NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

DATA SCIENCE & INFORMATION TECHNOLOGIES

MSc THESIS

A Spatial Attention Mechanism for Motion Prediction in Autonomous Driving

Nektarios I. Christou

Supervisor:
Associate Professor Aggelos Pikrakis, University of Piraeus

ATHENS

MARCH 2026

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΕΠΙΣΤΗΜΗ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΕΣ ΠΛΗΡΟΦΟΡΙΑΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μηχανισμός Χωρικής Προσοχής για την Πρόβλεψη Κίνησης στην Αυτόνομη Οδήγηση

Νεκτάριος Ι. Χρήστου

Επιβλέπων:
Αναπληρωτής Καθηγητής Άγγελος Πικράκης, Πανεπιστήμιο Πειραιώς

ΑΘΗΝΑ

ΜΑΡΤΙΟΣ 2026

MSc THESIS

A Spatial Attention Mechanism for Motion Prediction in Autonomous Driving

Nektarios I. Christou
S.N.: 7115152200002

SUPERVISOR:

Associate Professor Aggelos Pikrakis, University of Piraeus

EXAMINATION COMMITTEE:

Aggelos Pikrakis, Associate Professor, University of Piraeus
Konstantinos Koutroubas, Dr., National Observatory of Athens
Dimitrios Gounopoulos, Professor, National and Kapodistrian University of Athens

Examination Date: 02 March, 2026

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μηχανισμός Χωρικής Προσοχής για την Πρόβλεψη Κίνησης στην Αυτόνομη Οδήγηση

Νεκτάριος Ι. Χρήστου
Α.Μ.: 7115152200002

ΕΠΙΒΛΕΠΩΝ:

Αναπληρωτής Καθηγητής Άγγελος Πικράκης, Πανεπιστήμιο Πειραιώς

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ:

Άγγελος Πικράκης, Αναπληρωτής Καθηγητής, Πανεπιστήμιο Πειραιώς
Κωνσταντίνος Κουτρούμπας, Διευθυντής Ερευνών, Εθνικό Αστεροσκοπείο Αθηνών
Δημήτριος Γουνόπουλος, Καθηγητής, Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

Ημερομηνία Εξέτασης: 02 Μαρτίου 2026

ABSTRACT

Accurate and efficient motion forecasting is essential for safe autonomous driving, where an ego vehicle must anticipate the future trajectories of surrounding agents in complex urban environments. Recent transformer-based architectures such as HPTR provide a strong and lightweight backbone for trajectory prediction by jointly encoding map and agent information. In this thesis we build directly on HPTR and study how to augment its attention mechanism with an explicit spatial density prior. Our main contribution is a Spatial Density Module (SDM) that models the interaction space with a Gaussian Mixture Model and uses its normalized density as a soft, spatially aware filter on attention maps. This gives a Spatial Density Transformer (SDT) variant that keeps HPTR's encoder-decoder design, number of parameters, and training setup, and changes only the way attention is distributed over possible interaction regions.

We conduct experiments on the Waymo Open Motion Dataset and focus the analysis on the behavior of the Spatial Density Module. In an ablation study over the number of mixture components and SDM hyperparameters, and using simple diagnostic metrics for the mixtures (component utilization, spatial variance, and entropy), we observe that the learned density tends to place mass in regions where there are agents and map elements instead of empty areas. This suggests that the SDM does learn a meaningful notion of interaction density. In terms of standard forecasting metrics on WOMD, SDT reaches reasonable performance, but we do not carry out a full head to head comparison with HPTR under identical conditions, so we cannot draw firm conclusions about relative gains or losses. The main contribution of this work is therefore the design and analysis of the density-aware attention mechanism and its diagnostics, rather than a clear improvement in benchmark scores. The experiments indicate that adding a learned, density based attention filter on top of an existing lightweight transformer is a viable direction that could be explored further in future work, including variants where the mixtures are constrained to behave like k nearest neighbor neighborhoods as in HPTR to better understand how useful density-aware attention is in practice.

SUBJECT AREA: computer vision, autonomous driving, trajectory prediction, motion prediction, deep learning

KEYWORDS: trajectory prediction, autonomous vehicles, spatial density attention, spatial density module, motion prediction

ΠΕΡΙΛΗΨΗ

Η ακριβής και αποδοτική πρόβλεψη κίνησης αποτελεί βασικό στοιχείο για την ασφαλή λειτουργία αυτόνομων οχημάτων, τα οποία πρέπει να προβλέπουν τις μελλοντικές τροχιές των γύρω χρηστών του δρόμου σε σύνθετα αστικά περιβάλλοντα. Πρόσφατες αρχιτεκτονικές μετασχηματιστών, όπως το HPTR, προσφέρουν ένα ισχυρό και αποδοτικό αλγόριθμο για πρόβλεψη τροχιών, συνδυάζοντας πληροφορία από τον περιβάλλοντα χώρο και τους έτερους χρήστες (πράκτορες). Στην παρούσα διπλωματική βασιζόμαστε στο HPTR και μελετάμε πώς μπορεί να ενισχυθεί ο μηχανισμός προσοχής του μέσω ενός Νευρωνικού Δικτύου που προβλέπει ένα Μοντέλο Μίγματος Γκαουσιανών. Κύρια συνεισφορά μας αποτελεί ο Μηχανισμός Χωρικής Πυκνότητας (Spatial Density Module, SDM), ο οποίος μοντελοποιεί τον χώρο αλληλεπιδράσεων με ένα Μοντέλο Μίγματος Γκαουσιανών και χρησιμοποιεί την κανονικοποιημένη πυκνότητά του ως φίλτρο πάνω στους χάρτες προσοχής. Η προκύπτουσα παραλλαγή, Spatial Density Transformer (SDT), διατηρεί την αρχιτεκτονική κωδικοποιητή-αποκωδικοποιητή, τις παραμέτρους και τη διαδικασία εκπαίδευσης του HPTR και αλλάζει μόνο τον τρόπο με τον οποίο κατανέμεται η προσοχή στις πιθανές περιοχές αλληλεπίδρασης.

Πραγματοποιούμε πειράματα στο σύνολο δεδομένων Waymo Open Motion και εστιάζουμε στην ανάλυση της συμπεριφοράς του Μηχανισμού Χωρικής Πυκνότητας. Μελετώντας τον αριθμό των συνιστωσών του μίγματος και τις υπερπαραμέτρους του SDM με τη χρήση απλών διαγνωστικών μετρικών για τα μίγματα (component utilization, χωρική διασπορά και εντροπία), παρατηρούμε ότι η εκτιμώμενη πυκνότητα τείνει να συγκεντρώνεται σε περιοχές όπου υπάρχουν άλλα οχήματα αντί για κενές περιοχές. Αυτό υποδηλώνει ότι το SDM κωδικοποιεί με ουσιαστικό τρόπο την πυκνότητα των αλληλεπιδράσεων. Ως προς τις μετρικές πρόβλεψης του WOMD, το SDT παρουσιάζει ικανοποιητική απόδοση, αλλά δεν πραγματοποιούμε πλήρη σύγκριση με το HPTR υπό ακριβώς ίδιες συνθήκες εκπαίδευσης, επομένως δεν μπορούν να εξαχθούν ασφαλή συμπεράσματα για τυχόν βελτίωση της ευστοχίας του. Η κύρια συνεισφορά της εργασίας είναι η σχεδίαση και η ανάλυση του πυκνωτικά καθοδηγούμενου μηχανισμού προσοχής και των διαγνωστικών του, και όχι η επίτευξη βελτίωσης σε επίπεδο benchmark. Τα πειραματικά αποτελέσματα δείχνουν ότι η προσθήκη ενός εκμαθημένου, πυκνωτικού φίλτρου προσοχής σε συνδυασμό με μια αποδοτική αρχιτεκτονική μετασχηματιστή αποτελεί μια ενδιαφέρουσα κατεύθυνση που μπορεί να μελετηθεί περαιτέρω σε μελλοντική έρευνα, η οποία θα περιλαμβάνει παραλλαγές όπου τα μίγματα περιορίζονται ώστε να συμπεριφέρονται σαν περιοχές k nearest neighbor όπως στο HPTR ώστε να κατανοηθεί καλύτερα πόσο χρήσιμη είναι στην πράξη η προσοχή βασισμένη σε χωρική πυκνότητα.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: υπολογιστική όραση, αυτόνομη οδήγηση, πρόβλεψη τροχιάς, πρόβλεψη κίνησης, βαθιά μάθηση

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: πρόβλεψη τροχιάς, αυτόνομα οχήματα, μηχανισμός χωρικής προσοχής, μηχανισμός χωρικής πυκνότητας, πρόβλεψη κίνησης

To myself, for the persistence through challenges and the resolve to carry on.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my supervisor, Associate Professor Aggelos Pikrakis, for his continuous guidance and encouragement throughout this work. I am also thankful to a colleague who provided valuable help during this journey, and to my family and friends for their constant support and understanding.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

PREFACE

This thesis was carried out as part of the MSc in Data Science & Information Technologies at the National and Kapodistrian University of Athens. It reflects my interest in computer vision and my transition from law enforcement towards research in machine learning. The work focuses on motion prediction for autonomous driving and studies how to extend an existing transformer-based model in a careful and practical way.

1. INTRODUCTION

In recent years, autonomous driving has received growing attention from both academia and industry due to its potential to improve traffic safety, reduce congestion, and increase transportation efficiency. A modern self-driving system typically operates through a sequence of modules: perception, object tracking, motion prediction, and planning. Among these, motion prediction is critical, as it enables autonomous vehicles to anticipate the future behavior of surrounding road users and plan accordingly, much like how human drivers infer the intentions of others on the road.

Motion prediction refers to forecasting the future trajectories of agents such as vehicles, pedestrians, and cyclists based on their observed past behavior and interactions with the environment. Classical approaches to this task relied on physical motion models such as kinematics, Kalman filters, or Monte Carlo simulations [1, 2]. While computationally efficient, these models were often limited to short-term predictions and could not capture the uncertainty and complexity of real-world driving behavior.

Modern deep learning methods address these limitations by formulating motion prediction as a multi-agent, multi-modal forecasting task. These models typically rely on vectorized representations of the entire scene, including dynamic agents, static map features such as lanes and curbs, and semi-static elements such as traffic lights [3, 4]. Transformer-based architectures are then employed to capture the interactions between these heterogeneous components, enabling the system to reason about complex traffic scenarios and generate diverse possible futures [5, 6].

In this work, we build directly on the Heterogeneous Polyline Transformer (HPTR) [6] and study a density aware variant that we refer to as the **Spatial Density Transformer (SDT)**. We keep HPTR's overall encoder-decoder structure and feature encoders, and focus on modifying how attention is filtered. We add a unified Component Pairwise Encoding (CPE) that captures relative pose, relative motion, and type specific relationships, and a Spatial Density Module (SDM) that filters contextual information using a learned density prior. The SDM is used to bias the attention mechanism toward regions where interactions are more likely, while a multi modal decoding mechanism based on learnable anchors follows the design of HPTR. The goal of these changes is to explore how density aware attention can be integrated into an existing lightweight backbone for motion prediction without sacrificing its efficiency.

Code availability. The code for SDT used in this thesis is available at https://github.com/nekcht/sdt.

Throughout this thesis, the terms *trajectory prediction* and *motion prediction* are used interchangeably. Both refer to the task of forecasting the future states of agents in traffic scenes, with trajectory emphasizing the geometric path and motion emphasizing the predictive task as a whole.

1.1 Definition of a Trajectory

A trajectory is the path traced by an agent through space over time and can be formally represented as a sequence of states. Let $\mathbf{s}_t \in \mathbb{R}^d$ denote the state of an agent at time step t, where d is the dimensionality of the state space. A trajectory of length T is then defined as

$$\boldsymbol{\tau} = (\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_T).$$

The sequential nature of trajectories implies a temporal dependency among states, that is,

$$\mathbf{s}_t = f(\mathbf{s}_{<t}) \quad \text{for } t = 1, \ldots, T,$$

where f is an unknown function modeling the system dynamics and $\mathbf{s}_{<t}$ denotes the past states up to time $t-1$.

In autonomous driving, the agent can be a pedestrian, vehicle, or cyclist, and its state typically includes position coordinates in space. For planar motion, the state is often written as

$$\mathbf{s}_t = [x_t, y_t]^T,$$

or in more detailed form,

$$\mathbf{s}_t = [x_t, y_t, \theta_t, v_t, a_t, \omega_t]^T,$$

where x_t, y_t denote position, θ_t the heading angle, v_t the speed, a_t the acceleration, and ω_t the angular velocity.

