-Guided Multi-Modal Motion Prediction with Interaction-Aware GRU

Umut Özkan ^{1,*} , Ibraheem Shayea ^{2,*} , Leila Rzayeva ^{3,4} , Alisher Batkuldin ³  and Nursultan Nyssanov ^{3,4} 

¹ Control and Automation Engineering Department, Istanbul Technical University, Istanbul 34469, Türkiye

² Electrical Electronics & Communications Engineering Department, Faculty of Electrical and Electronics, Istanbul Technical University, Istanbul 34469, Türkiye

³ Research and Innovation Center “CyberTech”, Astana IT University, Astana 010000, Kazakhstan; l.rzayeva@astanait.edu.kz (L.R.); a.batkuldin@astanait.edu.kz (A.B.); nursultan.nyssanov@astanait.edu.kz (N.N.)

⁴ National Institute for Cyber Policy and Research, RSE on REM “Digital Government Support Center”, Astana 010000, Kazakhstan

* Correspondence: ozkanu23@itu.edu.tr (U.Ö.); shayea@itu.edu.tr (I.S.)

Abstract

In the Argoverse 2 experiments reported here, the simplest Constant Turn Rate and Acceleration (CTRA) decoder was stable but missed many interaction-driven turns and merges, while residual decoders without enough control improved early displacement but increased final-horizon error. This paper therefore studies a compact decoder in which each of the six futures is represented as a CTRA anchor plus an autoregressive position residual. The residual gated recurrent unit (GRU) is initialized from fused target-history, top- k neighbor, and lane-polyline context, and its contribution is scaled by a mode-specific gate and learned exponential decay. On the 10k/2k sanity ablations, CTRA-only decoding reached $\text{minFDE}_6 = 7.189$ m, while autoregressive residuals with a larger correction GRU reduced it to 4.157 m; removing the gate increased it again to 4.946 m. On the full Argoverse 2 validation split, the final configuration achieves a minimum average displacement error of $\text{minADE}_6 = 1.21$ m and a minimum final displacement error of $\text{minFDE}_6 = 2.78$ m. The reported diagnostics show that the compact model generates a useful six-mode set, but still needs better probability ranking for top-1 selection.

Keywords: trajectory prediction; physics-guided deep learning; CTRA; interaction modeling; multi-modal prediction; autonomous driving



Academic Editor: Juan Gabriel Avina-Cervantes

Received: 13 April 2026

Revised: 23 June 2026

Accepted: 25 June 2026

Published: 15 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

The experiment in this paper starts from a practical forecasting interface: for each Argoverse 2 focal agent, the model receives 5 s of history and must output 6 s of future occupancy at 10 Hz. A single observed history can still support lane keeping, yielding, a turn, or a merge, so the implemented decoder returns $K = 6$ trajectories rather than one averaged path [1–3]. The validation exports also made a second issue visible: a low oracle displacement error is not enough if the best trajectory is not ranked first, or if residual corrections drift away from the lane geometry near the end of the horizon.

For this model, the main tension appeared inside the decoder. Classical kinematic models such as Constant Velocity (CV), Constant Turn Rate and Velocity (CTRV), and CTRA give smooth and interpretable rollouts, but the later ablation results show that the CTRA-only decoder has much larger final error than the residual variants. Direct learned

correction is also not free: the no-gate ablation reported later shows that a larger residual GRU can worsen final displacement when its contribution is not controlled. The design therefore starts from the kinematic branch and adds a learned correction only through an explicit gate/decay term.

The decoder therefore represents each future mode as *CTRA rollout + gated autoregressive residual*. The CTRA part is initialized from the last observed state and mode-specific velocity/yaw-rate gains. The residual part is produced by a compact GRU that receives the evolving predicted state, not only the fixed scene embedding. A mode gate and learned exponential decay scale the residual over time, so short-term corrections can be large when the CTRA prior is locally wrong, while later timesteps remain tied to the physics anchor.

The decoder design was chosen after several compact sanity runs rather than from the benchmark result alone. The CTRA-only variant produced smooth trajectories but high final error, especially in turning and lane-change cases. Non-autoregressive residuals improved displacement but did not use the evolving predicted state. The autoregressive residual decoder improved the 10k/2k validation metrics, and the no-gate ablation showed why residual strength should be controlled instead of left unconstrained.

Evaluation uses the Argoverse 2 Motion Forecasting protocol: a 5 s observed history, a 6 s prediction horizon at 10 Hz, and $K = 6$ trajectory hypotheses [4]. The encoder uses three bounded context streams that match the exported tensors: a GRU over 50 target-history states, attention over the top- $k = 4$ neighboring tracks, and pooling over lane polylines inside a 50 m local radius. This bounded context is a deliberate part of the experiment, because the paper is testing how far a compact physics-guided decoder can go before richer topology modeling becomes necessary.

The paper makes three contributions tied to those runs. First, it introduces a CTRA-residual decoder whose correction GRU is conditioned on the evolving predicted state. Second, it regulates those residuals with mode-specific gates and learned decay, and the ablation study reports the cost of removing that control. Third, it reports mechanism-focused validation results, including oracle displacement, top-1 ranking behavior, scenario tags, latency, and negative results for discarded variants.

2. Background and Related Work

In the experiments below, the prediction module receives tracked histories and returns the six-mode forecast that a planner would later consume [1]. The related work is used as a design map rather than as a broad survey. Each group below is connected to one implementation decision in the compact Argoverse 2 model: the CTRA anchor, the GRU target-history encoder, the top- k neighbor slots, local lane pooling, six-mode decoding, or the decision not to model V2X communication failures in the current experiment [2,3,5].

2.1. Evolution of Prediction Paradigms

Rather than following a single linear progression, the trajectory prediction literature can be grouped into a few recurring modeling families. The relevant question for this paper is what each family contributed to the final implementation, and what was intentionally left out to keep the model small enough for the sanity ablations and latency measurements reported later [6,7].

2.1.1. Physics-Based Models

A common baseline in trajectory prediction is to assume a simple kinematic motion model, such as CV, CTRV, or bicycle dynamics, and to couple it with filtering for state estimation and short-horizon extrapolation [5,8]. These models provide an explicit structure that generally yields physically plausible rollouts and can be evaluated with low compu-

tational overhead. The limitation is largely model mismatch: when acceleration changes, curvature varies, or behavior is driven by interactions, the fixed form of a single kinematic model may accumulate error over the forecast horizon. Multiple-model filtering addresses part of this issue by maintaining several motion hypotheses and switching between them, and has also been extended with interaction-aware formulations [9]. This is why CTRA is used here only as the first branch of the decoder, not as the complete predictor. The ablation results reported later show the same point numerically: the CTRA-only decoder is stable but has $\text{minFDE}_6 = 7.189$ m on the 10k/2k split, far above the residual variants.

2.1.2. Learning-Based Models

Learning-based predictors treat motion forecasting as a sequence modeling problem and learn common behaviors directly from data. Recurrent models such as LSTMs and GRUs are frequently used to encode motion history and have been applied across different agent types and traffic settings [10,11]. Neighbor interactions are often incorporated by aggregating information from nearby agents, either with pooling schemes or with attention that assigns different weights to different neighbors; interaction-aware LSTM variants are typical examples of this direction [12,13]. Another widely used formulation builds an explicit relational structure by treating agents as nodes and their relations as edges, which supports heterogeneous interaction types and changing neighborhoods over time [14–17]. More recently, Transformer-style attention has been applied to fuse agent and map context within a single framework, especially in settings where richer context integration and longer temporal/spatial dependencies are beneficial [18–20]. Motion Transformer (MTR), for example, combines global intention localization with local movement refinement using learnable motion queries [21], while the convolutional-transformer trajectory prediction model (CMTT) uses convolutional feature extraction together with encoder–decoder attention for urban traffic prediction [22]. The proposed model borrows the compact temporal-memory idea from recurrent predictors, but it does not use a full scene Transformer. The top- $k = 4$ neighbor slots and 50 m lane radius were chosen to keep the context encoder bounded, even though the scenario-level diagnostics reported later show that this choice leaves route-intent errors in turning and merge cases.

2.1.3. Hybrid Physics-Guided Models

Hybrid formulations introduce a kinematic prior into learning-based predictors to stabilize rollouts and reduce physically implausible outputs, while using learned components to account for context-dependent deviations and interaction-driven behavior. One representative direction predicts control or latent actions and converts them into trajectories through a vehicle model, for example using mixture-based control prediction followed by bicycle-model integration [23]. Other work enforces physical consistency while modeling interactions explicitly in structured settings such as platooning [24]. Kinematics-aware graph models also combine residual learning with structured interaction reasoning, as in the kinematics-aware multigraph attention network (KA-MGAT) for heterogeneous traffic agents [25]. In complex urban scenes, hybrid designs have also been combined with richer scene encoders and trajectory parameterizations to improve feasibility at intersections and merges [26].

KA-MGAT is especially relevant because it also combines kinematic awareness, graph attention, and residual learning [25]. The difference is in where the physics prior is placed. KA-MGAT uses multigraph attention to represent heterogeneous agent interactions and then learns residual corrections inside a graph-based interaction model. The present model instead uses a compact top- k neighbor encoder and places the main physics constraint inside the decoder: each mode is first rolled out as a CTRA anchor and only then corrected

by a gated, time-decayed autoregressive residual. This makes the proposed method weaker than graph-based topology/heterogeneity models for complex interaction structure, but it gives a direct mechanism for limiting how much a residual decoder can move away from the kinematic anchor. Compared with bicycle-model or control-space hybrids, this implementation keeps the residual in position space because the negative-results diagnostics reported later showed less stable behavior for the control-space residual on the 10k/2k sanity split. The physics prior is therefore used as a coordinate anchor, while the learned component is a bounded position correction.

2.2. Context Encoding: Interactions and Environment

Motion forecasting is rarely determined by the target agent's past motion alone. In the exported validation cases, this was most visible in two places: dense-neighbor scenes still needed local interaction cues, and turns/merges needed lane context even when the last observed motion looked nearly straight. The current model uses only the parts of interaction and map encoding that fit the compact setup: four neighbor slots, masks for missing neighbors, and pooled lane polylines around the focal agent.

2.2.1. Social Interaction Modeling

Early interaction modules in trajectory predictors often compress neighbor motion into a single pooled representation. This keeps the model simple, but it can blur which neighbor matters and when, especially in dense scenes [12]. Graph-based formulations address this by keeping an explicit relational structure: agents are treated as nodes and interactions are modeled through edges, so relations can change over time instead of being averaged away. Multiscale spatial-temporal graph designs have been used to capture interaction patterns at different time scales [27], and traffic-state information has been incorporated to better reflect interactive driving conditions [28]. Diffusion-style graph models follow a similar motivation while emphasizing multi-modal generation and interaction-consistent futures [29]. In parallel, some recent approaches incorporate risk cues directly into the representation, for example by using collision probability fields for longitudinal prediction [30] or by focusing on high-risk interaction cases with adversarial training [31]. The implementation here does not build a full interaction graph; it keeps the closest four neighbors and applies masked attention over those slots. The scenario-level diagnostics reported later suggest that this local encoder is useful in dense-neighbor cases, where the error increase is smaller than in route-change cases.

2.2.2. Map and Lane Context Integration

Map information helps keep forecasts on drivable structure and can reduce ambiguity near intersections, merges, and lane changes. Many methods encode lanes as polylines or lane graphs and fuse these features with agent motion, so that predicted trajectories stay consistent with lane geometry and local topology. Generative formulations also use map cues to shape the latent space, explicitly tying predicted motion to lane-related structure and reducing off-road behavior [32]. Dynamic graph constructions that adapt the represented scene size are another practical option when both local lane detail and broader context are needed [33]. Risk-aware encoders and structured decoders further highlight that lane context becomes especially important in complex junction geometry where multiple maneuvers remain plausible over the same history [26]. In the present implementation, map context is weaker than a lane graph: nearby lane points are pooled after a 50 m radius query. The turning and lane-change/merge diagnostics reported later are therefore interpreted as evidence that the next version needs explicit lane connectivity, not only more local lane points.