Motion prediction aims to estimate the future states $(\mathbf{s}_{T+1}, \ldots, \mathbf{s}_{T+H})$ given the observed history $(\mathbf{s}_1, \ldots, \mathbf{s}_T)$, where H is the prediction horizon. This shows that motion prediction is inherently a sequence modeling problem, where temporal dependencies, agent dynamics, and scene context must all be considered [7].

1.2 Importance in Autonomous Driving

Motion prediction is a cornerstone of any autonomous driving system because it bridges perception and planning. Perception modules provide estimates of the current state of the world: the positions of vehicles, the presence of pedestrians and cyclists, and the status of traffic lights. Planning modules, in turn, are responsible for generating a safe and efficient path for the autonomous vehicle. Between these two stages lies motion prediction, whose role is to anticipate how other road users will move in the near future. Without reliable forecasts, the planning module has no stable foundation on which to reason about future safety or feasibility.

The criticality of motion prediction becomes clear when considering interactive traffic situations. At an intersection, the decision of whether to proceed depends not only on the current position of surrounding agents but also on their likely intentions. A vehicle approaching from the opposite direction may either continue straight or yield for a left turn; a pedestrian near a crosswalk may either wait or step into the road. Anticipating

such alternatives is essential to prevent collisions and to ensure smooth navigation. In highway scenarios, predicting lane changes or sudden braking maneuvers allows the self-driving car to adapt proactively rather than reactively, improving both safety margins and passenger comfort.

Equally important is the temporal horizon of prediction. While perception provides a snapshot of the present, planning must look ahead several seconds to evaluate potential trajectories of the ego vehicle. If motion forecasts are too short-sighted, the system may initiate maneuvers that later turn out to be unsafe. Conversely, long-horizon but unreliable predictions can mislead the planner. Thus, accurate motion prediction over a relevant horizon, typically three to eight seconds, is essential for effective decision-making.

Another aspect of importance lies in the social dimension of driving. Roads are shared spaces, and many maneuvers, such as merging, yielding, or negotiating priority at intersections, depend on implicit coordination between agents. A good prediction system captures not only kinematics but also intent, enabling the vehicle to behave in a way that aligns with human expectations. This is fundamental for trust: both passengers and surrounding road users must perceive the autonomous vehicle as a competent and socially compliant participant in traffic.

Finally, motion prediction has a direct impact on the reliability and acceptance of autonomous vehicles. Getting regulatory approval and earning public trust require clearly demonstrable safe behavior in various traffic conditions. By accurately modeling the uncertainty and multi-modality of human actions, motion prediction reduces the risk of unexpected collisions and provides interpretable outputs, such as alternative hypotheses with associated confidences, that planners can use for robust decision-making [7].

1.3 Challenges in Autonomous Driving

Motion prediction in autonomous driving faces multiple challenges that arise from uncertainty, heterogeneity, interactions, and strict computational requirements. A central difficulty is the inherent *multi-modality of human behavior*. In traffic, the same situation may lead to several plausible futures depending on hidden intent or reaction to other agents. To capture this uncertainty, the distribution over an agent's future trajectory is often modeled as a mixture of Gaussians [6]:

$$P\Big(\mathbf{s}^{(i)}_{T+1:T+H} \,\Big|\, \boldsymbol{\tau}^{(i)}, \mathcal{C}\Big) = \sum_{m=1}^{M} \pi^{(i)}_m \cdot \mathcal{N}\big(\mathbf{s}^{(i)}_{T+1:T+H} \,\big|\, \boldsymbol{\mu}^{(i)}_m, \Sigma^{(i)}_m\big).$$

Here, $\mathbf{s}^{(i)}_{T+1:T+H} \in \mathbb{R}^{H\times d}$ denotes the sequence of future states of agent i over horizon H, typically positions in $d = 2$ dimensions. The conditioning variables $\boldsymbol{\tau}^{(i)}$ and $\mathcal{C}$ represent, respectively, the past trajectory of agent i up to time T and the context

information such as the map or other agents' histories. The mixture has M components, each corresponding to a distinct possible future. The coefficient $\pi_m^{(i)} \geq 0$, with $\sum_{m=1}^{M} \pi_m^{(i)} = 1$, is the probability weight assigned to mode m, $\boldsymbol{\mu}_m^{(i)}$ is the mean trajectory for component m, and $\Sigma_m^{(i)}$ is the covariance matrix that encodes the uncertainty around that trajectory. This formulation explicitly acknowledges that multiple plausible continuations exist and allows the model to assign probability mass to several qualitatively different behaviors such as braking, turning, or continuing straight. Capturing this diversity is essential for safety, since averaging modes may yield infeasible paths that cut across lanes.

Another important challenge lies in *uncertainty estimation and calibration*. Beyond predicting trajectories, models must express confidence in their outputs. Overconfident predictors can underestimate risk, while underconfident ones may cause overly cautious planning. A calibrated predictor should satisfy

$$P(\mathbf{Y} \in \mathcal{E} \,|\, \hat{p}(\mathcal{E}) = q) \approx q,$$

where $\mathbf{Y}$ is the true outcome and $\mathcal{E}$ an event of interest. Here, $\hat{p}(\mathcal{E})$ denotes the predicted probability of $\mathcal{E}$ taking place, with $q \in [0, 1]$ the corresponding confidence level. This condition requires predicted probabilities to match observed frequencies, which is difficult to guarantee for structured outputs such as trajectories [7].

The driving scene is also *heterogeneous*, consisting of dynamic agents, semi-static traffic lights, and static map elements,

$$\mathcal{S} = \mathcal{A}_t \cup \mathcal{T}_t \cup \mathcal{M},$$

each with different properties and rules. Vehicles follow kinematic constraints, pedestrians move more freely, traffic lights switch among discrete states, and lane geometries impose semantic restrictions. A unified predictor must integrate all of these consistently.

A further challenge comes from *inter-agent interactions*. Vehicles rarely move independently; instead they must negotiate right-of-way, merges, and crossings. Formally, the prediction problem involves the joint distribution

$$P\left(\mathbf{Y}^{1:N} \,\middle|\, \mathbf{X}^{1:N}, \mathcal{M}\right),$$

where $\mathbf{Y}^{1:N}$ are the future trajectories of all N agents, $\mathbf{X}^{1:N}$ their observed histories, and $\mathcal{M}$ the scene context. The complexity of this distribution increases rapidly with N, making explicit enumeration of all dependencies infeasible. Practical models therefore rely on structured approximations or pruning strategies to capture the most relevant interactions [8, 9].

Predicted futures must also obey *geometric and semantic constraints*, such as staying within lane boundaries and respecting traffic signals. This requires tight coupling between motion prediction and HD maps [3]. Similarly, *short versus long-horizon forecasting* adds another layer of complexity: short-term motion is governed

by physics, but longer horizons depend on higher-level intentions and planned maneuvers, which are harder to infer.

In addition, rare but *safety-critical events* like sudden braking or jaywalking are severely underrepresented in datasets. Models biased toward frequent behaviors may miss such edge cases, even though they are crucial for safe deployment [7]. *Domain shift* is another issue: models trained on one region may fail in another due to cultural, infrastructural, or sensing differences, highlighting the difficulty of robust generalization.

Additionally, the system must satisfy *real-time constraints*. Prediction is embedded in the control loop of an autonomous vehicle, which leaves only a few tens of milliseconds per inference. This introduces a trade-off between prediction accuracy and computational cost, often formalized as

$$\min_{\theta} \quad \mathcal{L}_{\text{pred}}(\theta) + \lambda \cdot \mathcal{C}(\theta),$$

where $\mathcal{L}_{\text{pred}}$ is the forecasting error, $\mathcal{C}(\theta)$ measures latency or memory usage, and λ controls the balance.

1.4 Problem Formulation

Motion prediction in autonomous driving can be defined as the task of forecasting the future trajectories of traffic participants given their past states and the static and dynamic context of the scene. Let a traffic scene contain N dynamic agents indexed by $i \in \{1, \ldots, N\}$. The observed history of agent i up to time t is

$$\boldsymbol{\tau}_{1:t}^{(i)} = \left(\mathbf{s}_1^{(i)}, \ldots, \mathbf{s}_t^{(i)}\right),$$

where $\mathbf{s}_t^{(i)}$ denotes the state of agent i at time t, here represented by its position (x, y). In more detailed formulations, the state may additionally include heading θ, velocity v, acceleration a, or angular velocity ω. The prediction objective is to estimate the distribution of an agent's future states over a horizon H:

$$P\left(\mathbf{s}_{t+1:t+H}^{(i)} \,\middle|\, \boldsymbol{\tau}_{1:t}^{(i)}, \mathcal{C}\right),$$

where $\boldsymbol{\tau}_{1:t}^{(i)}$ is the observed history of agent i up to time t, and $\mathcal{C}$ denotes the scene context. The context typically comprises the HD map $\mathcal{M}$, the traffic light signals $\mathcal{T}_t$, and the trajectories of all other agents $\{\boldsymbol{\tau}_{1:t}^{(j)}\}_{j \neq i}$.

Several problem formulations have been proposed in the literature. The most widely adopted is the *marginal* formulation [4, 8, 10–12], where multi-modal future trajectories are predicted independently for each agent. This factorization reduces computational complexity and allows scalable deployment. In contrast, *joint* prediction [13–16] models the distribution of all agents' futures simultaneously, enabling

explicit reasoning about interactions such as yielding or merging at the cost of increased dimensionality and inference time. Beyond these open-loop formulations, which generate predictions from a fixed observation window without feeding intermediate predicted states back into the model to update subsequent predictions, *behavior simulation* [17–19] treats motion forecasting as a closed-loop process in which a learned policy is rolled out autoregressively [20], allowing predictions to adapt to feedback from evolving scene dynamics. More recent work also investigates coupling motion prediction with upstream or downstream modules, either by sharing features with perception and planning [21–24] or by learning end-to-end policies that map sensor measurements directly to driving actions [25, 26].

To capture the intrinsic uncertainty of human behavior, prediction is typically expressed as a multi-modal distribution. For agent i we write

$$P\Big(\mathbf{p}^{(i)}_{t+1:t+H} \mid \cdot\Big) = \sum_{m=1}^{M} \pi^{(i)}_m \, \mathcal{N}\big(\boldsymbol{\mu}^{(i)}_m, \Sigma^{(i)}_m\big),$$

where $\mathbf{p}^{(i)}_{t+1:t+H} \in \mathbb{R}^{2H}$ denotes the stacked sequence of future 2D positions over horizon H. Accordingly, $\boldsymbol{\mu}^{(i)}_m \in \mathbb{R}^{2H}$ is the mean position trajectory for mode m, and $\Sigma^{(i)}_m \in \mathbb{R}^{2H\times 2H}$ is its covariance (often implemented as block-diagonal with 2×2 blocks per time step). Each mode m corresponds to a plausible future maneuver (e.g., turning left, right, or going straight), parameterized by mixture weight $\pi^{(i)}_m$. In our model, additional kinematic quantities (e.g., yaw, speed, velocity) are predicted by separate heads and losses.

Formally, the task reduces to learning a mapping

$$f_\theta : \big(\{\boldsymbol{\tau}^{(i)}_{1:t}\}_{i=1}^{N},\ \mathcal{M},\ \mathcal{T}_t\big) \ \mapsto\ \{P(\mathbf{Y}^i)\}_{i=1}^{N},$$

parameterized by θ, where each $P(\mathbf{Y}^i)$ is a calibrated distribution over the possible future trajectories of agent i. An effective predictor must satisfy three criteria simultaneously: accuracy, by minimizing errors between predictions and ground truth; feasibility, by ensuring predicted paths conform to map geometry and traffic rules; and efficiency, by meeting the strict real-time constraints of an autonomous driving stack.

1.5 Scope and Contributions

This thesis focuses on the problem of multi agent motion prediction in complex urban environments, with particular emphasis on the *marginal forecasting* setting where multiple future hypotheses are predicted for each target agent individually. We restrict our evaluation to the Waymo Open Motion Dataset (WOMD), which offers large scale, interaction rich scenarios representative of real world driving [27].

Within this scope, the contributions can be summarized as follows. First, we define a density aware variant of HPTR, which we refer to as the **Spatial Density Transformer**

(SDT). This variant keeps the vectorized scene representation and multi modal decoding scheme of the HPTR's encoder-decoder architecture and introduces only localized changes related to attention filtering. Second, we propose the **Spatial Density Module (SDM)**, a mechanism that predicts Gaussian mixtures over the spatial domain and uses their normalized density to guide target selection and bias attention scores. This module is trained jointly with the rest of the network and is intended to provide a learned notion of where interactions are more likely to occur. Third, we integrate the SDM with a **Component Pairwise Encoding (CPE)** that combines relative pose and motion features, so that density estimation can depend both on static map geometry and on the dynamics of the agents involved. Finally, we evaluate SDT on the WOMD dataset and look at both standard benchmark metrics and internal diagnostics of the SDM. For the SDM we use simple measures such as component utilization, spatial variance, and entropy to get a sense of how the learned mixtures relate to the actual distribution of agents and map elements in the scene.

In these experiments, SDT reaches reasonable performance on the WOMD validation split, but we do not include a reimplementation of HPTR or other baselines trained under the same conditions, so the numbers are mainly used to characterize SDT itself. The SDM diagnostics suggest that the learned mixtures tend to place more mass in regions where tokens of interest (agents or map polylines) are present instead of empty areas, which is consistent with the goal of modeling interaction density.

2. FOUNDATIONS OF MOTION PREDICTION

2.1 Classical Physics-Based Models

Early approaches to motion forecasting relied on physical vehicle models that extrapolate future states from observed dynamics. The state of an agent is typically represented as

$$\mathbf{s}(t) = [x(t), y(t), \theta(t), v(t)]^\top,$$

where (x, y) are position coordinates, θ is the heading angle, and v the longitudinal velocity.

The most detailed formulation is the *dynamic bicycle model* [28], which accounts for mass distribution, tyre forces, and yaw inertia. While realistic, this model depends on parameters (cornering stiffness, load transfer, tyre saturation) that are rarely known in practice.

For this reason, simplified kinematic models are often preferred, such as the *kinematic bicycle model* [1]. Based on these models, three inference strategies became common: deterministic rollout (single predicted path), Kalman filtering (linear-Gaussian uncertainty propagation) [29], and Monte Carlo simulation (sampling controls or parameters to generate multi-modal outcomes) [30]. While computationally cheap, such forecasts rarely remain reliable beyond short horizons of ~ 1 s, as they cannot capture intent-driven changes, such as braking for pedestrians or yielding in traffic.

2.2 Classical Machine Learning Approaches

To overcome the rigidity of physics-only forecasting, classical machine learning methods introduced data-driven strategies.

Gaussian Processes (GPs) [30] encoded manoeuvre templates, predicting futures as mixtures of trajectory patterns conditioned on past observations. *Support Vector Machines (SVMs)* [31] classified manoeuvre classes (e.g. straight, left turn, right turn) from kinematic features, after which prototype trajectories were attached to the class label. *Hidden Markov Models (HMMs)* [2] and *Dynamic Bayesian Networks (DBNs)* [29] represented latent manoeuvre states with probabilistic transitions, producing multi-modal distributions by propagating state probabilities.

Instance-based methods such as *K-Nearest Neighbours* [32] or decision trees compared current motion histories with stored trajectories and extrapolated from the closest examples. These methods were conceptually simple but degraded quickly in sparsely sampled motion spaces.

While these approaches improved over physics-only methods by introducing data-driven multi-modality, they typically treated agents independently, relied on hand-

crafted manoeuvre labels, and either suffered from scalability bottlenecks (GPs, DBNs) or neglected uncertainty altogether (SVMs, KNNs). These shortcomings motivated the transition to deep learning.

2.3 Data Representations for Forecasting

A central design choice in motion forecasting is how to represent the scene from raw sensor inputs and high-definition (HD) maps. Two main paradigms dominate the literature: rasterised encodings and vectorised encodings.

Rasterised representations. In rasterisation, the environment is projected into a fixed-resolution bird's-eye-view (BEV) grid, where agent histories and static map features are rendered as multi-channel images. Convolutional neural networks (CNNs) [33] then extract spatial features from these tensors [8, 11]. Rasterisation is conceptually simple and benefits from mature image-based architectures, but introduces discretisation error, struggles with curved or fine-grained geometry, and scales poorly with resolution. Several works [23, 24] attempted to mitigate this by pre-computing static map features or using rotated region-of-interest alignment, reducing inference cost but still inheriting the limitations of gridded encodings.

Vectorised representations. The alternative, introduced by VectorNet [3], is to represent each spatial entity as a polyline:

$$\mathcal{L} = \big[(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\big],$$

capturing lane centerlines, crosswalks, or agent trajectories with exact geometric fidelity. Each polyline is embedded via lightweight encoders such as PointNet++ [34] and later aggregated using graph neural networks [35] or transformers [5]. Vectorised encodings preserve topology, reduce memory footprint, and align naturally with attention mechanisms.

Vectorisation has become the prevailing choice in modern benchmarks due to its efficiency and fidelity. In this work we also adopt a vectorised formulation, using pairwise-relative encodings to emphasise the structured nature of interactions.

2.4 Reference Frames

Another crucial design axis in motion forecasting is the choice of coordinate frame in which agent positions and map elements are expressed. The frame determines which invariances are built into the representation and how efficiently interactions can be modelled.

Scene-centric coordinates. In the simplest setting, all entities are expressed in a global map-aligned frame $\mathcal{W}$. Agent i at time t has position $\mathbf{p}_i(t) = (x_i(t), y_i(t))$ and

heading $\psi_i(t)$ expressed directly in $\mathcal{W}$. This enables a single, unified tensor representation for the whole scene and avoids recomputation for different targets. Rasterised pipelines in particular benefit from this global view [11]. The drawback is that the representation is *pose-variant*: a global rotation or translation changes the raw coordinates, forcing the model to learn invariances statistically. This often increases data requirements and reduces sample efficiency.

Agent-centric coordinates. To address invariance, many vectorised approaches [4, 36] transform the scene into the local body frame $\mathcal{B}_i$ of each target agent i. This frame is centred at the agent's current position $\mathbf{p}_i(t)$ and aligned with its heading $\psi_i(t)$. A point $\mathbf{q}$ in world coordinates is transformed as

$$\tilde{\mathbf{q}} = \mathbf{R}_i^\top (\mathbf{q} - \mathbf{p}_i), \qquad \mathbf{R}_i = \begin{bmatrix} \cos\psi_i & -\sin\psi_i \\ \sin\psi_i & \cos\psi_i \end{bmatrix}.$$

This formulation builds in translation and rotation invariance, meaning the network sees consistent patterns across different orientations. However, one local copy of the scene must be generated for each target agent, which can be expensive when many agents are considered simultaneously. Computational complexity grows with the number of targets, and multi-agent consistency is harder to enforce.

Pairwise–relative coordinates. An alternative is to encode only relative relations between pairs of entities (i, j). At time t we define

$$\Delta\mathbf{p}_{ij} = \mathbf{p}_j(t) - \mathbf{p}_i(t), \qquad \Delta\psi_{ij} = \mathrm{wrap}(\psi_j(t) - \psi_i(t)),$$

together with relative velocities and distances. These features are by construction invariant to global translation and rotation. Architectures such as GoRela and HiVT [36, 37] exploit this to encode interactions in a way that is independent of absolute scene orientation. The cost is that global map alignment can be lost unless additional features are injected, and the number of pairwise terms grows quadratically with the number of agents or polylines.

Hybrid strategies. Recent systems often combine these choices. LaneGCN [4] stores all polylines in a global map frame for efficiency but rotates local segments before processing. HiVT augments agent-centric encoders with a pairwise-relative decoder to model interactions. Such hybrids balance computational scalability with built-in invariances.

2.5 Prediction Paradigms

Two paradigms dominate motion prediction: marginal and joint.

Marginal prediction assumes factorisation across agents,

$$p(\mathbf{Y}^{1:N} \mid \mathbf{X}^{1:N}) \approx \prod_{i=1}^{N} p(\mathbf{Y}^i \mid \mathbf{X}^i, \mathcal{C}),$$

where $\mathcal{C}$ denotes shared context. This approach scales efficiently but ignores interaction consistency, often resulting in implausible collisions. In contrast, joint prediction models the full distribution

$$p(\mathbf{Y}^{1:N} \mid \mathbf{X}^{1:N}),$$

thereby capturing cooperative and competitive interactions explicitly. These forecasts are more realistic at intersections or during lane changes, but incur higher computational cost. Hybrid approaches attempt to balance both: marginal heads for distant agents and joint reasoning for a small set of interacting neighbours [7].

2.6 Deep Learning Architectures

Deep learning has significantly influenced motion prediction by learning the conditional distribution

$$p(\mathbf{Y} \mid \mathbf{X}, \mathcal{M}),$$

directly from large datasets, where $\mathbf{X}$ denotes agent histories and $\mathcal{M}$ encodes static map features. Instead of relying only on handcrafted manoeuvre templates or parametric dynamics, neural networks extract latent representations that capture temporal evolution, interaction structure, and scene semantics. Several families of architectures have been proposed.

Sequence models. Recurrent neural networks (RNNs) [38] and their gated variants, such as LSTMs [39] and GRUs [40], were the first to capture temporal dependence in agent trajectories. Given a history $\mathbf{X}^i_{1:T}$ of agent i, hidden states evolve as

$$\mathbf{h}^i_t = f_\theta(\mathbf{x}^i_t, \mathbf{h}^i_{t-1}),$$

with output $\hat{\mathbf{y}}^i_t = g_\theta(\mathbf{h}^i_t)$. Interaction cues were later incorporated via pooling mechanisms, e.g. Social-LSTM [41], which aggregates hidden states of nearby agents. While effective, RNNs suffer from vanishing gradients and limited parallelism, motivating alternative architectures.

Attention-based models. Transformers replace recurrence [5] with self-attention. For a set of agent embeddings arranged as a matrix $X \in \mathbb{R}^{n \times d_{\text{in}}}$, queries, keys, and values are formed as $Q = XW_Q,\ K = XW_K,\ V = XW_V$, where $W_Q \in \mathbb{R}^{d_{\text{in}} \times d}$, $W_K \in \mathbb{R}^{d_{\text{in}} \times d}$, and $W_V \in \mathbb{R}^{d_{\text{in}} \times d_v}$ are learned projection matrices mapping input embeddings to queries $Q \in \mathbb{R}^{n \times d}$, keys $K \in \mathbb{R}^{n \times d}$, and values $V \in \mathbb{R}^{n \times d_v}$. The embeddings are then updated via

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right) V.$$

This mechanism allows long range interactions across agents and map polylines, and can be stacked to form multi layer reasoning modules. Models such as Scene Transformer [42] and Wayformer [43] showed that attention scales naturally with vectorised scene encodings and has become a widely used approach in motion prediction.

Graph neural networks. An alternative view represents the traffic scene as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with nodes for agents and lane segments and edges encoding their relations. Message passing updates node features by

$$\mathbf{h}_v^{(l+1)} = \phi\Big(\mathbf{h}_v^{(l)}, \sum_{u \in \mathcal{N}(v)} \psi(\mathbf{h}_v^{(l)}, \mathbf{h}_u^{(l)}, \mathbf{e}_{uv})\Big),$$

where $\mathbf{e}_{uv}$ denotes an edge feature (or relation embedding) capturing the interaction between nodes u and v, such as their relative position, distance, heading difference, or lane connectivity. VectorNet [3] pioneered this approach with hierarchical polyline encoders, while LaneGCN [4] augmented GNN layers with lane-structured convolutions. GNNs [35] capture local topology and directional constraints, though their efficiency is often lower than transformers.

Probabilistic and generative models. Human driving behaviour is inherently uncertain and multi-modal. To capture this, probabilistic models introduce latent variables z with

$$p(\mathbf{Y} \mid \mathbf{X}) = \int p(\mathbf{Y} \mid z, \mathbf{X})\, p(z \mid \mathbf{X})\, dz.$$

Conditional VAEs (e.g. DESIRE [8]) instantiate z as a manoeuvre code sampled from an encoder network. Generative Adversarial Networks (GANs) [44] have been explored but are difficult to train stably. More recently, diffusion models generate futures by reversing a stochastic process, producing highly diverse samples with strong coverage of rare behaviours. These approaches yield calibrated multi-modality, crucial for safety-critical planning.