2.2.3. Vehicle-to-Everything and Vulnerable-Road-User Safety

Vehicle-to-everything (V2X) communication is often discussed in the context of improving awareness and coordination beyond line-of-sight sensing, especially for vulnerable road users (VRUs). Reviews of V2X for road-user safety emphasize that VRU localization, message reliability, and coverage limitations remain practical bottlenecks compared with vehicle-to-vehicle exchanges [7]. On the planning side, V2X has also been used to formulate multi-agent objectives such as interaction fairness and efficiency under partial decentralization [34]. The current experiments do not include packet age, message delay, link reliability, or cooperative-perception confidence. The V2X discussion is included to define the deployment scenario in which the same state variables could be received from shared messages, while the evaluated model itself starts after a cleaned Argoverse 2 track tensor has already been formed. Table 1 summarizes representative trajectory prediction approaches and indicates whether each method uses temporal, social, map, physics-prior, and multi-modal components.

Table 1. Comparison of Representative Trajectory Prediction Approaches.

Method	Year	Temporal Encoder	Social Aware	Map Aware	Physics Prior	Multi-Modal
KF-CTRV [5]	2022	–	–	–	✓	–
IA-LSTM [12]	2024	LSTM	✓	–	–	–
IMA-LSTM [13]	2024	LSTM	✓	–	–	–
HeteroEdge-GAT [14]	2022	–	✓	✓	–	–
ST-GCN [15]	2022	–	✓	–	–	–
Dual-Branch GNN [16]	2023	–	✓	✓	–	–
Diffusion GCN [29]	2024	–	✓	–	–	✓
ST Graph Attention Transformer [18]	2022	Transformer	✓	(✓)	–	–
MTR [21]	2022	Transformer	✓	✓	–	✓
HDGT [19]	2023	Transformer	✓	✓	–	–
KA-MGAT [25]	2024	GAT	✓	–	✓	✓
HATN [20]	2025	Transformer	✓	✓	(✓)	✓
CVAE (Multi-task) [32]	2025	LSTM	✓	✓	–	✓
MTGNet [33]	2025	Transformer	✓	✓	–	✓
CVAE+Bicycle [23]	2024	LSTM	✓	✓	✓	✓
HRGC [26]	2025	Transformer	✓	✓	✓	✓
CMTT [22]	2026	Transformer	✓	✓	–	✓
This Work	2026	GRU	✓	✓	CTRA+Gate	✓

Note: Checkmarks indicate explicit inclusion of the corresponding component; parenthesized checkmarks indicate partial or indirect use. Table abbreviations include Kalman filter (KF), graph attention network (GAT), graph convolutional network (GCN), spatial-temporal (ST), high-definition (HD), heterogeneous driving graph transformer (HDGT), and hybrid attention transformer network (HATN).

2.3. Multi-Modal Prediction and Uncertainty

Urban driving is inherently ambiguous: the same motion history can be consistent with lane keeping, lane changes, yielding, or turning, depending on local geometry and surrounding traffic. In this model the ambiguity is represented by six mode-specific CTRA branches, six residual sequences, and six scores. The validation diagnostics later separate two questions that are often mixed together: whether one of the six paths is close to the ground truth, and whether the scoring head assigns that path the highest probability.

Conditional variational formulations are a standard tool for producing diverse yet plausible multi-future rollouts. Conditional variational autoencoder (CVAE)-based predictors have been used to model intent and uncertainty while maintaining realism in generated trajectories [35]. Recurrent generative variants extend this idea by conditioning on attention-refined interaction context [36], and goal-conditioned predictors explicitly represent destination uncertainty before decoding trajectories [37]. Beyond generating

multiple modes, confidence quality matters in practice: planners benefit when mode probabilities reflect true likelihoods rather than overconfident rankings, which has motivated work that connects prediction outputs to planning performance through probability calibration and interaction-aware decision making [38]. The diagnostics reported later show why this distinction matters here: the proposed model has low oracle displacement but much worse ADE_1/FDE_1 , so the next experiment should improve the scoring head rather than only adding more trajectory capacity.

2.4. Joint Prediction and Planning

TOFG and related prediction–planning work are cited here mainly to define the boundary of the present experiment [39]. The model in this paper stops at the forecasting interface: it exports six 60-step trajectories, six mode scores, and auxiliary uncertainty values. No closed-loop planner is trained, no imitation objective is used, and the validation tables do not measure planning cost. This matters for interpreting the results. A planner that consumes the whole six-mode set may still benefit from the low oracle displacement error reported later, while a planner that selects only the maximum-probability mode would see the larger ADE_1/FDE_1 error.

2.5. Positioning of This Work

Figure 1 should be read as the version that survived the compact ablations, not as a full scene-forecasting stack. The model has no lane graph, no global scene Transformer, no closed-loop planner, and no communication-quality input. It keeps three bounded inputs from the loader: the target history, four masked neighbor slots, and nearby lane polylines. The decoder is where the main claim sits. The later ablations show why the final version is not CTRA-only, not control-space residual, and not no-gate residual correction.

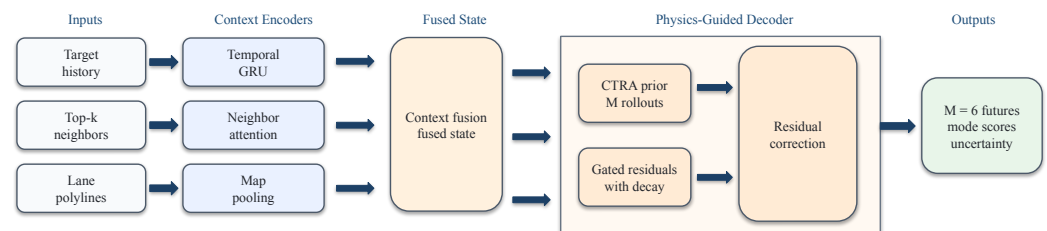


Figure 1. Architecture of the final compact CTRA–residual predictor. The model receives target history, four masked neighbor slots, and local lane polylines; the decoder rolls out six CTRA anchors, applies gated time-decayed autoregressive residual corrections, and outputs mode scores.

The comparison with larger topology-heavy predictors is therefore intentionally limited. This model tests whether a small physics-guided decoder can produce a useful six-mode set before adding richer map topology. The answer from the validation export is mixed: $\min ADE/\min FDE$ are competitive with early Argoverse 2 baselines, but the diagnostic results show that mode ranking is still weak. For that reason, the scoring head is treated as part of the method rather than as a minor output layer.

3. Vehicle Trajectory and Direction

This section fixes the state and scope assumptions used in the rest of the paper. In the implemented loader, each focal sample is converted into three local inputs before batching: a target-history tensor with 50 observed states, a masked top- k neighbor tensor with $k = 4$, and a set of lane polylines collected inside a 50 m radius. All three inputs are expressed in the same agent-centric frame defined by the final observed target pose. The V2V/V2X

framing is therefore used as a deployment context for where these state variables may come from, not as a claim that the experiments simulate a communication protocol.

3.1. Vehicle Trajectory

In this work, a trajectory is the sampled state history of one tracked road user after preprocessing. For the focal agent, the stored state is position, velocity, and heading; the heading is later represented by $\sin \psi$ and $\cos \psi$ in the temporal encoder. The last observed focal pose defines the local coordinate frame. Target history, neighbor histories, and lane polylines are translated and rotated by the same pose, so the GRU input, neighbor slots, and map features all refer to the same coordinate system. Figure 2 illustrates this V2V/V2X-oriented interface.

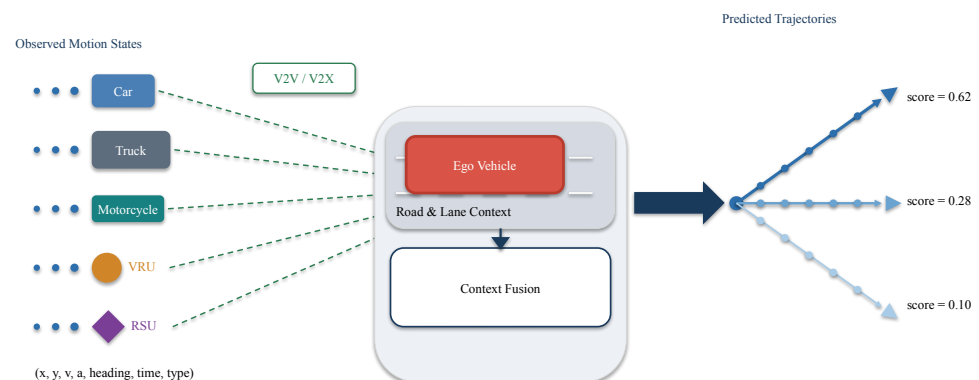


Figure 2. V2V/V2X-oriented forecasting interface used to explain the input scope. Colored blocks denote heterogeneous agent and context inputs, and arrows indicate the fusion path from observed states and road/lane context to ego-centered trajectory prediction. Observed motion states from heterogeneous agents and road/lane context are fused at the ego vehicle before the predictor outputs candidate trajectories with mode scores.

Two groups of variables are deliberately absent from this tensor. The first group belongs to communication or perception quality: packet age, sensor source, covariance, synchronization error, and message reliability. The second group belongs to richer map reasoning: lane successor links, traffic-light state, and route hypotheses. The experiments below do not test those variables. They test the narrower Argoverse 2 forecasting case after the track and lane tensors have already been cleaned and transformed into the focal-agent frame.

3.2. Prediction of Vehicle Trajectory

Given $T_h = 50$ observed timesteps, the model predicts $M = 6$ future trajectories over $T_f = 60$ timesteps and assigns one probability to each mode. This is not only a formatting choice for the benchmark. In the decoder, each mode has its own CTRA gains, yaw bias, residual state, gate/decay values, and confidence logit. The six outputs therefore represent six separate rollouts from the same fused context rather than post-processed perturbations of a single path. Figure 3 summarizes the upstream data interface assumed by this study.

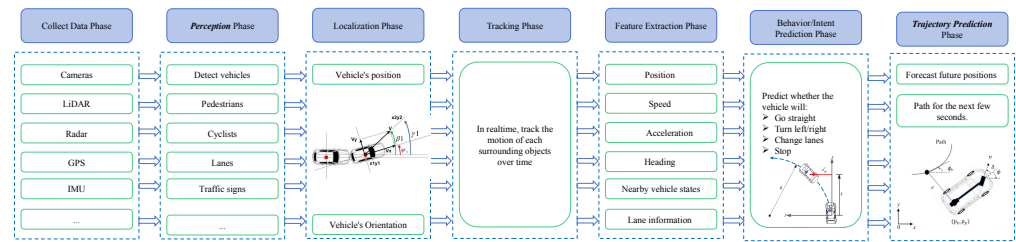


Figure 3. Data interface assumed in this study. Perception, localization, tracking, feature extraction, and behavior/intent estimation are treated as upstream stages; the trajectory predictor starts after target history, masked neighbor states, and lane context have already been converted into the target-centric frame. Ellipses denote omitted intermediate upstream processing steps and do not change the predictor inputs used in this study.