Hybrid architectures. State-of-the-art systems often combine multiple elements. For example, VectorNet fuses polyline encoders with GNN reasoning [3], LaneGCN alternates graph convolutions and attention [4], and Wayformer leverages transformers with hierarchical context [43]. Goal-conditioned methods such as DenseTNT [45] introduce interpretable intermediate anchors (goals) that structure prediction. These hybrids reflect a broader trend toward architectures that try to unify representation learning, interaction reasoning, and uncertainty modeling in a single pipeline.

3. MODEL INTRODUCTION

We introduce the **Spatial Density Transformer (SDT)**, a density-aware attention based model for multi-agent motion prediction that builds directly on HPTR [6]. The key idea is that dense attention across all tokens can be inefficient, since in many scenes only a subset of interactions has a strong influence on future motion. To address this, SDT incorporates a *Spatial Density Module (SDM)* that predicts a grid aligned mixture of Gaussians over the scene. These densities are used in two complementary ways. First, they provide a *top-k* selection that prunes many targets, making attention sparse. Second, their log probabilities are injected as additive biases into the attention logits, so that the retained interactions are weighted according to spatial relevance. The goal of this design is to reduce computational cost compared to dense neighborhoods while keeping predictive performance close to that of the HPTR backbone.

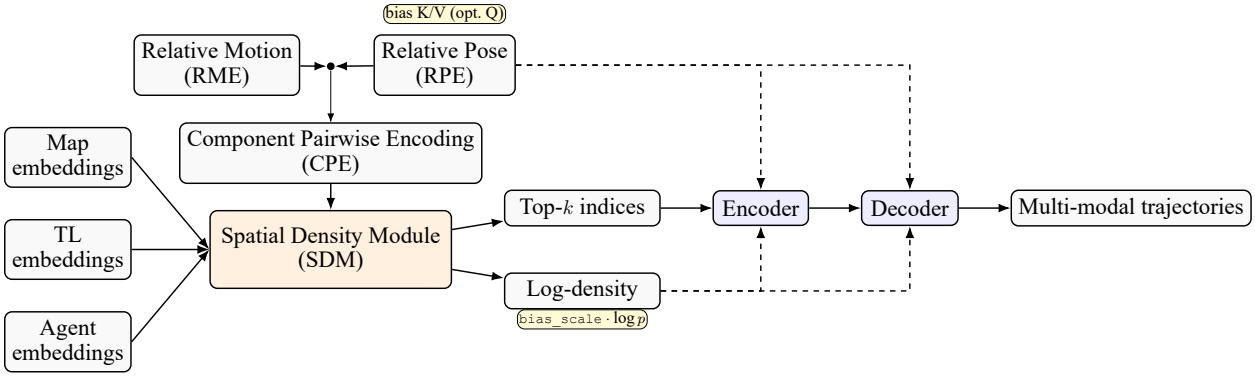


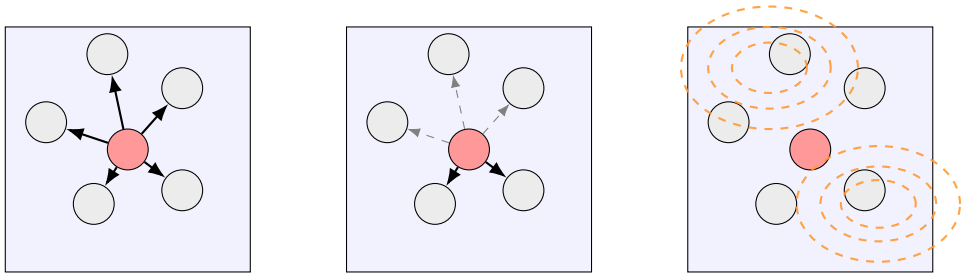
Figure 3.1: Overview of SDT. Map, traffic light and agent embeddings feed the SDM, which predicts Gaussian mixtures. From these, top-k selections restrict attention to a subset of targets, while log densities act as attention biases. RPE and RME are fused into CPE for density estimation; RPE additionally biases transformer attention directly.

The model processes three token types: map polylines, traffic light states, and dynamic agents. Each is embedded through lightweight encoders that follow the HPTR design. Pairwise structure is introduced through two encodings: *Relative Pose Encoding (RPE)*, which describes geometric offsets such as displacement and heading, and *Relative Motion Encoding (RME)*, defined only between agents to capture velocity and acceleration cues. These are fused into a joint *Component Pairwise Encoding (CPE)*, which augments the SDM with motion aware signals. RPE also biases attention directly by modulating key and value projections, as in HPTR.

The SDM outputs mixtures of Gaussians over the map, assigning probability to regions of interest. Evaluating these densities at candidate targets provides scores used both for pruning (top-k) and for biasing attention logits. The encoder then processes the resulting sparse neighborhoods, and the decoder attends to this refined context to generate multi-modal trajectory hypotheses anchored in learned prototypes, each with an associated confidence. In this way, SDT combines geometric structure, dynamic

cues, and spatial density in order to focus attention on a smaller set of interactions.

Motivation. Dense attention across all tokens scales quadratically with the number of scene elements. In crowded intersections or highways, this can be costly and may also overwhelm the model with interactions that have little effect on the future motion of the target agent. Existing heuristics, such as selecting a fixed set of nearest neighbors, are efficient but brittle because they may discard important but slightly more distant agents, such as a pedestrian about to cross. SDT aims to address this by predicting spatial densities that adapt dynamically to the scene, so that the model can keep a fixed budget of neighbors while still paying attention to regions that look important according to the learned density. This direction is motivated by real time constraints in autonomous vehicles, although a full system level real time evaluation is outside the scope of this thesis.



(a) Dense Attention (b) HPTR: Fixed k-NN (c) SDT: Density-Aware

Figure 3.2: Comparison of interaction selection. (a) Dense attention attends to all targets. (b) HPTR restricts neighborhoods using a fixed k-nearest neighbor rule in Euclidean space (here k=2). (c) SDT predicts spatial densities and selects top-k interactions dynamically (the densities isovalue curves are denoted by the dashed, orange-coloured lines).

Relation to HPTR and prior work. The proposed model is most closely related to the Heterogeneous Polyline Transformer (HPTR). Both architectures operate on vectorized inputs and use relative pose encodings to inject geometric structure into attention. HPTR restricts neighborhoods with a fixed k nearest neighbor rule in Euclidean space. In this thesis we keep the same general setting and build on the HPTR design, but replace the heuristic k nearest selection with a density-based mechanism driven by the Spatial Density Module (SDM). Instead of choosing neighbors only by Euclidean distance, the SDM predicts Gaussian mixtures that highlight regions where interactions are likely to occur, and these mixtures are used for differentiable top$k$ selection and as a bias term in the attention logits. In practice this means that sparse attention is guided by a learned probabilistic prior rather than by a hand crafted rule. Large transformer models such as **Wayformer** [43] or staged refinement models such as **MTR++** (Motion Transformer) [46] explore a different direction that focuses on higher capacity and stronger benchmark performance. Here, the goal is more modest: to study how a density-aware filtering mechanism behaves inside an HPTR style backbone, under a limited training budget, rather than to propose a new state of the art system.

Interpretability and broader impact. A useful side effect of the SDM is that it exposes its internal relevance estimates as Gaussian fields over the map. These fields can be visualized and inspected as an additional attention map that shows which agents, lanes, or regions are assigned higher density. Such qualitative plots can be helpful for understanding typical behaviors of the model and for debugging specific failure cases, although a full user study or safety analysis is outside the scope of this work. In this thesis SDT is evaluated only on the Waymo Open Motion Dataset, but the basic idea of density guided sparse attention is generic and could, in principle, be adapted to other driving datasets or related domains, such as pedestrian tracking or crowd simulation. A careful investigation of such extensions and their impact on downstream planning or safety is left for future work.

4. RELATED WORK

Motion prediction for autonomous driving has progressed from rule based and physics driven models toward data driven architectures capable of representing complex interactions and multi modal futures. Classical methods such as constant velocity or constant turn rate models offered computational efficiency but were unable to anticipate intent driven behaviors. Early machine learning approaches introduced Gaussian Processes, Hidden Markov Models, and Support Vector Machines, which leveraged trajectory data to recognize maneuver classes or latent behavioral states. While these methods improved short horizon forecasting, they generally treated agents independently and scaled poorly to dense urban traffic.

The first wave of deep learning methods represented the scene in rasterized bird's eye view form [8, 11]. Agent histories and high definition maps were projected onto image grids and processed with convolutional neural networks. This formulation benefited from mature 2D architectures but incurred heavy memory cost and introduced discretization errors, blurring fine lane geometry and map topology.

Aspect	Rasterized Encodings	Vectorized Encodings
Representation	Scene rendered into bird's eye view images with multiple channels (agents, map, time layers).	Scene expressed as polylines (ordered point sequences) for lanes, agents, and map elements.
Geometric fidelity	Limited by grid resolution, curved geometry blurred.	Exact lane topology and geometry preserved.
Memory footprint	High: scales with image size $H \times W \times C$.	Low: scales with number of polylines and nodes.
Architectures	Leverages mature CNN backbones, pretrained image models.	Uses PointNet/MLP encoders, GNNs, transformers.
Scalability	Raster size fixed, robust to number of agents.	Complexity grows with number of polylines, but efficient for sparse scenes.
Context modeling	Global context easily captured within receptive field.	Requires explicit relational reasoning (attention, message passing).
Interpretability	Less transparent; information blurred in pixels.	More interpretable: polylines directly correspond to map elements and agent paths.

Table 4.1: Comparison of rasterized and vectorized scene encodings in motion forecasting. Rasterized methods leverage 2D CNNs but suffer from discretization; vectorized encodings preserve geometry and align naturally with attention based architectures.

A shift came with vectorized encodings, pioneered by **VectorNet** [3], which decomposed maps and agent trajectories into polylines embedded with lightweight MLPs. This preserved geometric fidelity while reducing memory, and the paradigm has since become very common. Extensions include **Wayformer** [43], which showed that transformers with structured attention can scale effectively to production scale

datasets, and **DenseTNT** [45], which introduced goal conditioned forecasting by enumerating candidate endpoints over the drivable surface and conditioning trajectory generation on selected goals. Other scalable approaches, such as **MTR++** (Motion Transformer) [46] refine candidate sets stage by stage, progressively narrowing the hypothesis space, utilizing task-specific attention mechanisms.

Since future motion is inherently uncertain, capturing multi modality has been central. Variational formulations such as DESIRE [8] explicitly parameterize mixtures of latent trajectory generators. More recent work explores diffusion based forecasters and normalizing flow models, which learn expressive generative distributions that can produce diverse futures consistent with scene context. These approaches emphasize calibrated probabilities, which are important for downstream risk sensitive planning.

Beyond open loop forecasting, several works explore behavior simulation [17–19], where learned policies are rolled out in closed loop to generate long horizon trajectories. Others investigate joint formulations of perception, prediction, and planning [21–23], or end to end driving policies [25, 26] that map sensor input directly to actions. While promising, such approaches are computationally demanding and remain challenging to validate for safety in real deployments.

5. METHOD

This chapter describes the components of the **Spatial Density Transformer (SDT)** and how they integrate into a complete motion forecasting system. Section 5.1 introduces the polyline representation used to convert raw scene elements into a unified vector format. Section 5.2 defines the Component Pairwise Encoding (CPE), which augments geometric relations with motion cues. Section 5.3 presents the Spatial Density Module (SDM), a Gaussian mixture mechanism that aims to highlight more relevant interactions. Section 5.4 then introduces the Spatial Density Attention (SDA), which integrates SDM outputs into the transformer by combining density guided top-k selection with log probability biases. Section 5.5 explains how the encoder updates agents, map elements, and traffic lights using this density-aware attention. Section 5.6 details the multi-modal decoder that generates multiple future trajectory hypotheses. Finally, Section 5.7 summarizes the training objectives, evaluation metrics, and diagnostics of the SDM.

5.1 Polyline Representation and Embedding

We adopt the polyline based representation introduced by VectorNet [3] and extended by HPTR [6]. Each scene element, such as agents, lanes, and traffic lights, is expressed as a polyline, i.e. an ordered sequence of nodes. To remove dependence on the global frame, all coordinates are transformed into a local system: for agents this is centered at the current position and aligned with heading, while for map and traffic light polylines it is anchored at the first node.