3.3. Challenges of Predicting Vehicle Trajectory

Three practical issues drove the final architecture. First, the history tensor can contain missing or weak neighbor information, so the social encoder uses a fixed top- k slot representation with masks rather than assuming a dense set of valid agents. Second, route intent is ambiguous at intersections and merges; lane context and multi-modal decoding are therefore required even when the last few frames look nearly straight. Third, learned residuals can accumulate over a 60-step horizon. The CTRA anchor, residual gate, and decay term were added specifically to limit this failure mode, and the no-gate ablation reported later is the direct check for that choice.

The boxes in Figure 4 correspond directly to implementation choices. Missing neighbor information is handled by masks in the top- k slots. Intent ambiguity is handled by six mode-specific CTRA branches. Map topology is represented only through nearby lane polylines, which explains the higher error in turning and lane-change/merge cases in the scenario diagnostics reported later. Long-horizon residual drift is addressed by the gate/decay term, while the latency constraint is reported separately in the efficiency analysis.

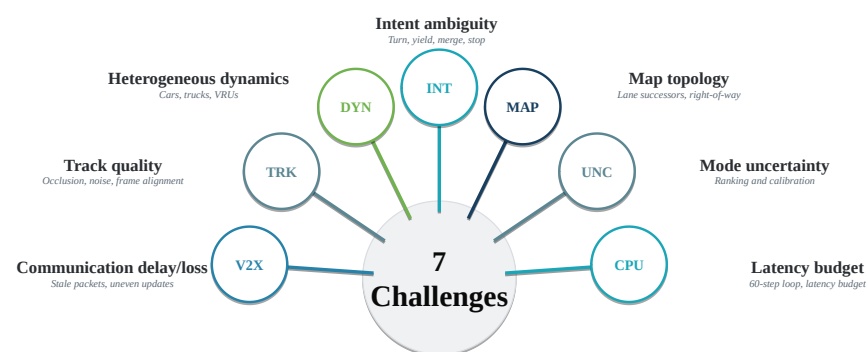


Figure 4. Seven trajectory-prediction challenges represented in the revised circular graphic: communication delay/loss, track quality, heterogeneous dynamics, intent ambiguity, map topology, mode uncertainty, and latency budget.

4. Algorithm Selection and Development

The architecture was chosen as a pruning result from compact development runs. A variant was kept only if it satisfied four checks: it had to roll out all 60 future steps without obvious drift, keep recurrent state small enough for the compact setting, use a bounded number of neighboring agents, and preserve six distinct futures. These checks were applied before the full validation run, mostly on 10k/2k sanity splits. The section therefore records the design path that led to the submitted model; it is not a claim that every graph or Transformer architecture was exhaustively searched.

4.1. Investigated Algorithms

Figure 5 records the components that remained after this pruning. CV, CTRV, and CTRA models were useful as rollout anchors, but the later ablation results show that the anchor is too rigid by itself. GRUs were kept for the fixed 50-step target history because they gave a compact temporal state and did not require the larger recurrent state of a Long Short-Term Memory (LSTM) network. Attention was kept local rather than global: the model attends over four neighbor slots and pools lane points from the 50 m query. This choice is also a limitation, since the scenario diagnostics reported later show large errors in turns and lane-change/merge cases.

Item	Physics-based kinematic models	Recurrent neural networks (RNNs)	Attention-based interaction models
Basic concept	Constant Velocity (CV) Constant Turn Rate and Velocity (CTRV) rollout	Long Short-Term Memory (LSTM) and gated recurrent unit (GRU) memory over motion history	Attention weights neighbors and lanes
Main advantage	Fast, feasible, interpretable	Compact temporal representation	Focus on relevant interaction context
Main limitation	Weak intent and interaction modeling	Needs scene and interaction context	Costly when used as global attention
Used in this work	Physics anchor	GRU temporal backbone	Lightweight context encoder

Figure 5. Table-style comparison of the investigated algorithm families. The final model keeps the physics-based family as the rollout anchor, the recurrent family as the GRU temporal backbone, and the attention family as a lightweight context encoder.

4.2. Selected and Developed Methods

The selected architecture uses a two-layer GRU over $(x, y, v_x, v_y, \sin \psi, \cos \psi)$ target-history features. The trigonometric heading representation avoids the discontinuity at $\pm\pi$, which matters because the hidden state is later used to initialize mode-specific CTRA parameters. The PhysicsGuidedDecoder then rolls out six CTRA anchors, one per mode, and refines them with autoregressive position residuals. The residual GRU is conditioned on the current predicted position and a learned time embedding, so the correction at timestep t depends on the state produced by earlier corrections rather than only on the original scene embedding.

The gate/decay branch stayed in the model because the no-gate run failed a direct ablation check. With the same larger correction GRU, removing the gate changes minADE₆ from 1.824 m to 2.121 m and minFDE₆ from 4.157 m to 4.946 m in the later ablation table. In the final decoder, the residual is therefore not allowed to act as a second free trajectory generator. It is a position correction attached to the CTRA rollout, scaled by a mode gate and damped over the 60-step horizon. Figure 6 summarizes the selected hybrid prediction pipeline.

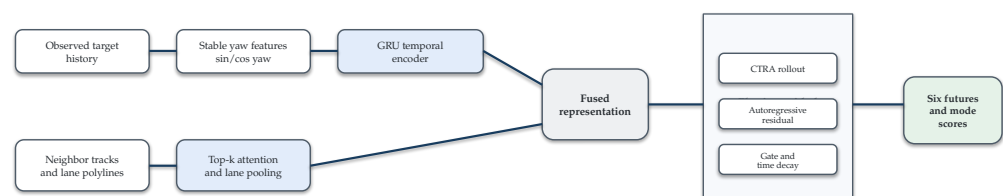


Figure 6. Selected hybrid prediction pipeline after the sanity ablations. The upper branch encodes target history, the lower branch encodes neighbor/lane context, and the fused representation drives the CTRA-residual decoder.

Figure 7 separates the implementation enhancements from the main architectural flow.

Developed enhancements added beyond the investigated baseline families				
E1: Yaw discontinuity handled by sin/cos	E2: Mode-specific CTRA parameters	E3: Residuals conditioned on predicted state	E4: Per-mode gate and learned decay	E5: Lightweight neighbor and lane context

Figure 7. Developed enhancements added beyond the investigated baseline families. These items are separated from the pipeline figure to distinguish architectural flow from implementation changes.

5. Research Methodology

All reported numbers start from the same Argoverse 2 forecasting interface: 50 observed target states, four masked neighbor slots, lane polylines from a 50 m local query, and a 60-step future target. This section documents the exact preprocessing and model interface used for those numbers. Perception, tracking, communication delay, and V2X message quality are outside the evaluated module; they are upstream of the tensor passed to the forecaster.

5.1. Dataset Description

Argoverse 2 is used here because its forecasting split matches the intended stress test: fixed 5 s histories, 6 s futures, multi-agent urban scenes, and lane-level map context [4]. The benchmark also makes the compact-model question measurable, because the same validation split contains easy lane-following cases and harder turn/merge cases.

5.1.1. Dataset Overview

Argoverse 2 motion forecasting sequences are sampled at 10 Hz and span 11 s in total. The protocol provides 5 s of observed history and requires forecasting the next 6 s, corresponding to $T_h = 50$ and $T_f = 60$ timesteps. The benchmark contains 199,908 training sequences, 24,988 validation sequences, and 24,984 test sequences [4]. Those sizes matter here because the full validation export is large enough to separate lane-following, turning, lane-change/merge, and dense-neighbor cases without changing the model between subsets.

The dataset is collected across six U.S. cities and includes intersections, merges, and lane-change contexts [4]. In this manuscript, those scene types are not used as labels during training; they are used after validation to diagnose where the compact CTRA-residual decoder succeeds and where it misses route intent.

Coordinate Frame

For every focal sample, the last observed target pose defines the local frame. Target states, neighbor states, and lane points are translated by the same reference position and rotated by the negative reference yaw. Velocities are rotated but not translated. The reported displacement metrics are then computed in this local meter-scale frame. This normalization is important for the decoder: the CTRA branch starts from the final observed target state, and the residual GRU predicts corrections in the same coordinate system.

5.1.2. Dataset Splits and Statistics

Table 2 reports the number of sequences per split under the standard benchmark partition.

During early development iterations, experiments used smaller random subsets to reduce turnaround time. A typical setup used 20,000 training sequences and 2000 validation sequences sampled from the official splits.

Table 2. Argoverse 2 Motion Forecasting split sizes (number of sequences).

Split	Sequences
Training	199,908
Validation	24,988
Test	24,984

5.1.3. Data Preprocessing

Each agent state at timestep t is represented as a 5D vector,

$$\mathbf{s}_t = [x_t, y_t, v_{x,t}, v_{y,t}, \psi_t]^\top, \quad (1)$$

where (x_t, y_t) denotes 2D position in the agent-centric frame after normalization, $(v_{x,t}, v_{y,t})$ denotes velocity components, and ψ_t denotes heading.

The temporal window follows the benchmark protocol with a 5-s history window ($T_h = 50$ at 10 Hz) and a 6-s prediction horizon ($T_f = 60$ at 10 Hz), using $\Delta t = 0.1$ s.

Neighbor agents are selected through a proximity filter. For a focal agent i , the neighbor set is defined as

$$\mathcal{N}_i = \{j \mid \|\mathbf{p}_i - \mathbf{p}_j\| \leq r_{\max}, j \neq i\}, \quad (2)$$

with $r_{\max} = 30$ m. The $k = 4$ closest neighbors are kept based on minimum distance over the history window, with masks used when fewer than k valid neighbors are available.

Map context is extracted by collecting lane centerlines within a fixed radius (50 m) around the focal agent. Each lane is represented as a polyline of waypoints with basic geometric features such as position and local heading, which can then be pooled or attended over by the map encoder.

5.1.4. Track Quality and Filtering

Argoverse 2 groups tracks by quality and evaluation relevance. Forecasting targets are defined by FOCAL_TRACK agents, while surrounding context can include SCORED_TRACK and UNSCORED_TRACK agents. TRACK_FRAGMENT entries correspond to partial or low-quality tracks and are excluded to avoid introducing noisy motion histories into the training data [4].

5.1.5. Data Augmentation

To improve robustness, lightweight augmentations are applied during training while preserving scene consistency. The current setup adds small Gaussian noise to positions ($\sigma = 0.05$ m) and applies a random rotation sampled from $[-0.1, 0.1]$ radians (approximately $\pm 5.7^\circ$). Positions and velocities are rotated consistently, and headings are updated accordingly.

5.2. Model Architecture

The model follows the encoder-decoder structure in Figure 1. Temporal motion history, neighbor context, and map context are encoded by separate modules and then fused into a single representation that drives a physics-guided, multi-modal decoder.

5.2.1. Temporal Encoder: PhysicsAwareGRU

The temporal encoder processes the focal agent history with a two-layer GRU [40]. To avoid discontinuities in angular representation, heading is embedded using trigonometric components:

$$\mathbf{x}_t = [x_t, y_t, v_{x,t}, v_{y,t}, \sin(\psi_t), \cos(\psi_t)]^\top \in \mathbb{R}^6. \quad (3)$$

The recurrent update is

$$\mathbf{h}_t = \text{GRU}(\mathbf{x}_t, \mathbf{h}_{t-1}), \quad t = 1, \dots, T_h. \quad (4)$$

Layer normalization and dropout ($p = 0.1$) are applied to stabilize optimization. The temporal summary uses the final hidden state of the GRU:

$$\mathbf{h}_{\text{temp}} = \mathbf{h}_{T_h}. \quad (5)$$

5.2.2. Social Context Encoder: NeighborEncoder

Neighbor context is represented using the top- k neighbor slots. For each neighbor j , the feature vector includes relative motion cues and geometric descriptors:

$$\mathbf{f}_j = [\Delta x_j, \Delta y_j, \Delta v_{x,j}, \Delta v_{y,j}, \Delta \psi_j, \theta_j, \Delta s_j, d_j]^\top. \quad (6)$$

Here θ_j is the relative bearing, Δs_j is speed difference, and d_j is the distance at the last observed timestep. The features are embedded with a linear projection,

$$\mathbf{e}_j = \text{Linear}(\mathbf{f}_j), \quad (7)$$

and refined using multi-head attention (MHA) over the slot set:

$$\mathbf{e}'_j = \text{MHA}(\mathbf{e}_j, \{\mathbf{e}_k\}_{k \in \mathcal{N}_i}). \quad (8)$$

A masked mean pooling aggregates valid slots into a single social vector:

$$\mathbf{h}_{\text{social}} = \frac{\sum_{j=1}^k m_j \mathbf{e}'_j}{\sum_{j=1}^k m_j}, \quad (9)$$

where $m_j \in \{0, 1\}$ indicates whether slot j contains a valid neighbor.

5.2.3. Environmental Context Encoder: MapEncoder

Map context is built from nearby lane centerlines represented as waypoint sequences. Each lane waypoint is encoded with a 7-D feature vector (position, heading, and lane attributes):

$$\mathbf{l}_i \in \mathbb{R}^7. \quad (10)$$

A two-layer multi-layer perceptron (MLP) produces lane point embeddings:

$$\mathbf{g}_i = \text{MLP}(\mathbf{l}_i), \quad (11)$$

and attention pooling forms the final map vector:

$$\beta_i = \frac{\exp(\mathbf{u}^\top \mathbf{g}_i)}{\sum_{i'} \exp(\mathbf{u}^\top \mathbf{g}_{i'})}, \quad \mathbf{h}_{\text{map}} = \sum_i \beta_i \mathbf{g}_i. \quad (12)$$

5.2.4. Context Fusion

Temporal, social, and map representations are concatenated (map optional) and passed through a fusion layer with LayerNorm and ReLU:

$$\mathbf{h}_{\text{fused}} = \text{ReLU}\left(\text{LayerNorm}\left(\mathbf{W}_f[\mathbf{h}_{\text{temp}}; \mathbf{h}_{\text{social}}; \mathbf{h}_{\text{map}}]\right)\right). \quad (13)$$

5.2.5. Physics-Guided Decoder

The decoder outputs $M = 6$ trajectory hypotheses. For each mode m , a physics head predicts mode-specific gains and acceleration:

$$g_m^v, g_m^\omega, a_m = \text{MLP}_{\text{phys}}(\mathbf{h}_{\text{fused}}), \quad (14)$$

and a separate head predicts a yaw bias $b_m = \text{MLP}_{\text{bias}}(\mathbf{h}_{\text{fused}})$. The initial kinematic parameters are adjusted as

$$v_0^m = v_0(1 + g_m^v), \quad \omega_0^m = \omega_0(1 + g_m^\omega) + b_m, \quad (15)$$

with gains bounded by a tanh nonlinearity.

Given the initial state, the baseline prediction $\hat{\tau}_{m,1:T_f}^{\text{CTRA}}$ is generated by a CTRA integration with piecewise handling of the near-straight regime ($|\omega| \approx 0$) and the curved regime:

$$\hat{x}_{t+1} = \hat{x}_t + \int_t^{t+\Delta t} v(\tau) \cos(\psi(\tau)) d\tau, \quad (16)$$

$$\hat{y}_{t+1} = \hat{y}_t + \int_t^{t+\Delta t} v(\tau) \sin(\psi(\tau)) d\tau, \quad (17)$$

$$v(t + \Delta t) = v(t) + a\Delta t, \quad (18)$$

$$\psi(t + \Delta t) = \psi(t) + \omega\Delta t. \quad (19)$$

Autoregressive Residuals

To model deviations from rigid kinematics, time-varying residuals are predicted with a lightweight GRU. A learned time embedding $e_t = \text{Emb}(t)$ is concatenated with the current predicted position, and the GRU state is initialized from the fused context:

$$\mathbf{h}_0^m = \text{MLP}_{\text{init}}(\mathbf{h}_{\text{fused}}), \quad (20)$$

$$\mathbf{h}_t^m = \text{GRU}([e_t, \hat{\mathbf{x}}_t^m], \mathbf{h}_{t-1}^m), \quad (21)$$

$$\mathbf{r}_t^m = \text{MLP}_{\text{out}}(\mathbf{h}_t^m) \in \mathbb{R}^2. \quad (22)$$

A per-mode gate g_m and decay base d_m are predicted from the context:

$$g_m = \sigma(\text{MLP}_{\text{gate}}(\mathbf{h}_{\text{fused}})), \quad d_m = \sigma(\text{MLP}_{\text{decay}}(\mathbf{h}_{\text{fused}})). \quad (23)$$

Residuals are injected in position space after the CTRA step:

$$\boldsymbol{\tau}_{m,t} = \hat{\boldsymbol{\tau}}_{m,t}^{\text{CTRA}} + g_m d_m^t \mathbf{r}_t^m, \quad t = 1, \dots, T_f. \quad (24)$$

This construction keeps long-horizon behavior close to the kinematic prior while allowing short-horizon corrections that depend on the evolving predicted state. The decay term also provides a simple bounded-correction interpretation. If $\|\mathbf{r}_t^m\|_2 \leq R_m$, $0 \leq g_m \leq 1$, and $0 < d_m < 1$, then the injected correction at step t is bounded by $g_m d_m^t R_m$, and the cumulative residual magnitude over a horizon is bounded by $g_m R_m d_m(1 - d_m^{T_f}) / (1 - d_m)$. Without decay, the same worst-case bound grows linearly with T_f . The mechanism therefore does not impose a hard vehicle-dynamics constraint, but it regularizes how much the learned component can dominate the CTRA anchor as the horizon increases.

CTRA is selected because it is low-dimensional, differentiable, and compatible with the tracked centroid states available in Argoverse 2. It also covers the most common local behaviors needed for short-to-medium horizon extrapolation: straight motion, acceleration, and smooth turning. However, it is intentionally simpler than bicycle, tire-force, or car-

following models such as the Intelligent Driver Model (IDM). It does not model steering dynamics, tire saturation, speed-dependent yaw-rate limits, or explicit leader-follower reactions. The residual decoder and context encoders compensate for part of this mismatch, but aggressive maneuvers, sharp junction decisions, and dense interaction cases remain natural failure modes. This limitation is reflected in the scenario-level analysis in Section 6.2.

For uncertainty, an auxiliary head predicts per-timestep aleatoric scale parameters:

$$\sigma_m = \text{Softplus}(\text{MLP}_{\sigma}(\mathbf{h}_{\text{fused}})) \in \mathbb{R}^{T_f \times 2}. \quad (25)$$

Mode probabilities are obtained from a classification head:

$$\mathbf{p} = \text{Softmax}(\text{MLP}_{\text{cls}}(\mathbf{h}_{\text{fused}})) \in \mathbb{R}^M. \quad (26)$$

5.3. Training Procedure

5.3.1. Loss Function

Training uses a weighted sum of regression, classification, negative log-likelihood (NLL), ranking, and top-K terms:

$$\mathcal{L} = \mathcal{L}_{\text{reg}} + \lambda_{\text{cls}} \mathcal{L}_{\text{cls}} + \lambda_{\text{NLL}} \mathcal{L}_{\text{NLL}} + \lambda_{\text{rank}} \mathcal{L}_{\text{rank}} + \lambda_{\text{topk}} \mathcal{L}_{\text{topk}}. \quad (27)$$

The regression term follows a winner-takes-all formulation with SmoothL1:

$$\mathcal{L}_{\text{reg}} = \min_{m=1}^M \text{SmoothL1}(\tau_m, \tau_{\text{gt}}). \quad (28)$$

The classification target is defined by the mode that is closest to the ground truth trajectory:

$$m^* = \arg \min_m \|\tau_m - \tau_{\text{gt}}\|_2, \quad \mathcal{L}_{\text{cls}} = -\log p_{m^*}. \quad (29)$$

A lightweight probabilistic term is included via a diagonal Gaussian NLL over the horizon, using per-timestep scales:

$$\mathcal{L}_{\text{NLL}} = -\frac{1}{T_f} \sum_{t=1}^{T_f} \log \mathcal{N}(\mathbf{y}_t^{\text{gt}} | \boldsymbol{\mu}_{m^*,t}, \boldsymbol{\Sigma}_{m^*,t}). \quad (30)$$

Mode ranking is encouraged by weighting the final displacement error (FDE) with the predicted probabilities:

$$\mathcal{L}_{\text{rank}} = \sum_{m=1}^M p_m \cdot \text{FDE}(\tau_m, \tau_{\text{gt}}). \quad (31)$$

To promote diversity and multiple good hypotheses, a top-K regression term is applied to the best three modes (by ADE), with weights 1, 1/2, 1/3:

$$\mathcal{L}_{\text{topk}} = \frac{1}{3} \sum_{k=1}^3 \frac{1}{k} \text{SmoothL1}(\tau_{m_k}, \tau_{\text{gt}}). \quad (32)$$

Unless stated otherwise, $\lambda_{\text{cls}} = 1.0$, $\lambda_{\text{NLL}} = 0.2$, $\lambda_{\text{rank}} = 0.2$, and $\lambda_{\text{topk}} = 0.2$ are used.

The loss weights were selected through coarse validation runs on the 10k/2k development subset before full validation training. The objective was not to maximize a single oracle displacement metric only, but to avoid a configuration that improves minADE at the cost of mode ranking or unstable residual behavior. Table 3 therefore reports representative sensitivity diagnostics rather than a dense hyperparameter heatmap. The default setting kept $\lambda_{\text{cls}} = 1.0$, $\lambda_{\text{NLL}} = 0.2$, $\lambda_{\text{rank}} = 0.2$, and $\lambda_{\text{topk}} = 0.2$. The auxiliary waypoint/lane losses changed the supervision balance, while the soft-

assignment/calibration row changed the classification target and added a confidence loss. The gate-bias and no-gate rows are included because residual strength interacts with the same regression/classification objective.

The table shows why the final setting was retained despite some rows having slightly higher Top1Acc. The gate-bias 0.0 run improved Top1Acc but increased both displacement metrics, while the low-weight waypoint/lane auxiliary run improved Top1Acc but reduced Top1Conf and did not improve minADE/minFDE. Soft assignment with an explicit calibration loss reduced confidence sharply and degraded oracle displacement. In this compact regime, improving the classification/scoring objective alone was therefore not enough; it had to preserve the winner-takes-all trajectory supervision and the residual-gate behavior.

Table 3. Loss-weight, scoring, and residual-gate sensitivity on 10k/2k sanity runs. Rows are representative coarse validation runs, not a full grid search.

Variant	minADE ₆	minFDE ₆	Top1Acc	Top1Conf
Default weights, gate bias 1.0	1.861	4.306	0.328	0.324
Gate bias 0.5	1.930	4.452	0.209	0.206
Gate bias 0.0	1.905	4.387	0.366	0.329
No physics gate	2.121	4.946	0.270	0.267
Waypoint/lane aux. (0.10/0.05)	1.867	4.297	0.264	0.263
Waypoint/lane aux. (0.03/0.01)	1.866	4.304	0.347	0.299
Soft assignment + calibration (0.05)	2.139	5.745	0.239	0.174
TTC/risk neighbor weight 1.0	1.974	4.490	0.277	0.249

5.3.2. Optimization Configuration

Optimization is carried out with AdamW [41]. ReduceLROnPlateau is used as a scheduler, and gradient norms are clipped to improve stability. Table 4 lists the training hyperparameters.

Table 4. Training hyperparameters.