Focusing on a single agent, given node positions $\mathbf{p}_t \in \mathbb{R}^2$ and headings θ_t, the local transform is

$$\mathbf{p}_t^{\text{local}} = \mathbf{R}^\top(\mathbf{p}_t - \mathbf{p}_0), \qquad \theta_t^{\text{local}} = \theta_t - \theta_0,$$

with $\mathbf{R}$ the rotation matrix from θ_0. We encode angles via sine/cosine pairs to preserve periodicity and normalize distances by a fixed length scale for numerical stability. Following the relative polyline encoding used in HPTR, we construct a geometric descriptor $\mathbf{g}_t \in \mathbb{R}^7$ from the local polyline node position and direction. Let $\mathbf{d}_t \in \mathbb{R}^2$ denote the local direction (for agents, $\mathbf{d}_t = [\cos\theta_t^{\text{local}}, \sin\theta_t^{\text{local}}]^\top$). We compute the closest point to the origin along the segment starting at $\mathbf{p}_t^{\text{local}}$ with direction $\mathbf{d}_t$ as

$$\alpha_t = \min\left(1, \max\left(0, \frac{-\langle \mathbf{p}_t^{\text{local}}, \mathbf{d}_t \rangle}{\|\mathbf{d}_t\|^2}\right)\right), \qquad \mathbf{c}_t = \mathbf{p}_t^{\text{local}} + \alpha_t \mathbf{d}_t, \qquad r_t = \|\mathbf{c}_t\|.$$

and define

$$\mathbf{g}_t = \left[r_t,\ \frac{\mathbf{c}_t}{r_t},\ \frac{\mathbf{d}_t}{\|\mathbf{d}_t\|},\ \|\mathbf{d}_t\|,\ \left\|\mathbf{p}_t^{\text{local}} + \mathbf{d}_t - \mathbf{c}_t\right\| \right] \in \mathbb{R}^7.$$

Finally, we concatenate $\mathbf{g}_t$ with raw attributes $\mathbf{a}_t$ to form the node feature $\mathbf{x}_t = [\mathbf{g}_t; \mathbf{a}_t]$. For agent nodes, $\mathbf{a}_t$ consists of local velocity (v_x, v_y), speed, yaw, acceleration, size

(length, width, height), a one-hot encoding of semantic type over three classes (vehicle, pedestrian, cyclist), and the (scene frame) position (x, y), giving $\mathbf{a}_t \in \mathbb{R}^{13}$ and thus $\mathbf{x}_t \in \mathbb{R}^{20}$.

A lightweight PointNet encoder applies shared MLPs to $\mathbf{x}_t$ and aggregates along the polyline using max pooling:

$$\mathbf{h}_t = \phi(\mathbf{x}_t), \qquad \mathbf{u} = \max_t \mathbf{h}_t,$$

producing one fixed length embedding $\mathbf{u}$ per polyline. Max pooling ensures permutation invariance along the nodes while retaining the most salient local features. The resulting set $\{\mathbf{u}_i\}$ over all agents, forms the input token collection for relational reasoning.

5.2 Relative Encodings

Attention is permutation invariant unless provided with positional structure. We inject pairwise geometric and motion cues via Relative Pose Encoding (RPE) and Relative Motion Encoding (RME). For entities i and j with poses (x, y, θ), the relative positional encoding (RPE) is defined as

$$\mathrm{RPE}_{ij} = \Big(x_j - x_i,\ y_j - y_i,\ \sin(\theta_j - \theta_i),\ \cos(\theta_j - \theta_i)\Big).$$

This representation captures the displacement in position together with the relative orientation between the two entities. RPE is computed for all pairs of elements, such as agents, map segments, and traffic lights, and serves to bias the attention mechanism toward locality awareness by providing explicit geometric relations.

For dynamic interactions, geometry alone is insufficient. Let s_i be speed, $v_i \in \mathbb{R}^2$ velocity, $a_i \in \mathbb{R}^2$ acceleration, and $p_i \in \mathbb{R}^2$ position of agent i. For agent pairs (i, j) we use three motion cues:

$$o_{ij} = \sigma\big(\beta\,(s_i - s_j)\,\cos\angle(v_i, v_j)\big), \quad d_{ij} = \tfrac{1}{2}(1 - \cos\angle(a_i, a_j))\,,$$

$$\Delta p_{ij} = p_j - p_i,\ \ \Delta v_{ij} = v_j - v_i,\ \ \mathrm{roc}_{ij} = \frac{\Delta p_{ij} \cdot \Delta v_{ij}}{\|\Delta p_{ij}\|},\ \ a_{ij} = 2\big(\sigma(\gamma\ \max(0, -\mathrm{roc}_{ij})) - \tfrac{1}{2}\big),$$

where σ is the logistic sigmoid and $\beta, \gamma > 0$ are user-defined scaling parameters (inverse-temperature like) controlling the sharpness of the sigmoid responses. The triplet (o_{ij}, d_{ij}, a_{ij}) respectively reflects overtaking likelihood, intent divergence from acceleration alignment, and approach score from normalized closing rate. For pairs with equal (near-zero) speed and approximately zero velocity and acceleration, these terms are set to zero.

We fuse these into the Component Pairwise Encoding (CPE),

$$\mathrm{CPE}_{ij} = \big[\mathrm{RPE}_{ij},\ o_{ij},\ d_{ij},\ a_{ij},\ \delta_j^{\text{motion}}\big],$$

where $\delta_j^{\text{motion}} \in \{0, 1\}$ flags dynamic targets. CPE is used exclusively inside the SDM to inform density with both geometry and motion. In parallel, RPE additionally biases transformer attention directly (Section 5.4).

5.3 Spatial Density Module

The SDM predicts, for each source token i, a mixture of Gaussians over a regular grid that tessellates the scene bounds. Let the grid have resolution $R \times R$ (cells indexed by $m = 1, \ldots, R^2$) and let each component have weight π_{im}, mean $\mu_{im} \in \mathbb{R}^2$ constrained within its cell, and diagonal covariance $\Sigma_{im} = \text{diag}(\sigma_{x,im}^2, \sigma_{y,im}^2)$ whose scales are bounded relative to the cell size. The spatial density conditioned on i is

$$p(x \mid i) = \sum_{m=1}^{R^2} \pi_{im}\, \mathcal{N}\big(x;\, \mu_{im}, \Sigma_{im}\big), \qquad \sum_m \pi_{im} = 1, \;\; \pi_{im} \geq 0.$$

To couple density with context, we augment each source embedding with a permutation-invariant summary of its pairwise context pose encodings. Concretely, for a source element i we have $\text{CPE}_{ij} \in \mathbb{R}^{d_{\text{cpe}}}$ for each target element j describing j relative to i. Since the number of valid targets varies per scene, we pool across j using a masked mean

$$\bar{\mathbf{c}}_i = \frac{\sum_j m_j \, \text{CPE}_{ij}}{\sum_j m_j} \in \mathbb{R}^{d_{\text{cpe}}},$$

where $m_j \in \{0, 1\}$ indicates whether target j is valid. We then concatenate this pooled context vector to the source embedding and feed the result to the density head,

$$\mathbf{z}_i = [\mathbf{h}_i; \bar{\mathbf{c}}_i], \qquad \{\pi_{im}, \mu_{im}, \Sigma_{im}\} = f_{\text{dens}}(\mathbf{z}_i),$$

where f_{dens} is a lightweight multi-layer perceptron with three output heads that predict mixture logits for π, bounded offsets for μ, and bounded log standard deviations that parameterize Σ. This preserves a clear spatial interpretation: each component behaves like a localized focus field whose support roughly matches a map patch.

Given candidate target positions $\{x_j\}$, the SDM provides two signals. First, it yields log densities

$$\ell_{ij} \;=\; \log p(x_j \mid i),$$

which are added to attention logits as a density bias (Section 5.4). Second, it enables top-k target selection per source by ranking $\{\ell_{ij}\}_j$. We use a straight through estimator (STE) with Gumbel perturbations [47] to maintain discrete top-k neighborhoods in the forward pass while allowing gradients to flow to the continuous scores during backpropagation. This dual role makes SDM both a selector and a soft prior over attention.

Computationally, replacing dense neighborhoods with top-k reduces the per layer cost from $O(M^2 d)$ to $O(Mkd)$ (M tokens, head dimension d), with $k \ll M$. The density

bias then concentrates probability mass on targets that receive higher density scores among those retained. This is intended to improve efficiency by focusing computation on a smaller set of candidates while still keeping attention aligned with semantically relevant regions.

5.4 Spatial Density Attention

Attention is computed with multi head scaled dot products, augmented by (i) relative pairwise encodings injected into keys and values, and (ii) SDM log densities added as external biases. Let n_s be the number of source elements (queries), n_t the number of target elements (keys and values), h the number of heads, and d the per head key/query dimension. For each head, the projected tensors are

$$Q \in \mathbb{R}^{h \times n_s \times d}, \qquad K \in \mathbb{R}^{h \times n_t \times d}, \qquad V \in \mathbb{R}^{h \times n_t \times d_v},$$

(where typically $d_v = d$). In self-attention, $n_s = n_t$, while in cross-attention n_s and n_t may differ. Let r_{ij} denote the relative pairwise encoding (RPE) between elements i and j. Linear maps project r_{ij} to additive corrections for keys and values:

$$\tilde{K} = K + \Phi_K(r), \qquad \tilde{V} = V + \Phi_V(r),$$

with $\Phi_K(r) \in \mathbb{R}^{h \times n_s \times n_t \times d}$ and $\Phi_V(r) \in \mathbb{R}^{h \times n_s \times n_t \times d_v}$ broadcasting to match K and V per (i, j) pair. Unnormalized attention logits are

$$L = \tfrac{1}{\sqrt{d}}\, Q\tilde{K}^\top \in \mathbb{R}^{h \times n_s \times n_t}.$$

The SDM provides a log density field aligned with the source target layout; we add it to the logits with a scalar bias scale $\beta > 0$:

$$\hat{L} = L + \beta \log p.$$

Attention weights and outputs are

$$A = \mathrm{softmax}(\hat{L}) \in \mathbb{R}^{h \times n_s \times n_t}, \qquad O = A\tilde{V} \in \mathbb{R}^{h \times n_s \times d_v}.$$

RPE sharpens discrimination among geometrically plausible matches, while the SDM bias tilts attention toward higher density regions identified by the density model. Because $\log p$ enters before the softmax, gradients propagate from attention back into SDM, aligning thus learned densities with the patterns the transformer finds useful. The bias scale β controls the balance: $\beta \to 0$ recovers content driven attention; larger β puts more weight on density-aware focusing.

5.5 Encoder

The encoder updates embeddings separately for agents, lanes, and traffic lights. Within each class, attention is restricted to the top-k targets selected by SDM, with logits biased by the corresponding log densities. Writing the per head logits as

$$L = \tfrac{1}{\sqrt{d}} QK^\top + \beta \log p,$$

the block applies softmax normalization, value aggregation with the RPE modulated $\tilde{V}$, and standard residual norm MLP layers. Class specific stacks reflect the distinct roles of each token type: lane geometry refines static context, traffic light embeddings capture signal state, and agent embeddings integrate peer interactions. This structure results in sparse, geometry aware, and density guided updates by construction.

5.6 Decoder

For each agent, the decoder produces a small set of trajectory hypotheses with associated confidences. We use learnable anchors $\{a^{(m)}\}_{m=1}^{n_{\text{pred}}}$ that act as maneuver prototypes (e.g. go straight, turn left, yield). Given the final encoder embedding h_i, a lightweight fusion $\Phi(h_i, a^{(m)})$ forms mode specific inputs $E^{(m)}$. Two prediction heads operate on $\{E^{(m)}\}$: a trajectory head and a confidence head.

The trajectory head outputs, for each mode m and step t, a vector

$$\hat{y}_t^{(m)} = \big(\mu_t^{(m)}, \sigma_t^{(m)}, \rho_t^{(m)}, \hat{s}_t^{(m)}, \hat{v}_t^{(m)}, \hat{\psi}_t^{(m)}\big),$$

where (μ, Σ) parameterize a bivariate Gaussian over 2D position (x_t, y_t) and (s, v, ψ) denote kinematics. The covariance is

$$\Sigma_t = \begin{bmatrix} \sigma_{x,t}^2 & \rho_t \sigma_{x,t}\sigma_{y,t} \\ \rho_t \sigma_{x,t}\sigma_{y,t} & \sigma_{y,t}^2 \end{bmatrix}, \qquad |\rho_t| < 1.$$

The confidence head outputs logits $\hat{c}^{(m)}$ normalized across modes to form a categorical distribution over hypotheses. An optional ensemble of decoders can be used to increase diversity by concatenating their mode sets at inference.