Parameter	Value	Parameter	Value
Optimizer	AdamW	Learning rate	1×10^{-4}
Weight decay	1×10^{-4}	Batch size	1024
Epochs	65	Gradient clipping	5.0
Scheduler	ReduceLROnPlateau	Patience (scheduler)	3 epochs
Learning-rate decay factor	0.5	Minimum learning rate	1×10^{-6}
Dropout rate	0.2	Hidden dimension	256
Neighbor hidden dim	128	Map hidden dim	128
Correction GRU hidden	128	Correction GRU layers	2
Gate bias init	1.0	Number of modes	6
Neighbor radius	30 m	Top- <i>k</i> neighbors	4

5.4. Evaluation Metrics

Evaluation follows the Argoverse 2 forecasting setting with $K = 6$ hypotheses. Displacement-based metrics are computed using the closest hypothesis to the ground truth.

The minADE₆ metric is defined as

$$\text{minADE}_6 = \min_{m=1}^M \frac{1}{T_f} \sum_{t=1}^{T_f} \|\tau_{m,t} - \tau_t^{\text{gt}}\|_2. \quad (33)$$

The minFDE₆ metric is

$$\text{minFDE}_6 = \min_{m=1}^M \|\tau_{m,T_f} - \tau_{T_f}^{\text{gt}}\|_2. \quad (34)$$

In addition, the average displacement error (ADE) of the most confident hypothesis is tracked via

$$\begin{aligned} \text{ADE}_1 &= \frac{1}{T_f} \sum_{t=1}^{T_f} \left\| \tau_{m_{\text{conf}},t} - \tau_t^{\text{gt}} \right\|_2, \\ m_{\text{conf}} &= \arg \max_m p_m. \end{aligned} \quad (35)$$

Miss rate (MR) is computed as the fraction of samples where all hypotheses miss the final position by more than a threshold:

$$\begin{aligned} \text{MR}_6 &= \frac{1}{N} \sum_{i=1}^N \mathbb{I} \left[\min_m \left\| \tau_{m,T_f}^{(i)} - \tau_{T_f}^{\text{gt},(i)} \right\|_2 > \tau_{\text{miss}} \right], \\ \tau_{\text{miss}} &= 2.0 \text{ m}. \end{aligned} \quad (36)$$

When mode probabilities are available, confidence behavior can be summarized with top-1 confidence (Top1Conf) and top-1 accuracy (Top1Acc), the fraction of cases where the top-1 mode is also the closest mode:

$$\begin{aligned} \text{Top1Conf} &= \frac{1}{N} \sum_{i=1}^N \max_m p_m^{(i)}, \\ \text{Top1Acc} &= \frac{1}{N} \sum_{i=1}^N \mathbb{I} \left[m_{\text{conf}}^{(i)} = m^{*,(i)} \right]. \end{aligned} \quad (37)$$

For calibration analysis, samples are binned by top-1 confidence and summarized with expected calibration error (ECE):

$$\text{ECE} = \sum_{b=1}^B \frac{|S_b|}{N} |\text{acc}(S_b) - \text{conf}(S_b)|, \quad (38)$$

where S_b is the set of samples whose top-1 confidence falls in bin b , $\text{acc}(S_b)$ is the fraction for which the top-1 mode is the closest mode, and $\text{conf}(S_b)$ is the mean top-1 confidence in the same bin.

5.5. Implementation Details

The model is implemented in PyTorch 2.11.0 and trained with Compute Unified Device Architecture (CUDA) acceleration on a single graphics processing unit (GPU). Data loading relies on the Argoverse 2 application programming interface (API) and a custom loader. Experiments reported here use 32-bit floating point (FP32) training with automatic mixed precision (AMP) disabled and default random seeding. Checkpoint selection follows the lowest validation minADE; the final configuration reported in Table 5 corresponds to the checkpoint selected under this criterion.

The full-validation results, qualitative diagnostics, reliability diagram, and latency measurements reported in Sections 6.1–6.6 were generated for this selected configuration. The 10k/2k rows are reported only as development diagnostics and are not intended as a separate benchmark release. Scenario tags were generated from the validation export using yaw-change, lateral-displacement, lane-flag, and neighbor-count heuristics; the dense-neighbor tag uses at least three valid neighboring tracks. The latency values were measured on the development laptop named in the runtime table and include the 60-step autoregressive decoder loop but exclude dataset loading.

Table 5. Quantitative results on Argoverse 2 ($K = 6$). Argoverse 2 baseline rows are reported from the Argoverse 2 dataset paper; QCNet/ProphNet are high-capacity published references. Top1Acc/Top1Conf are reported for the proposed model to summarize mode ranking quality.

Method	Reference Group	minADE ₆	minFDE ₆	Top1Acc	Top1Conf
Argoverse 2 nearest-neighbor baseline [4]	Dataset baseline	2.18	4.94	–	–
Argoverse 2 WIMP baseline [4]	Dataset baseline	1.47	2.90	–	–
Proposed model	Compact physics-guided	1.21	2.78	0.318	0.278
QCNet [42]	High-capacity reference	0.73	1.27	–	–
ProphNet [43]	High-capacity reference	0.68	1.33	–	–

6. Results and Discussion

Results are reported on the Argoverse 2 validation split. Table 5 includes two early Argoverse 2 dataset baselines, the proposed compact model, and two high-capacity references for scale. The table should not be read as a leaderboard claim. It is used to place the compact CTRA-residual decoder between simple dataset baselines and transformer/topology-heavy methods, while Top1Acc/Top1Conf record the mode-ranking behavior of the submitted model.

6.1. Quantitative Analysis

On the full validation export, the proposed model reaches $\text{minADE}_6 = 1.21$ m and $\text{minFDE}_6 = 2.78$ m under the $K = 6$ setting. This is 0.26 m lower in minADE and 0.12 m lower in minFDE than the WIMP row from the Argoverse 2 dataset paper, but it remains behind the high-capacity references. The gap to QCNet and ProphNet is 0.48–0.53 m in minADE and 1.45–1.51 m in minFDE. This gap is consistent with the architecture size: the proposed model uses local top- k neighbor attention and lane pooling, whereas these references use richer scene/topology modeling. The comparison should therefore be read as a compactness–accuracy trade-off, not as a claim of state-of-the-art performance.

The same checkpoint has average gate ≈ 0.31 and decay ≈ 0.95 in the residual branch. The later ablation and residual-influence analyses explain why these values matter: the residual term improves the CTRA anchor, but the learned decay damps the correction relative to a no-decay counterfactual over the 60-step horizon. This is a mechanism-level check on the decoder, separate from the headline validation numbers.

Table 6 records the mode-selection problem directly. $\text{ADE}_1/\text{FDE}_1$ are 3.12 m/8.17 m, much higher than oracle minADE/minFDE, even though the endpoint spread is 15.57 m and the normalized mode entropy is 0.903. The six trajectories are diverse enough to cover different futures in many cases, but the closest one is not reliably the top-ranked one. ECE is 0.044, so the immediate failure is not severe overconfidence; it is the ordering of modes.

Table 6. Additional validation diagnostics for mode selection, diversity, and calibration.

Metric	Value
$\text{ADE}_1 / \text{FDE}_1$	3.12 m / 8.17 m
MR_6 at 2 m	48.2%
ECE	0.044
Mean endpoint spread	15.57 m
Mean path spread	5.74 m
Normalized mode entropy	0.903

6.2. Qualitative and Scenario-Level Analysis

Figure 8 visualizes six validation exports selected to match the diagnostic tables: lane following, intersection turning, lane-change/merge geometry, dense-neighbor traffic, wrong top-1 ranking, and a high-error case. The wrong-top-1 example is especially important: one predicted mode remains near the ground truth, but its probability is lower than the selected mode. That visual case is the same failure measured by ADE_1/FDE_1 in Table 6. The high-error example is different because all six modes miss the future endpoint, so the problem is not only ranking but missing route intent.

To make the residual mechanism visible rather than only showing final trajectories, Figure 9 compares the full model with its CTRA anchor on selected validation samples. The CTRA anchor is obtained by setting the decoder residual correction scale to zero at inference time, so the dashed gray curve represents the physics rollout before residual injection. This diagnostic isolates where the gated residual shifts each mode relative to the CTRA prior: sustained correction helps in turning and merge cases, while decay limits long-horizon residual drift.

Table 7 reports the corresponding case-study metrics.

Table 7. Concrete case-study metrics from Figure 9. CTRA FDE is measured on the same mode that is oracle-best after residual correction.

Case	Neigh.	CTRA FDE	Full-Best FDE	Top-1 FDE	Gate/Decay
Intersection sustained correction	0	23.0	6.7	6.7	0.32 / 0.999
Lane-change/merge success	4	15.7	4.3	4.3	0.39 / 0.994
Decay-damped turn	4	2.6	1.9	1.9	0.27 / 0.000
Dense traffic ranking miss	3	21.7	0.7	11.2	0.40 / 1.000
High-error route miss	4	36.7	7.4	8.0	0.52 / 0.999

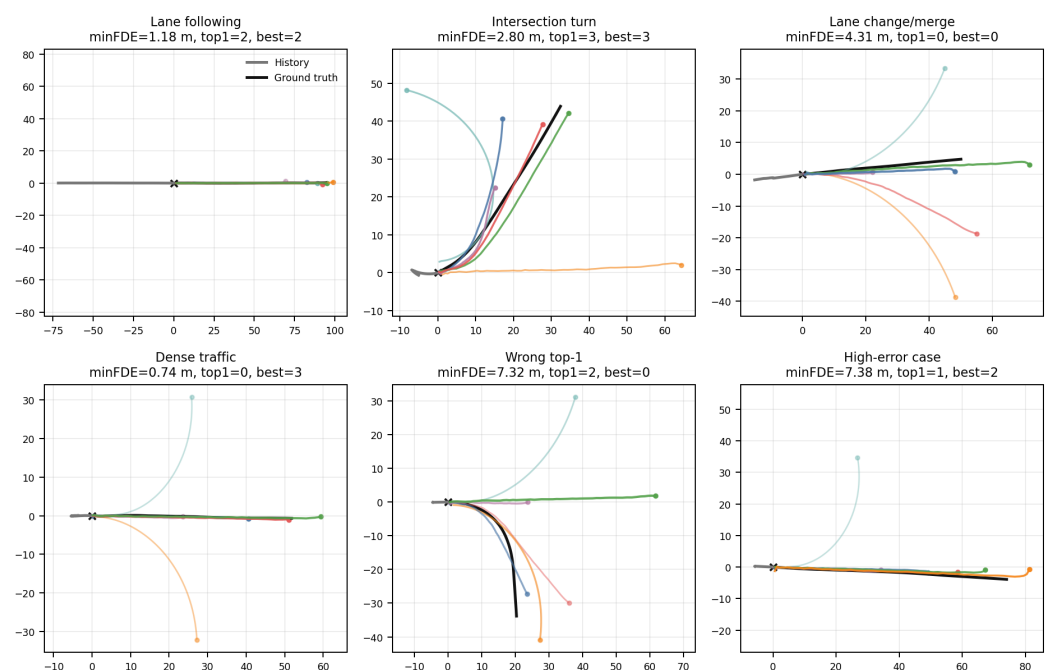


Figure 8. Validation cases selected from the diagnostic export. Black shows observed history, green shows the future ground truth, and colored curves show the six predicted modes, with opacity indicating mode probability. The cases are not random examples: they correspond to the failure groups discussed in Tables 6 and 8, including wrong top-1 ranking and cases where all six modes miss the future endpoint.

The case studies separate three failure modes that were hidden in the aggregate counts. In the first two rows, the residual branch makes sustained corrections to a CTRA anchor that bends too weakly or follows the wrong lateral trend. In the decay-damped turn, the anchor is already close and the learned decay suppresses long-horizon residual influence, so the full model only makes a small correction. In the dense-neighbor row, the model generates an accurate mode, but the scoring head ranks another mode first. In the high-error row, the residual substantially improves the CTRA anchor but the remaining endpoint error is still large, which is a route-hypothesis failure rather than only a residual-stability failure.

Table 8 was generated as a debugging table rather than as a mutually exclusive taxonomy. Tags can overlap because the same sample may be an intersection case, a turn, and a dense-neighbor case at the same time. The easiest group is lane following, with 0.87 m minADE₆ and 1.81 m minFDE₆. The two hardest groups are turning and lane change/merge, with miss rates of 86.4% and 86.1%. This pattern matches the qualitative plots: the local CTRA anchor handles smooth continuation, but pooled lane points are not enough to choose the correct route through junctions or merges.

Table 8. Scenario-level diagnostic breakdown on the validation split. Scenario tags are heuristic and can overlap.

Scenario	<i>n</i>	minADE ₆	minFDE ₆	MR ₆	Top1Acc
Lane following	1083	0.87	1.81	31.6%	0.299
Intersection context	22,396	1.24	2.85	49.0%	0.329
Turning	5831	2.18	5.63	86.4%	0.235
Lane change/merge	6284	2.13	5.43	86.1%	0.221
Dense neighbors (≥ 3)	13,040	1.17	2.67	45.7%	0.356

6.3. Calibration and Failure Analysis

Figure 10 gives a different failure signal from the displacement tables. ECE is 0.044, so the model is not simply assigning extreme confidence to wrong modes. The larger problem is ranking: Top1Acc is only about 0.32, and ADE₁/FDE₁ are much worse than oracle minADE/minFDE. In practical terms, the six-mode set is more useful than the single highest-probability mode.

As a post-hoc calibration check, scalar temperature scaling was fitted on a deterministic 20% calibration subset of the saved validation probabilities and evaluated on the remaining 80%. Since the saved export contains probabilities, the temperature transform was applied to log probabilities, which is equivalent to rescaling logits up to an additive constant. Table 9 shows that temperature scaling improves Top1Conf and ECE slightly, but it leaves Top1Acc unchanged because this monotonic transform does not change the order of modes. This confirms that the main problem is not only probability calibration; the ranking head often assigns the wrong mode the largest score. A future training run should therefore change the ranking supervision or add a separate mode-ordering head rather than applying calibration alone after inference.

Table 9. Post-hoc temperature scaling from saved validation probabilities. The split uses a deterministic 20% calibration subset and 80% evaluation subset.

Setting	Temperature	Top1Acc	Top1Conf	ECE
Uncalibrated	1.00	0.324	0.282	0.043
Temperature scaled	0.80	0.324	0.303	0.038

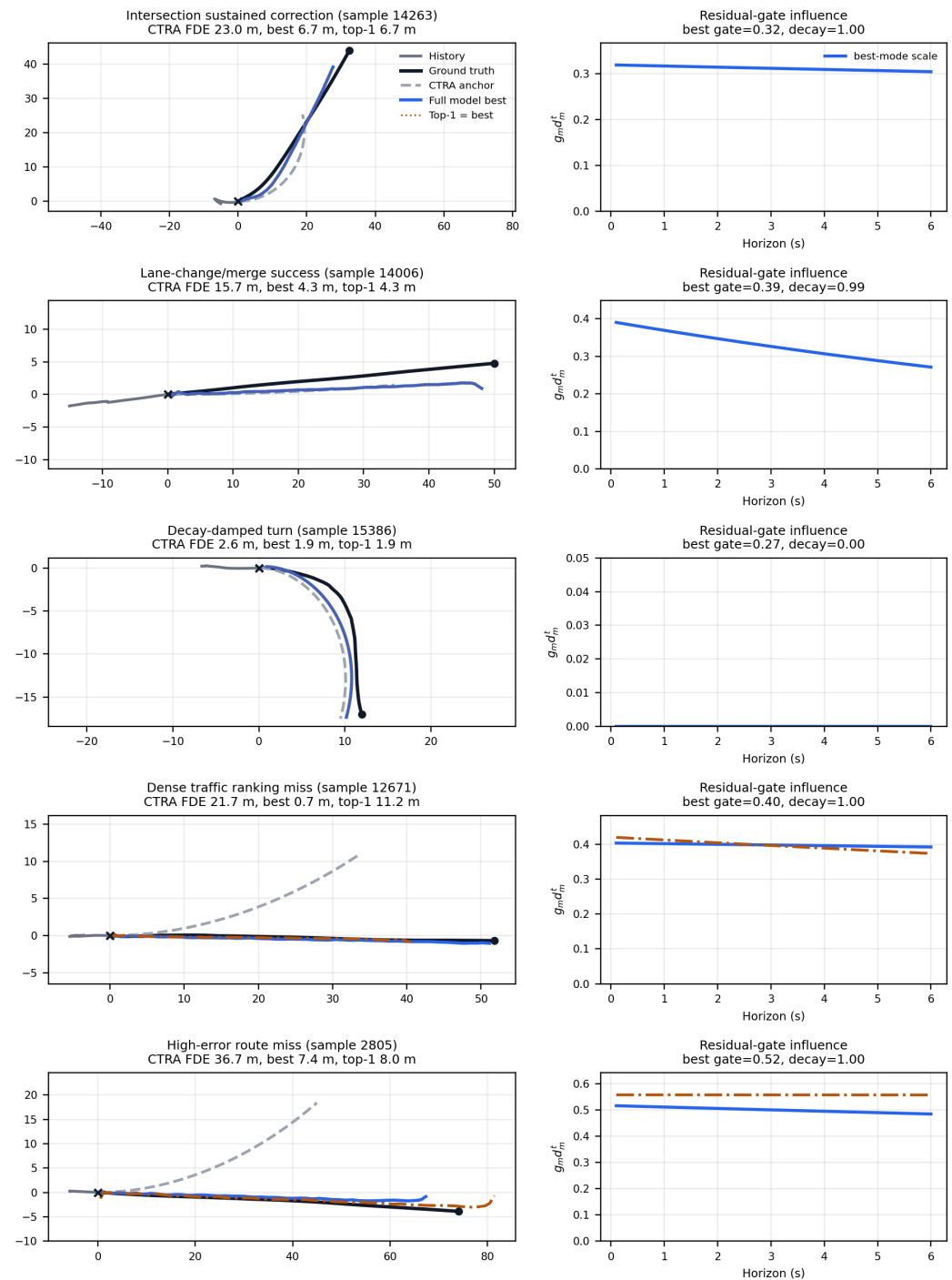


Figure 9. Mechanism-level comparison between the CTRA anchor and the full CTRA-residual decoder on representative validation cases. In the left column, gray shows observed history, black shows the future ground truth, dashed light gray shows the CTRA anchor obtained by disabling residual correction at inference time, solid blue shows the full-model best trajectory, and orange shows the top-ranked trajectory when it differs from the best trajectory. The right column shows the effective residual scale ($g_m d_m^t$) over the prediction horizon for the corresponding mode(s), illustrating sustained correction, decay-damped correction, ranking failure, and route-hypothesis failure.

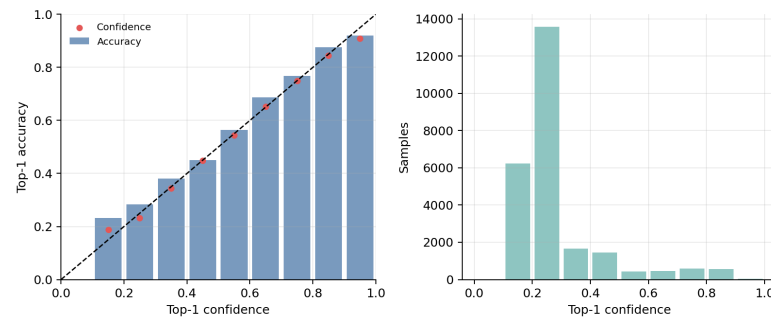


Figure 10. Calibration diagnostics for top-1 mode confidence on the validation export. **(Left):** reliability diagram, where the dashed line denotes perfect calibration. **(Right):** distribution of top-1 confidence values across validation samples.

Failure cases were also analyzed by selecting the worst 10% of validation samples according to minFDE. The threshold for this group is 5.94 m, corresponding to 2499 samples. Within these high-error samples, 2064 also have a wrong top-1 mode, 2320 occur in intersection-context scenes, 1961 are tagged as lane-change/merge or high-lateral-deviation cases, and 1209 include at least three nearby neighbors. These counts point to two separate fixes: better route hypotheses for the cases where all modes miss, and better scoring for the cases where the oracle mode exists but is ranked lower.

6.4. Ablations

Table 10 reports controlled sanity ablations (10k train / 2k val) under the same compact setting. The table is the main evidence for the decoder design. CTRA-only decoding gives $\text{minFDE}_6 = 7.189$ m. Adding non-autoregressive residuals reduces this to 4.532 m, and making the residual autoregressive with a larger correction GRU reduces it further to 4.157 m. Removing the gate increases the same metric to 4.946 m, while control-space residuals are worse at 6.159 m.

Table 10. Sanity ablations on a 10k/2k subset ($K = 6$). All runs use $H = 256$ and the legacy decoder.

Variant	minADE ₆	minFDE ₆	Top1Acc	Top1Conf
CTRA-only (no residuals)	2.939	7.189	—	—
Non-autoregressive residuals (baseline)	1.915	4.532	0.312	0.296
Autoregressive residuals	1.890	4.331	0.322	0.249
Autoregressive + bigger GRU	1.824	4.157	0.399	0.337
Autoregressive + bigger GRU + gate bias 1.0	1.861	4.306	0.328	0.324
Autoregressive + bigger GRU (no gate)	2.121	4.946	0.270	0.267
Control-space residuals	2.468	6.159	0.298	0.300

The selected decoder is not pure CTRA and not an unconstrained residual model. Table 10 shows that the CTRA anchor needs residual correction, and the residual correction needs gate/decay control.

6.5. Residual Influence over Time

Figure 11 plots the mean residual influence magnitude $\mathbb{E}[\|g_m d_m^t \mathbf{r}_{m,t}\|_2]$ across timesteps on a validation subset of 5120 instances, averaged over modes. The no-decay curve is computed from the same forward pass by setting $d_m = 1$, so the separation between the two curves isolates the learned decay factor rather than a different model. The residual

influence still grows with horizon because the future becomes less certain, but the learned decay keeps the injected correction below the no-decay counterfactual.

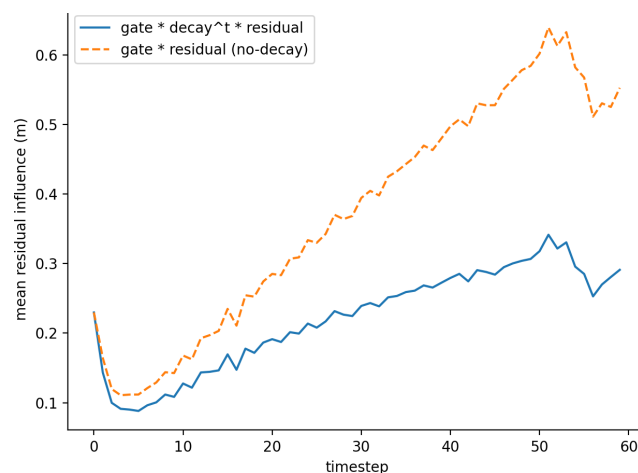


Figure 11. Mean residual influence magnitude vs. time, computed over 5120 validation instances and averaged over $M = 6$ modes. The plotted residual influence is the product of gate, decay, and residual magnitude. Although residual magnitude tends to grow with horizon, learned decay damps its influence relative to a no-decay counterfactual derived from the same forward pass.