Anchors encourage explicit multi-modality and stabilize training by providing a consistent basis across scenes. Because SDM prunes part of the context and biases attention, the anchor fusion operates on density refined embeddings, which is intended to help separate different maneuver hypotheses.

Practical simplifications. The original HPTR decoder includes an all-to-all attention layer that jointly refines mode embeddings across agents and uses caching to avoid recomputing some intermediate encodings. In our implementation we removed

this global decoder attention, since for large batch sizes it exceeded the memory limits of the available 12 GB GPU. We also disabled the caching logic and always recompute the required encodings inside the decoder. These changes keep the basic anchor based decoding scheme, but result in a slightly simpler and more lightweight decoder than the one described in the HPTR paper. All results in this thesis refer to this simplified variant.

5.7 Training Loss and Evaluation Metrics

Training follows a hard assignment strategy. For each agent, the predicted mode with lowest average displacement error to the ground truth is selected:

$$m^\star = \arg\min_m \tfrac{1}{T} \sum_{t=1}^{T} \|p_t - \mu_t^{(m)}\|_2.$$

Regression losses are applied only to this mode. For positions we use the negative log likelihood under the predicted Gaussian,

$$\mathcal{L}_{\text{pos}} = -\tfrac{1}{T} \sum_{t=1}^{T} \log \mathcal{N}(p_t \mid \mu_t^{(m^\star)}, \Sigma_t^{(m^\star)}).$$

Yaw is also periodic, so we model it with a von Mises negative log likelihood,

$$\mathcal{L}_\psi = -\tfrac{1}{T} \sum_{t=1}^{T} \log \left(\frac{\exp\big(\kappa \cos(\psi_t - \hat{\psi}_t^{(m^\star)})\big)}{2\pi I_0(\kappa)} \right),$$

with concentration $\kappa > 0$ and modified Bessel function $I_0(\cdot)$. Other circular likelihoods could also be employed. Speeds and velocities use Huber losses for robustness:

$$\mathcal{L}_s = \tfrac{1}{T} \sum_{t=1}^{T} \text{Huber}(s_t - \hat{s}_t^{(m^\star)}), \qquad \mathcal{L}_v = \tfrac{1}{T} \sum_{t=1}^{T} \|v_t - \hat{v}_t^{(m^\star)}\|_\delta.$$

Confidence is trained with cross entropy,

$$\mathcal{L}_{\text{conf}} = -\log \frac{\exp(\hat{c}^{(m^\star)})}{\sum_m \exp(\hat{c}^{(m)})},$$

and the total objective is

$$\mathcal{L} = \mathcal{L}_{\text{pos}} + \mathcal{L}_\psi + \mathcal{L}_s + \mathcal{L}_v + \mathcal{L}_{\text{conf}}.$$

We evaluate using the WOMD validation metrics: mAP, Min ADE, Min FDE, Miss Rate, and Overlap Rate. In addition, we monitor SDM diagnostics to understand how density evolves and guides attention. For the mixture

$$p(x \mid i) = \sum_{k=1}^{K} \pi_{ik} \, \mathcal{N}(x; \mu_{ik}, \Sigma_{ik}),$$

we log entropy

$$H_i = -\sum_{k=1}^{K} \pi_{ik} \log \pi_{ik},$$

utilization mass above a density threshold ϵ,

$$U_i = \mathbb{P}_{x \sim p(\cdot|i)}\big(p(x \mid i) > \epsilon\big),$$

that is, the probability mass of samples drawn from $p(\cdot \mid i)$ whose density exceeds the threshold. In other words, U_i measures how much of the distribution lies in regions of relatively high density. Finally, normalized variances

$$\sigma_{ik}^{\text{norm}} = \frac{\frac{1}{2}\,\text{tr}(\Sigma_{ik})}{\Delta^2},$$

where Δ is the grid cell size. In practice, H_i tracks mode concentration, U_i reflects sparsity of the spatial prior, and $\sigma_{ik}^{\text{norm}}$ measures component sharpness. These statistics are diagnostic (not optimized) and help interpret how SDM focuses attention during training.

6. EXPERIMENTS

6.1 Experimental setup

We conduct all experiments on the *Waymo Open Motion Dataset (WOMD)*[1]. At the raw-data level, WOMD contains 103,354 mined driving segments, each 20 seconds long and sampled at 10 Hz, amounting to about 574 hours of driving over roughly 1,750 km of roadway across six U.S. cities: San Francisco, Phoenix, Mountain View, Los Angeles, Detroit, and Seattle. The official release partitions these raw segments into training, validation, and test splits in a 70%/15%/15% ratio. The benchmark contains 486,995 training scenarios, 44,097 validation scenarios, and 44,920 test scenarios.

Each training and validation scenario contains 10 history frames, 1 current frame, and 80 future frames, for a total of 91 samples, while the test split exposes only the 11 observed samples and withholds future trajectories for leaderboard evaluation. Each frame provides 3D bounding boxes and persistent track IDs for vehicles, pedestrians, and cyclists, together with HD map elements such as lane centerlines, lane boundaries, road boundaries, crosswalks, stop signs, speed bumps, driveway entrances, and dynamic traffic-light states. WOMD is designed for behavior prediction in highly interactive traffic situations, including merges, lane changes, unprotected turns, and pedestrian crossings, and is therefore widely used for both marginal and joint motion forecasting.

The task is defined as marginal motion prediction: for each scenario, the model must generate six future trajectories for up to eight target agents. Each trajectory spans 80 steps into the future, corresponding to 8 seconds at a sampling interval of 0.1 seconds. The observation history length is fixed at 11 steps (1.1 seconds). We follow the official evaluation protocol and compute metrics using the WOMD leaderboard evaluation tool, where the ranking metric is the soft mean average precision (soft mAP). Additional metrics such as minADE and minFDE are also reported for better accuracy assessment.

6.2 Implementation details

Our model is implemented in PyTorch Lightning and trained on a single NVIDIA GeForce RTX 4070 (12 GB). For each scenario we consider up to $N_{\text{map}} = 1024$ map polylines, $N_{\text{tl}} = 40$ traffic light elements, and $N_{\text{ag}} = 64$ agents. Each polyline is represented by up to $N_{\text{node}} = 20$ one meter segments. The Spatial Density Module operates on a discretized grid with resolution $R \times R$, where we set $R = 4$ unless otherwise spec-

[1] S. Ettinger *et al.*, "Large-Scale Interactive Motion Forecasting for Autonomous Driving: The Waymo Open Motion Dataset," *ICCV*, 2021.

ified. For prediction we follow the WOMD protocol and produce $N_{\text{pred}} = 6$ multi-modal futures per agent.

The transformer backbone employs pre layer normalization and is trained with the AdamW optimizer and cosine learning rate decay. Dropout is applied at a rate of 0.1 in attention and feed forward layers. No ensembling or trajectory aggregation is applied at inference time; only confidence scores are adjusted through a light-weight post processing step.

Differences to the original HPTR implementation. Due to memory constraints, our implementation does not include the full decoder used in HPTR. In particular, we removed the all-to-all attention layer at the decoder stage and turned off the feature caching mechanism that avoids recomputing some intermediate encodings. These changes were necessary to keep training within the 12 GB GPU budget and mainly affect how much cross interaction the decoder can model and how much computation is reused. The encoder architecture, the use of relative pose encodings, and the anchor based prediction heads follow the HPTR design.

6.3 Evaluation under different SDM settings

We study how three design choices affect SDT: the Component Pairwise Encoding (CPE), the strength of the spatial density log bias (`bias_scale` $\in \{0.5, 1.0\}$), and the spatial grid resolution of the Spatial Density Module (`grid_size` $\in \{4, 9, 16\}$). Each configuration is trained with the same short schedule on WOMD. For each one we measure validation *minADE*, *minFDE*, *mAP*, and some gradient norm statistics to see how stable training is. We mainly look at *mAP* to rank the runs, but because the training budget is small, these results should be seen as indicative and not as final.

From Table 6.1 we can see three basic patterns.

First, the density log bias seems to work better when it is not too strong. When we reduce `bias_scale` from 1.0 to 0.5, accuracy and stability often improve. For example, with CPE=ON and grid=16, *mAP* goes from 0.16482 to 0.17777, and *minADE*/*minFDE* improve from $(1.01492,\ 2.18288)$ to $(0.93433,\ 1.96221)$. At the same time, the maximum gradient drops from about 2.03×10^6 to about 5.55×10^3. So a smaller bias still lets density influence attention, but it does not overpower the content and it also reduces gradient spikes.

Second, using a finer grid usually helps. Moving from grid size 4 or 9 to 16 often improves *minADE* and *minFDE*, and in several cases also *mAP*. With CPE=ON and `bias_scale` $= 0.5$, grid size 16 gives the best displacement errors overall ($0.93433/1.96221$) and the highest *mAP* among the CPE=ON runs. This is in line with the idea that a finer grid lets the SDM place density more precisely in space, which should lead to cleaner top-k neighborhoods and more focused attention, even though we do not isolate that effect on its own.

CPE	bias_scale	grid	minADE ↓	minFDE ↓	mAP ↑	Grad. mean	Grad. max
ON	0.5	16	**0.93433**	**1.96221**	0.17777	197.96	5552.93*
ON	0.5	9	1.07653	2.22840	0.13962	339.18	22633.48
ON	0.5	4	1.01359	2.16053	0.17073	13911.93	2071695.25
ON	1.0	16	1.01492	2.18288	0.16482	5876.63	2026656.50
ON	1.0	9	1.03631	2.18702	0.15835	601.13	90397.03
ON	1.0	4	1.09102	2.24696	0.12747	8732.33	976845.69
OFF	0.5	16	0.97873	2.05912	0.18295	258.47	7740.69
OFF	0.5	9	1.10363	2.21213	0.14623	4021.03	446699.72
OFF	0.5	4	1.05832	2.22646	**0.18470**	4430.73	416781.97
OFF	1.0	16	1.06346	2.22644	0.16005	336.02	12977.04
OFF	1.0	9	0.95405	2.00842	0.16910	8665.51	1562453.50
OFF	1.0	4	1.14825	2.45938	0.14789	39392.66	1455208.13

Table 6.1: Validation results for SDT under different configurations of CPE, density bias scale, and grid resolution. Bold indicates the best value per column across all runs.
* marks the configuration used for the main model (CPE=ON, bias_scale=0.5, grid=16).

Third, CPE seems to help both accuracy and stability even when the differences in *mAP* are small. The best *mAP* in the table is with CPE=OFF, $\texttt{bias\_scale} = 0.5$, grid 4 (0.18470), but this run has worse displacement errors (1.05832/2.22646) and much higher gradients (4.17×10^5). In contrast, CPE=ON, $\texttt{bias\_scale} = 0.5$, grid 16 has the lowest *minADE*/*minFDE* and much lower gradient maxima, while *mAP* is only slightly lower (0.17777). Given this trade off, we use this configuration in the remaining experiments.

In summary, we choose CPE=ON, $\texttt{bias\_scale} = 0.5$, and grid size 16 as the main setting (first row in Table 6.1). Within the configurations we tested, this setup gives the lowest displacement errors (0.93433 m / 1.96221 m), a reasonable *mAP* of 0.17777, and a relatively mild gradient profile compared to other strong settings.

GMM diagnostics. Besides the usual validation metrics, we also looked at simple statistics of the Gaussian mixtures predicted by the SDM to understand their behavior (Table 6.2). We track three quantities: utilization above a density threshold (0.05), saturation variance, and entropy of the mixture weights.

Utilization shows how much probability mass lies in regions where the density is above the chosen threshold. Runs with CPE ON tend to have higher utilization for the large grid (for example 0.646 for grid 16 with bias 0.5), which matches the idea that motion features help the SDM focus on useful parts of the scene.

Variance shows how wide or narrow the Gaussian components are. Very small values (for example 0.078 with CPE ON, bias 1.0, grid 16) mean very sharp components that can hurt coverage, while very large values (for example 0.355 with CPE ON, bias 1.0, grid 4) mean that components spread too much. The main configuration (CPE ON, bias 0.5, grid 16) sits in between, at 0.127.