6.6. Efficiency

The final model contains 1.78M parameters (FP32 size $\approx$ 6.8 megabytes (MB)). This number is reported to make the compactness claim concrete; it is not a substitute for a full embedded benchmark. Table 11 reports the model size. Direct wall-clock comparison with QCNet and ProphNet is implementation- and hardware-dependent, so Table 12 reports measured forward-pass latency for the proposed implementation and Table 13 separately lists published latency context from the two transformer-based references. The comparison is protocol-mismatched and should not be interpreted as a universal speed ranking.

Table 11. Model size (final configuration).

Metric	Value
Parameters	1.78 M
Model size (FP32)	6.8 MB

Table 12. Forward-pass latency of the proposed model on the development laptop. Central processing unit (CPU) results use an AMD Ryzen 7 7735HS; CUDA results use an NVIDIA GeForce RTX 4050 Laptop GPU.

Device	Batch Size	ms/Sample	Samples/s
CPU	1	82.7	12.1
CPU	32	3.5	286.6
CUDA	1	144.3	6.9
CUDA	32	4.8	206.4
CUDA	256	1.2	810.6

Table 12 should be interpreted by operating regime. For a single focal agent, the CPU forward pass is 82.7 ms, corresponding to 12.1 forecasts/s and remaining within the 10 Hz forecasting interval in this measured setup. The single-sample CUDA row is not the favorable deployment point because the autoregressive decoder contains many small sequential operations. CUDA becomes useful when independent focal agents or candidate scenes are batched: at batch size 256, the implementation reaches 1.2 ms/sample, or

810.6 focal-agent forecasts/s. The architecture is therefore better characterized by small memory footprint and batched focal-agent throughput than by single-sample GPU latency.

Table 13. Published latency context for transformer-based references. These rows use different hardware and prediction protocols, so they are provided as context rather than a direct benchmark against Table 12.

Method	Hardware	Published Latency Protocol
Proposed model	Ryzen 7 7735HS / RTX 4050 laptop	This work: focal-agent forward pass; 82.7 ms at CPU batch 1, 3.5 ms/sample at CPU batch 32, and 1.2 ms/sample at CUDA batch 256.
QCNet [42]	NVIDIA A40	The QCNet supplementary material reports 46 ms for predicting all agents in an average Argoverse 2 scene and 72 ms for a dense scene with 190 agents.
ProphNet [43]	NVIDIA V100	Published latency analysis reports 28.0 ms and 0.40 GFLOPs with 64 agents.

This comparison defines the deployment claim more precisely. QCNet and ProphNet report scene-level or multi-agent inference on different accelerator hardware, whereas Table 12 measures a focal-agent forward pass on a laptop-class system. The proposed model should therefore not be read as universally faster than those transformer-based references. Its supported efficiency claim is a different operating point: a 1.78 M-parameter, 6.8 MB model with measured focal-agent CPU feasibility and batched focal-agent GPU throughput. For all-agent scene-level inference with richer topology reasoning, the larger transformer-based systems remain stronger references.

6.7. Negative Results

Table 14 lists variants that were tried and discarded on the 10k/2k sanity split. Goal conditioning, map cross-attention, soft assignment, multi-head decoding, TTC/risk weighting, and control-space residuals all increased error relative to the selected autoregressive-residual configuration. These rows are included because they explain why the final model stayed simple instead of accumulating extra modules.

The failures match the later scenario analysis. Map cross-attention added another attention operation over local lane embeddings, but the input still had no lane successor/predecessor graph, traffic signal state, or route hypothesis. It therefore increased fusion capacity without adding the missing topology variable that turning and merge cases need. Goal conditioning made the decoder sensitive to wrong intermediate lane goals; when a selected goal was wrong, the autoregressive residual inherited that error over the full horizon. Soft assignment weakened the winner-takes-all signal that separates the six modes, which reduced both displacement accuracy and confidence sharpness. Multi-head decoding increased specialization pressure per head, but each head then received fewer winner-mode updates in the 10k/2k setting. Control-space residuals were less stable than position residuals because small acceleration or yaw-rate errors integrate over time and can quickly dominate the CTRA anchor. The TTC/risk row improved neither route selection nor scoring because the top- k neighbor features still lacked explicit interaction outcomes such as yielding or priority.

Table 14. Negative results on sanity runs (10k/2k, $K = 6$). The effect column summarizes the observed failure mode.

Variant	minADE ₆	minFDE ₆	Observed Effect
Goal conditioning	1.953	4.623	Wrong lane goals propagate through residual rollout.
Map cross-attention	2.100	4.791	More local geometry, but no lane connectivity or route state.
Soft assignment + calibration	2.139	5.745	Weaker winner signal; confidence collapses to 0.174 Top1Conf.
TTC/risk neighbor weight	1.974	4.490	Local risk cue does not resolve route intent.
Control-space residuals	2.468	6.159	Small control errors integrate into large endpoint drift.
Multi-head decoder	2.220	5.161	Per-head supervision is fragmented in the compact subset.

6.8. Discussion

Table 10 shows why the decoder was not left as either pure CTRA or pure residual correction. In the 10k/2k ablations, CTRA-only decoding reaches $\text{minADE}_6 = 2.939$ m and $\text{minFDE}_6 = 7.189$ m. Adding autoregressive residuals and a larger correction GRU reduces these values to 1.824 m and 4.157 m. The no-gate row is the clearest evidence for the gate: with the same larger correction GRU, removing it increases minADE from 1.824 m to 2.121 m and minFDE from 4.157 m to 4.946 m.

The full validation run shows the same trade-off at benchmark scale. The model improves over early Argoverse 2 baselines in oracle displacement metrics, reaching $\text{minADE}_6 = 1.21$ m and $\text{minFDE}_6 = 2.78$ m, but it remains behind transformer/topology-heavy references such as QCNet and ProphNet. The gap is not surprising from the model definition: this implementation uses a small context encoder and a physics-guided decoder rather than full scene-level topology reasoning.

The main remaining weakness is mode ranking. Table 6 shows a large gap between oracle metrics and top-1 metrics: $\text{ADE}_1/\text{FDE}_1$ are 3.12 m/8.17 m even though the best-of-six trajectory is often much closer. The reliability analysis gives $\text{ECE} = 0.044$, so the issue is not simply overconfidence; the scoring head often fails to rank the closest mode first. A planner that consumes all six modes can still benefit from the oracle set, whereas a planner that uses only the highest-probability mode would see substantially higher risk.

Table 8 points to the next modeling target: route intent, not local neighbor encoding alone. Lane-following cases are easiest ($\text{minADE}_6 = 0.87$ m, $\text{minFDE}_6 = 1.81$ m), which matches the expectation that a CTRA anchor is already strong for smooth local motion. Turning and lane-change/merge cases have high miss rates (86.4% and 86.1%), indicating that the current lane-pooling module does not resolve route-level intent. Dense-neighbor cases degrade less severely than turning cases, suggesting that local neighbor attention captures some interaction pressure but does not replace explicit lane-topology reasoning.

Runtime measurements also narrow the deployment claim. The model is compact in memory (1.78M parameters, approximately 6.8 MB in FP32), and the measured CPU latency is suitable for single-focal-agent prediction at a 10 Hz cycle. GPU use is more appropriate for batched focal-agent inference, where the sequential residual decoder cost can be amortized across many independent samples.

Overall, the results position the method as a compact physics-guided predictor with visible failure modes rather than as a general replacement for larger topology-heavy models. The next architectural changes should target route hypotheses, lane connectivity, and probability ranking while preserving the gate/decay behavior that stabilizes the residual rollout.

6.9. Limitations

The proposed model is not state-of-the-art on Argoverse 2 compared with transformer- and topology-heavy methods, and the compact context encoders limit intent disambiguation at complex junctions. The CTRA prior is also a simplified kinematic model: it does not represent steering dynamics, tire saturation, speed-dependent yaw limits, or explicit car-following behavior. In addition, probability calibration and ranking remain imperfect, as indicated by moderate Top1Acc/Top1Conf despite low minADE. Future work should incorporate richer map-topology reasoning, better probabilistic scoring, and stronger vehicle-dynamics alternatives such as bicycle-model or IDM-inspired priors, but these changes should be tested against the no-gate and control-space residual failures already reported here. For embedded or V2X-oriented deployments, upstream data reduction is also relevant: machine-centric FDCT vision processing has been proposed as a way to reduce robotic perception data volume before downstream modules consume it [44]. That direction is complementary to this paper because the present experiments start from cleaned track and lane tensors rather than raw images.

7. Conclusions

This paper reports a compact Argoverse 2 forecaster whose decoder is CTRA plus gated autoregressive position residuals. The result is not a state-of-the-art Argoverse 2 score. Its value is narrower: the ablations show which part of a compact physics-guided decoder matters and where the compact version fails.

On the Argoverse 2 validation split, the final model reaches $\text{minADE}_6 = 1.21$ m and $\text{minFDE}_6 = 2.78$ m. The ablation study explains where this gain comes from: CTRA-only is too rigid, autoregressive residuals reduce most of the displacement error, and removing gate/decay control worsens the same larger-residual setting. The residual is therefore useful because it is conditioned on the evolving predicted state and constrained by a mode-specific gate and decay.

The next version should first address the errors already visible in Tables 6–8: top-1 mode ranking, route hypotheses at turns and merges, and lane-connectivity encoding. These changes should be tested without removing the residual gate/decay mechanism, because the no-gate ablation is worse. Deployment-oriented extensions should also measure the full perception-to-prediction chain, including possible machine-centric compression of upstream visual data [44], because the latency of map and neighbor context formation is outside the present forward-pass benchmark.

Author Contributions: Conceptualization, U.Ö. and I.S.; methodology, U.Ö. and I.S.; software, U.Ö.; validation, U.Ö. and L.R.; validation of revised materials, A.B. and N.N.; formal analysis, U.Ö.; investigation, U.Ö.; resources, L.R., A.B. and N.N.; data curation, U.Ö.; writing—original draft preparation, U.Ö.; writing—review and editing, U.Ö., I.S., L.R., A.B. and N.N.; visualization, U.Ö.; supervision, I.S. and L.R.; project administration, I.S., L.R., A.B. and N.N. All authors have read and agreed to the published version of the manuscript.

Funding: This study was carried out with the financial support of the Committee of Science of the Ministry of Science and Higher Education of the Republic of Kazakhstan under Contract 388/PTF24-26 dated 1 October 2024 under the scientific project IRN BR24993232, “Development of innovative technologies for conducting digital forensic investigations using intelligent software-hardware complexes”.

Data Availability Statement: The data analyzed in this study are publicly available from the Argoverse 2 Motion Forecasting dataset at <https://www.argoverse.org/av2.html>, accessed on 24 June 2026. Additional analysis outputs supporting the reported results are available from the corresponding author upon reasonable request.

Acknowledgments: The authors acknowledge the International Fund for covering the article processing charge.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Teng, S.; Hu, X.; Deng, P.; Li, B.; Li, Y.; Ai, Y.; Yang, D.; Li, L.; Xuanyuan, Z.; Zhu, F.; et al. Motion Planning for Autonomous Driving: The State of the Art and Future Perspectives. *IEEE Trans. Intell. Veh.* **2023**, *8*, 3692–3711. [\[CrossRef\]](#)
2. Barrios, F.A.; Biswas, A.; Emadi, A. Deep Learning-Based Motion Prediction Leveraging Autonomous Driving Datasets: State-of-the-Art. *IEEE Access* **2024**, *12*, 93318–93332. [\[CrossRef\]](#)
3. Galvão, L.G.; Huda, M.N. Pedestrian and vehicle behaviour prediction in autonomous vehicle system—A review. *Expert Syst. Appl.* **2024**, *238*, 121983. [\[CrossRef\]](#)
4. Wilson, B.; Qi, W.; Agarwal, T.; Lambert, J.; Singh, J.; Khandelwal, S.; Pan, B.; Kumar, R.; Hartnett, A.; Pontes, J.K.; et al. Argoverse 2: Next Generation Datasets for Self-Driving Perception and Forecasting. In Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS), New Orleans, LA, USA, 10–16 December 2023.
5. Huang, Y.; Du, J.; Yang, Z.; Zhou, Z.; Zhang, L.; Chen, H. A Survey on Trajectory-Prediction Methods for Autonomous Driving. *IEEE Trans. Intell. Veh.* **2022**, *7*, 652–674. [\[CrossRef\]](#)
6. Mozaffari, S.; Al-Jarrah, O.Y.; Dianati, M.; Jennings, P.; Mouzakitis, A. Deep Learning-Based Vehicle Behavior Prediction for Autonomous Driving Applications: A Review. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 33–47. [\[CrossRef\]](#)
7. Adnan Yusuf, S.; Khan, A.; Souissi, R. Vehicle-to-everything (V2X) in the autonomous vehicles domain—A technical review of communication, sensor, and AI technologies for road user safety. *Transp. Res. Interdiscip. Perspect.* **2024**, *23*, 100980. [\[CrossRef\]](#)
8. Liu, J.; Luo, Y.; Zhong, Z.; Li, K.; Huang, H.; Xiong, H. A Probabilistic Architecture of Long-Term Vehicle Trajectory Prediction for Autonomous Driving. *Engineering* **2022**, *19*, 228–239. [\[CrossRef\]](#)
9. Lefkopoulos, V.; Menner, M.; Domahidi, A.; Zeilinger, M.N. Interaction-Aware Motion Prediction for Autonomous Driving: A Multiple Model Kalman Filtering Scheme. *IEEE Robot. Autom. Lett.* **2021**, *6*, 80–87. [\[CrossRef\]](#)
10. Lin, L.; Li, W.; Bi, H.; Qin, L. Vehicle Trajectory Prediction Using LSTMs With Spatial–Temporal Attention Mechanisms. *IEEE Intell. Transp. Syst. Mag.* **2022**, *14*, 197–208. [\[CrossRef\]](#)
11. Chen, X.; Zhang, H.; Zhao, F.; Hu, Y.; Tan, C.; Yang, J. Intention-Aware Vehicle Trajectory Prediction Based on Spatial-Temporal Dynamic Attention Network for Internet of Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 19471–19483. [\[CrossRef\]](#)
12. Yang, J.; Chen, Y.; Du, S.; Chen, B.; Principe, J.C. IA-LSTM: Interaction-Aware LSTM for Pedestrian Trajectory Prediction. *IEEE Trans. Cybern.* **2024**, *54*, 3904–3917. [\[CrossRef\]](#) [\[PubMed\]](#)
13. Yin, X.; Wen, J.; Lei, T.; Xiao, G.; Zhan, Q. IMA-LSTM: An Interaction-Based Model Combining Multihead Attention with LSTM for Trajectory Prediction in Multivehicle Interaction Scenario. *Int. J. Intell. Syst.* **2024**, *2024*, 3058863. [\[CrossRef\]](#)
14. Mo, X.; Huang, Z.; Xing, Y.; Lv, C. Multi-Agent Trajectory Prediction With Heterogeneous Edge-Enhanced Graph Attention Network. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 9554–9567. [\[CrossRef\]](#)
15. Sheng, Z.; Xu, Y.; Xue, S.; Li, D. Graph-Based Spatial-Temporal Convolutional Network for Vehicle Trajectory Prediction in Autonomous Driving. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 17654–17665. [\[CrossRef\]](#)
16. Zhang, X.; Angeloudis, P.; Demiris, Y. Dual-Branch Spatio-Temporal Graph Neural Networks for Trajectory Prediction. *Pattern Recognit.* **2023**, *139*, 109466. [\[CrossRef\]](#)
17. Zhang, L.; Ma, J. A spatiotemporal graph wavelet neural network for traffic flow prediction. *J. Inf. Intell.* **2025**, *3*, 173–188. [\[CrossRef\]](#)
18. Zhang, K.; Feng, X.; Wu, L.; He, Z. Trajectory Prediction for Autonomous Driving Using Spatial-Temporal Graph Attention Transformer. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 22343–22353. [\[CrossRef\]](#)
19. Jia, X.; Wu, P.; Chen, L.; Liu, Y.; Li, H.; Yan, J. HDGT: Heterogeneous Driving Graph Transformer for Multi-Agent Trajectory Prediction via Scene Encoding. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 13860–13875. [\[CrossRef\]](#) [\[PubMed\]](#)
20. Wang, B.; Liu, Y.; Wang, R.; Sun, Q. Trajectory Prediction Model of Electric Vehicle Autonomous Driving Based on Hybrid Attention Transformer Network. *IET Intell. Transp. Syst.* **2025**, *19*, e70022. [\[CrossRef\]](#)
21. Shi, S.; Jiang, L.; Dai, D.; Schiele, B. Motion Transformer with Global Intention Localization and Local Movement Refinement. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 6531–6543. [\[CrossRef\]](#)
22. Li, A.; Xu, Z.; Pan, Y.; Zhang, J.; Chen, N.; Yu, H.; Chen, Y.; Li, Y. A Convolutional Transformer Network for Vehicle Trajectory Prediction in Urban Traffic Scenarios. *J. Transp. Eng. Part A Syst.* **2026**, *152*, 04025132. [\[CrossRef\]](#)
23. Ge, M.; Ohtani, K.; Ding, M.; Niu, Y.; Zhang, Y.; Takeda, K. Multimodal Trajectory Prediction for Diverse Vehicle Types in Autonomous Driving with Heterogeneous Data and Physical Constraints. *Sensors* **2024**, *24*, 7323. [\[CrossRef\]](#) [\[PubMed\]](#)
24. Wang, J.; Pant, Y.V.; Jiang, Z. Learning-Based Modeling of Human–Autonomous Vehicle Interactions in Platooning. *Transp. Res. Part C Emerg. Technol.* **2024**, *162*, 104600. [\[CrossRef\]](#)

25. Sheng, Z.; Huang, Z.; Chen, S. Kinematics-aware multigraph attention network with residual learning for heterogeneous trajectory prediction. *J. Intell. Connect. Veh.* **2024**, *7*, 138–150. [\[CrossRef\]](#)
26. Sun, S.; Wang, C.; Xiao, B.; Liu, X.; Shi, C.; Sun, R.; Han, R. Heterogeneous Multi-Agent Risk-Aware Graph Encoder with Continuous Parameterized Decoder for Autonomous Driving Trajectory Prediction. *Electronics* **2025**, *14*, 105. [\[CrossRef\]](#)
27. Tang, L.; Yan, F.; Zou, B.; Li, W.; Lv, C.; Wang, K. Trajectory prediction for autonomous driving based on multiscale spatial-temporal graph. *IET Intell. Transp. Syst.* **2023**, *17*, 386–399. [\[CrossRef\]](#)
28. Vishnu, C.; Abhinav, V.; Roy, D.; Mohan, C.K.; Babu, C.S. Improving Multi-Agent Trajectory Prediction Using Traffic States on Interactive Driving Scenarios. *IEEE Robot. Autom. Lett.* **2023**, *8*, 2708–2715. [\[CrossRef\]](#)
29. Wu, K.; Zhou, Y.; Shi, H.; Li, X.; Ran, B. Graph-Based Interaction-Aware Multimodal 2D Vehicle Trajectory Prediction Using Diffusion Graph Convolutional Networks. *IEEE Trans. Intell. Veh.* **2024**, *9*, 3630–3643. [\[CrossRef\]](#)
30. Na, Y.; Lee, M.; Kang, J.; Sunwoo, M.; Jo, K. Collision Probability Field Based Interaction-Aware Longitudinal Motion Prediction. *IEEE Trans. Intell. Transp. Syst.* **2024**, *25*, 12095–12107. [\[CrossRef\]](#)
31. Chai, J.; Liu, J.; Huang, J.; Huang, C. GACNet: Interactive Prediction of Surrounding Vehicles Behavior Under High Collision Risk. *Adv. Intell. Syst.* **2025**, *7*, 2401040. [\[CrossRef\]](#)
32. Chen, H.; Li, N.; Shan, H.; Liu, E.; Xiang, Z. Multi-Task Trajectory Prediction Using a Vehicle-Lane Disentangled Conditional Variational Autoencoder. *Sensors* **2025**, *25*, 4505. [\[CrossRef\]](#) [\[PubMed\]](#)
33. Dai, Y.; Zhang, Y.; Zhou, X.; Wang, Q.; Song, X.; Wang, S. MTGNet: Multi-Agent End-to-End Motion Trajectory Prediction with Multimodal Panoramic Dynamic Graph. *Appl. Sci.* **2025**, *15*, 5244. [\[CrossRef\]](#)
34. Liu, Z.; Lei, J.; Yi, P.; Hong, Y. An interaction-fair semi-decentralized trajectory planner for connected and autonomous vehicles. *Auton. Intell. Syst.* **2025**, *5*, 1. [\[CrossRef\]](#)
35. Ivanovic, B.; Leung, K.; Schmerling, E.; Pavone, M. Multimodal Deep Generative Models for Trajectory Prediction: A Conditional Variational Autoencoder Approach. *IEEE Robot. Autom. Lett.* **2021**, *6*, 295–302. [\[CrossRef\]](#)
36. Bertugli, A.; Calderara, S.; Coscia, P.; Ballan, L.; Cucchiara, R. AC-VRNN: Attentive Conditional-VRNN for multi-future trajectory prediction. *Comput. Vis. Image Underst.* **2021**, *210*, 103245. [\[CrossRef\]](#)
37. Yao, Y.; Atkins, E.; Johnson-Roberson, M.; Vasudevan, R.; Du, X. BiTraP: Bi-Directional Pedestrian Trajectory Prediction With Multi-Modal Goal Estimation. *IEEE Robot. Autom. Lett.* **2021**, *6*, 1463–1470. [\[CrossRef\]](#)
38. Medina-Lee, J.F.; Trentin, V.; Hortelano, J.L.; Artuñedo, A.; Godoy, J.; Villagra, J. $IA(MP)^2$: Framework for Online Motion Planning Using Interaction-Aware Motion Predictions in Complex Driving Situations. *IEEE Trans. Intell. Veh.* **2024**, *9*, 357–371. [\[CrossRef\]](#)
39. Wen, Z.; Zhang, Y.; Chen, X.; Wang, J.; Li, Y.H.; Huang, Y.K. TOFG: Temporal Occupancy Flow Graph for Prediction and Planning in Autonomous Driving. *IEEE Trans. Intell. Veh.* **2024**, *9*, 2850–2863. [\[CrossRef\]](#)
40. Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv* **2014**, arXiv:1406.1078. [\[CrossRef\]](#)
41. Loshchilov, I.; Hutter, F. Decoupled weight decay regularization. In Proceedings of the International Conference on Learning Representations (ICLR), New Orleans, LA, USA, 6–9 May 2019.
42. Zhou, Z.; Wang, J.; Li, Y.H.; Huang, Y.K. Query-Centric Trajectory Prediction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 17–24 June 2023.
43. Wang, X.; Su, T.; Da, F.; Yang, X. ProphNet: Efficient Agent-Centric Motion Forecasting with Anchor-Informed Proposals. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 17–24 June 2023.
44. Wiseman, Y. Achieving Robotic Data Efficiency Through Machine-Centric FDCT Vision Processing. *Sensors* **2026**, *26*, 518. [\[CrossRef\]](#) [\[PubMed\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.