Entropy measures how many components are effectively active. High entropy, around

CPE	Bias	Grid	Utilization ↑	Variance ↓	Entropy$_{50}$ ↑	Entropy$_{mean}$ ↑
ON	0.5	16	0.646	0.127	2.361	2.111
ON	0.5	9	0.635	0.245	1.472	1.348
ON	0.5	4	0.645	0.230	0.772	0.726
ON	1.0	16	0.580	0.078	2.088	1.934
ON	1.0	9	0.904	0.008	2.117	2.073
ON	1.0	4	0.624	0.355	0.745	0.697
OFF	0.5	16	0.549	0.255	1.838	1.723
OFF	0.5	9	0.821	0.042	1.983	1.942
OFF	0.5	4	0.857	0.090	1.181	1.100
OFF	1.0	16	0.542	0.208	1.831	1.720
OFF	1.0	9	0.703	0.129	1.651	1.597
OFF	1.0	4	0.664	0.081	0.817	0.764

Table 6.2: GMM statistics for different settings of CPE, bias scale, and grid size. Higher utilization and entropy values indicate that more mixture components are used and remain active, while lower variance corresponds to more stable and less diffuse component spread.

2.3, means that several components are used in practice, while low entropy, around 0.7, means that only one or two components dominate.

The 3×3 grid (grid size 9) gives very high utilization in some cases (0.904 with CPE ON, bias 1.0), but its variance almost collapses (0.008), which indicates nearly degenerate Gaussians that put mass into very narrow spikes. This is not ideal for coverage. The 4×4 grid (grid size 16) trades a small drop in utilization for more reasonable variance and higher entropy, which makes the density field easier to interpret and less fragile.

Overall, these statistics support the choice of CPE=ON, bias 0.5, grid 16 as a balanced setting for the rest of the experiments.

6.4 Model Diagnostics

GMM behavior and insights. The Spatial Density Module (SDM) forms a central part of SDT's relational reasoning by predicting Gaussian mixtures over the spatial domain. To better understand its internal behavior over time, we analyzed the mixture diagnostics across epochs using the same three quantities: (i) utilization above a density threshold (0.05), (ii) saturation variance of the Gaussian components, and (iii) entropy of the mixture weights.

Utilization gradually increased during training, which suggests that the SDM learned to concentrate probability mass on specific interaction zones rather than dispersing it

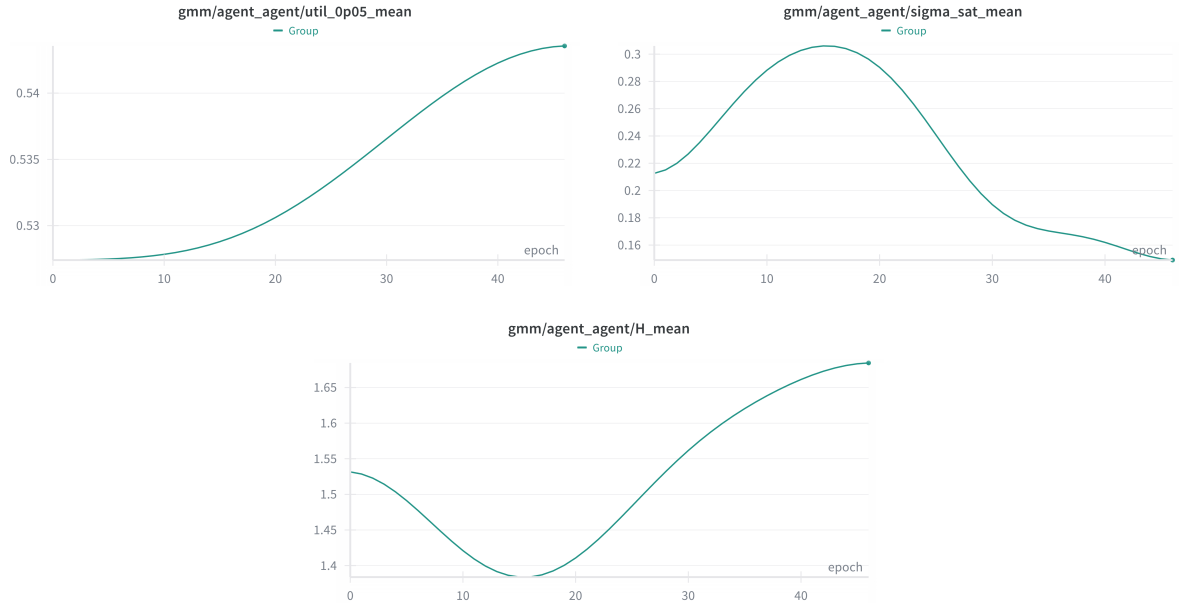


Figure 6.1: Evolution of GMM diagnostics over epochs for agent-agent interactions. The mixture maintains moderate variance and stable entropy.

uniformly. Variance values stabilized around 0.12 to 0.15, so the Gaussian components neither collapsed into extremely sharp peaks nor spread excessively. Entropy hovered around 2.1 to 2.3, indicating that several mixture components remained active.

These trends are compatible with the intended role of SDM as a density prior that regularizes attention, acting as a soft, spatially aware filter that suppresses attention toward very low density regions while keeping multiple hypotheses active. However, we do not provide a direct comparison with random or k NN filtering in this thesis, so a more precise quantification of the benefit of density-aware filtering over purely geometric heuristics is left for future work.

6.5 Benchmark Results

Table 6.3 summarizes the quantitative results of the **Spatial Density Transformer (SDT)** and **HPTR** on the Waymo Open Motion Dataset validation split. For SDT, we train the model for 50 epochs and report the checkpoint from epoch 42, which was selected based on validation loss. At this checkpoint, SDT achieves a mean average precision (**mAP**) of 0.2790, a minimum average displacement error (**Min ADE**) of 0.7189 m, and a minimum final displacement error (**Min FDE**) of 1.4903 m. The corresponding **Miss Rate** and **Overlap Rate** are 0.2106 and 0.1437, respectively. For reference, HPTR achieves an **mAP** of 0.4150, **Min ADE** of 0.5378 m, **Min FDE** of 1.0923 m, **Miss Rate** of 0.1326, and **Overlap Rate** of 0.1357 on the same validation

split.

Model	mAP ↑	Min ADE ↓	Min FDE ↓	Miss Rate ↓	Overlap Rate ↓
SDT(Ours)	0.2790	0.7189	1.4903	0.2106	0.1437
HPTR (reference)	0.4150	0.5378	1.0923	0.1326	0.1357

Table 6.3: Validation results of SDT and HPTR on the WOMD validation split. Arrows indicate preferred directions for each metric.

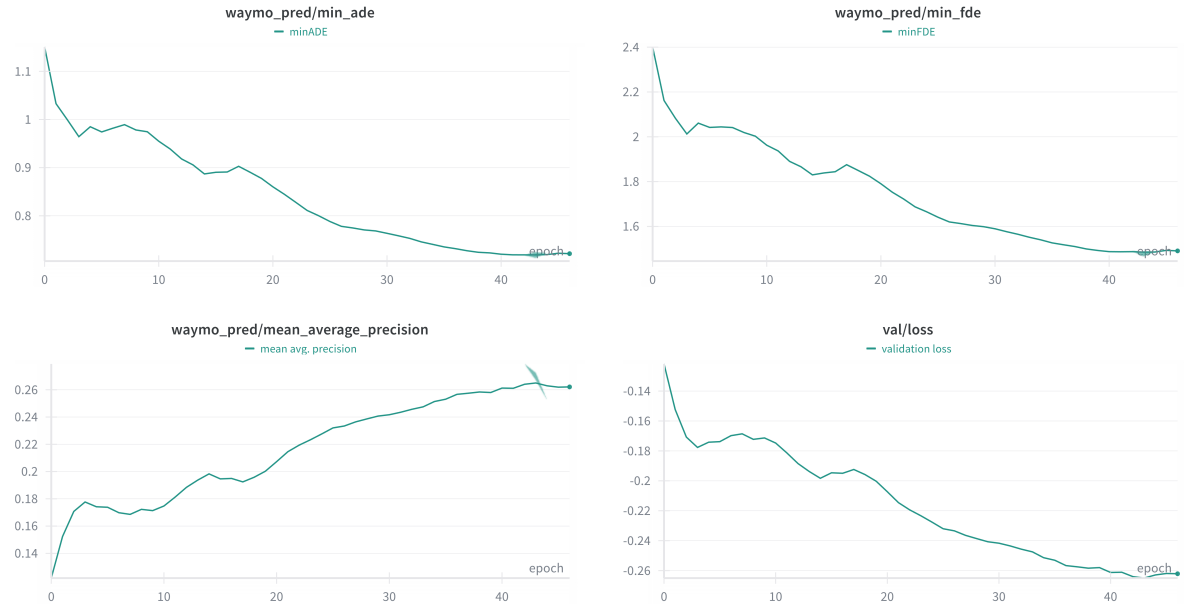


Figure 6.2: Evolution of key validation metrics during training. The curves flatten around epoch 40, with mAP changing more slowly afterwards.

All trajectory related errors (position, velocity, speed) decrease steadily over epochs, which suggests that the model fits the kinematic variables in a stable way. The yaw error (orientation) behaves less smoothly and increases slightly toward the end of training. Since the loss on yaw is based on a von Mises distribution rather than direct regression, this can happen when positional accuracy improves faster than angular calibration.

The overall training loss stabilizes around 2.97, and the validation loss around 0.26 in absolute value, which indicates that the optimization process has mostly converged within the given budget. GPU memory utilization remains close to 55 percent of the available 12 GB during training, showing that the density-based attention mechanism keeps a predictable memory footprint.

A more extensive benchmark that includes HPTR and other baselines trained under identical conditions is left as future work.

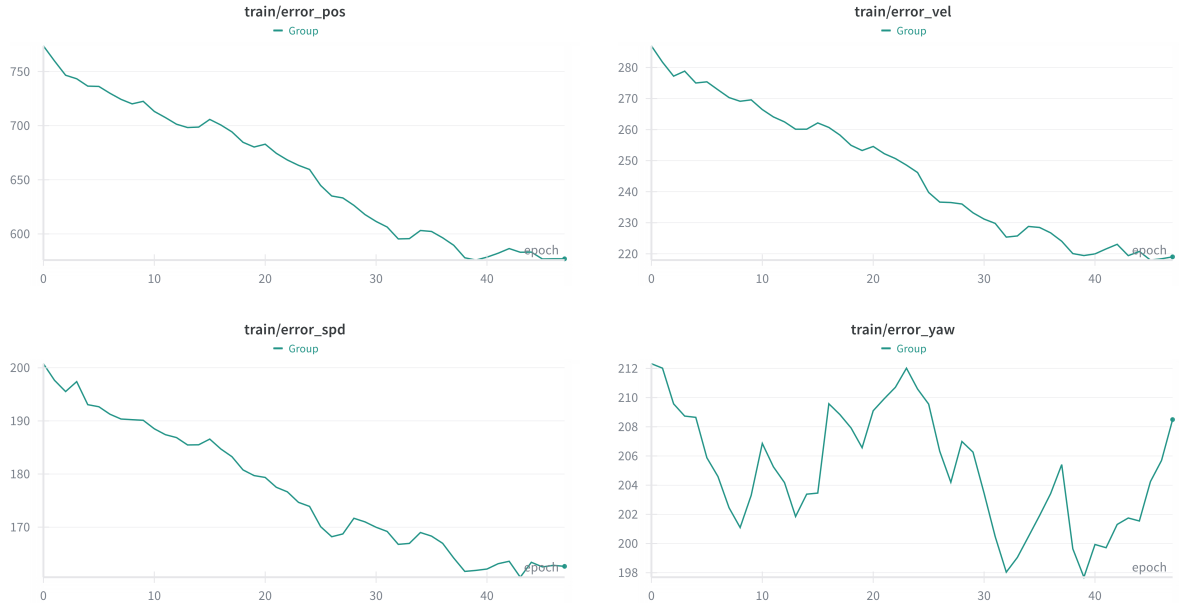


Figure 6.3: Evolution of component errors for position, velocity, speed, and yaw. All components except yaw show a clear downward trend.

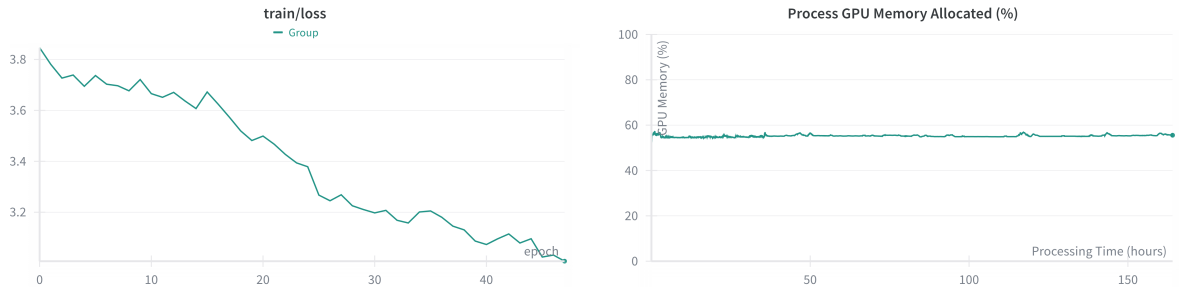


Figure 6.4: Training loss and GPU memory utilization over epochs. The memory footprint stays stable with no sign of accumulation.

6.6 Qualitative Analysis

Qualitative examples. To visually assess the behavior of the model, Figure 6.5 shows two representative prediction scenes from the WOMD validation set. Each scene depicts the ego centric map and the predicted future trajectories for a selected target vehicle. The ground truth trajectory is rendered in orange, while the predicted modes are shown in cyan. The most confident prediction is emphasized with increased line thickness, reduced transparency, and a larger terminal marker.

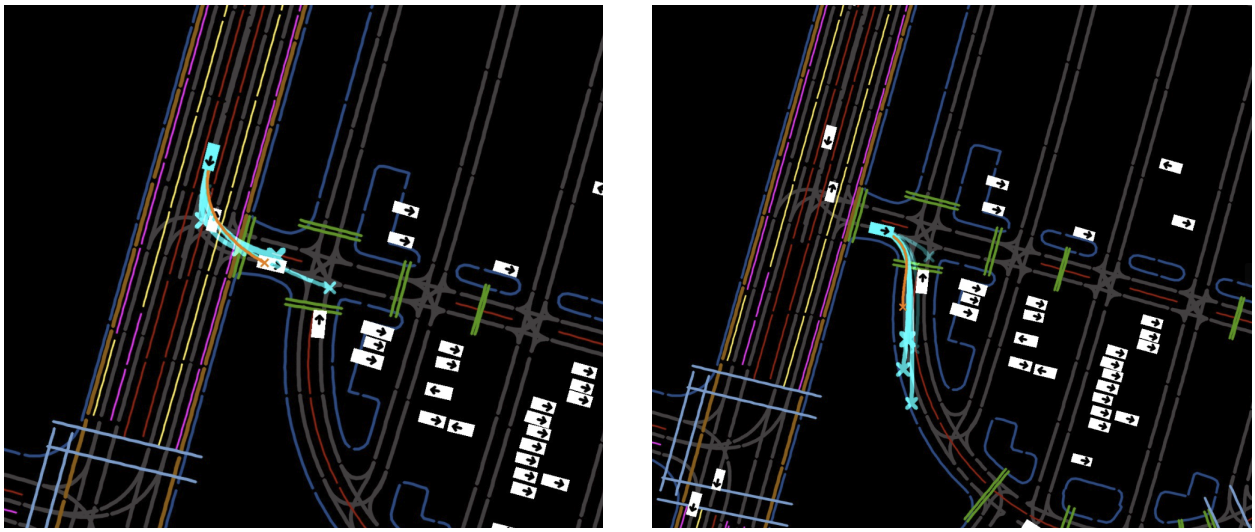


Figure 6.5: Qualitative results on WOMD validation scenes. The ground truth is shown in orange, predicted trajectories in cyan. Thicker and less transparent lines indicate higher confidence. In these examples the top mode matches the ground truth maneuver.

In the first example, the target vehicle approaches an intersection where it may either continue straight or turn right toward a side street or parking area. SDT produces multiple hypotheses and, in this case, assigns highest confidence to the turning maneuver that matches the recorded ground truth. In the second example, several hypotheses correspond to different turning angles at a junction, capturing uncertainty in driver intent.

In both scenes, the predicted modes stay roughly aligned with lane geometry and local map structure, which is consistent with the intended effect of density-aware attention focusing on nearby lanes and relevant context.

7. CONCLUSION

This thesis presented the **Spatial Density Transformer (SDT)**, a motion prediction model for autonomous driving that builds directly on the HPTR architecture. The main idea is to replace fixed, heuristic neighbor selection with a learned *Spatial Density Module* (SDM) that predicts Gaussian mixtures over the spatial domain and uses them to guide attention. The SDM provides both a soft density prior and a top-k selection mechanism, so that attention focuses on regions where interactions are more likely instead of treating all pairs of tokens equally.

Experiments on the Waymo Open Motion Dataset showed that SDT can reach reasonable accuracy under a limited training budget. The best checkpoint on the validation split achieved minADE of 0.72 m, minFDE of 1.49 m and mAP of 0.28, with moderate miss and overlap rates. Within this setup, training and validation curves were stable, and memory usage stayed roughly constant during training, which suggests that the density based attention mechanism can be implemented without introducing extra instability in practice.

The ablation study and the GMM diagnostics gave some insight into how the SDM behaves. Configurations with Component Pairwise Encoding (CPE), a moderate density bias scale and a finer spatial grid tended to give better displacement errors, smoother gradients and more interpretable mixtures. The mixture statistics also indicated that, in these settings, density tends to concentrate on areas that contain agents rather than on empty regions, and that several mixture components remain active instead of collapsing into a single peak.

At the same time, there are clear limitations. The model is evaluated on a single dataset and backbone, without a full head to head comparison against HPTR or other baselines trained under exactly the same conditions. The attention mechanism is restricted to sparse neighborhoods, which may reduce the ability to capture very long range interactions in crowded or highly unstructured scenes. Orientation errors remain harder to optimize than positional errors, partly due to the circular nature of yaw and the choice of von Mises loss. Finally, the analysis of the SDM focuses on simple mixture statistics and a small set of hyperparameters, and does not exhaust all possible behaviors of the module.

There are several directions for future work. One is to run a more systematic comparison between density guided filtering and simpler strategies such as fixed k nearest neighbors or random pruning, using matched training schedules for SDT and HPTR. Another is to study variants where the mixtures are constrained to behave closer to k nearest neighbor neighborhoods, or where CPE and SDM are disabled, in order to better understand how much each component contributes. Extending SDT to joint prediction, to other datasets, or to settings where it is combined with upstream perception or downstream planning modules would also be interesting. Overall, the thesis suggests that adding a learned, density aware attention filter on top of an existing lightweight transformer such as HPTR is a promising direction, and that it can be

explored further with stronger baselines and more extensive experiments in future work.

ABBREVIATIONS - ACRONYMS

AV	Autonomous Vehicle
TP	Trajectory Prediction
WOMD	Waymo Open Motion Dataset
HD	High-Definition
VAE	Variational Autoencoder
MLP	Multi-Layer Perceptron
GMM	Gaussian Mixture Model
SDT	Spatial Density Transformer
SDA	Spatial Density Attention
SDM	Spatial Density Module
RPE	Relative Pose Encoding
RME	Relative Motion Encoding
CPE	Component Pairwise Encoding
STE	Straight-Through Estimator
mAP	mean Average Precision
FDE	Final Displacement Error
ADE	Average Displacement Error

REFERENCES

[1] R. Rajamani, *Vehicle Dynamics and Control*, 2nd ed. Springer, 2011.

[2] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[3] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid, "VectorNet: Encoding HD Maps and Agent Dynamics from Vectorized Representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 525–11 533.

[4] M. Liang, B. Yang, R. Hu, Y. Chen, R. Liao, S. Feng, and R. Urtasun, "Learning Lane Graph Representations for Motion Forecasting," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2020, pp. 541–556.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention Is All You Need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS)*, 2017, pp. 5998–6008.

[6] Z. Zhang, A. Liniger, C. Sakaridis, F. Yu, and L. V. Gool, "Real-Time Motion Prediction via Heterogeneous Polyline Transformer with Relative Pose Encoding," *arXiv preprint arXiv:2310.12970*, 2023.

[7] Y. Huang, J. Du, Z. Yang, Z. Zhou, L. Zhang, and H. Chen, "A Survey on Trajectory-Prediction Methods for Autonomous Driving," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 3, pp. 652–674, 2022.

[8] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. S. Torr, and M. Chandraker, "DESIRE: Distant Future Prediction in Dynamic Scenes with Interacting Agents," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2165–2174.

[9] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 2255–2264.

[10] S. Casas, C. Gulino, S. Suo, K. Luo, R. Liao, and R. Urtasun, "Implicit Latent Variable Model for Scene-Consistent Motion Forecasting," in *European Conference on Computer Vision (ECCV)*, 2020.

[11] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov, "MultiPath: Multiple Probabilistic Anchor Trajectory Hypotheses for Behavior Prediction," in *Conference on Robot Learning (CoRL)*, 2019, pp. 86–99.

[12] B. Ivanovic and M. Pavone, "The Trajectron: Probabilistic Multi-Agent Trajectory Modeling with Dynamic Spatiotemporal Graphs," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 2375–2384.

[13] T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, and F. Moutarde, "THOMAS: Trajectory Heatmap Output with Learned Multi-Agent Sampling," in *International Conference on Learning Representations (ICLR)*, 2022.

[14] R. Girgis, F. Golemo, F. Codevilla, M. Weiss, J. A. D'Souza, S. E. Kahou, F. Heide, and C. Pal, "Latent Variable Sequential Set Transformers for Joint Multi-Agent Motion Prediction," in *International Conference on Learning Representations (ICLR)*, 2022.

[15] X. Mo, Y. Xing, and C. Lv, "Heterogeneous Edge-Enhanced Graph Attention Network for Multi-Agent Trajectory Prediction," *arXiv preprint arXiv:2106.07161*, 2021.

[16] Q. Sun, X. Huang, J. Gu, B. C. Williams, and H. Zhao, "M2I: From Factored Marginal Trajectory Prediction to Interactive Prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 6543–6552.

[17] D. Rempe, J. Philion, L. J. Guibas, S. Fidler, and O. Litany, "Generating Useful Accident-Prone Driving Scenarios via a Learned Traffic Prior," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 17 305–17 315.

[18] S. Suo, S. Regalado, S. Casas, and R. Urtasun, "TrafficSim: Learning to Simulate Realistic Multi-Agent Behaviors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 10 400–10 409.

[19] Z. Zhang, A. Liniger, D. Dai, F. Yu, and L. V. Gool, "TrafficBots: Towards World Models for Autonomous Driving Simulation and Motion Prediction," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[20] H. Caesar, J. Kabzan, K. S. Tan, W. K. Fong, E. Wolff, A. Lang, L. Fletcher, O. Beijbom, and S. Omari, "nuPlan: A Closed-Loop ML-Based Planning Benchmark for Autonomous Vehicles," *arXiv preprint arXiv:2106.11810*, 2021.

[21] S. Casas, A. Sadat, and R. Urtasun, "MP3: A Unified Model to Map, Perceive, Predict and Plan," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 14 403–14 412.

[22] J. L. Vazquez Espinoza, A. Liniger, W. Schwarting, D. Rus, and L. Van Gool, "Deep Interactive Motion Prediction and Planning: Playing Games with Motion Prediction Models," in *Proceedings of the Learning for Dynamics and Control Conference (L4DC)*, 2022, pp. 992–1004.

[23] A. Hu, Z. Murez, N. Mohan, S. Dudas, J. Hawke, V. Badrinarayanan, R. Cipolla, and A. Kendall, "FIERY: Future Instance Prediction in Bird's-Eye View from Surround Monocular Cameras," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 15 203–15 212.

[24] A. Kamenev, L. Wang, O. B. Bohan, I. Kulkarni, B. Kartal, A. Molchanov, S. Birchfield, D. Nistér, and N. Smolyanskiy, "PredictionNet: Real-Time Joint Probabilistic Traffic Prediction for Planning, Control, and Simulation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 8938–8944.

[25] A. Hu, G. Corrado, N. Griffiths, Z. Murez, C. Gurau, H. Yeo, A. Kendall, R. Cipolla, and J. Shotton, "Model-Based Imitation Learning for Urban Driving," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[26] Y. Hu, J. Yang, L. Chen, K. Li, C. Sima, X. Zhu, S. Chai, S. Du, T. Lin, W. Wang, L. Lu, X. Jia, Q. Liu, J. Dai, Y. Qiao, and H. Li, "Planning-Oriented Autonomous Driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 17 853–17 862.

[27] S. Ettinger, S. Cheng, B. Caine, C. Liu, H. Zhao, S. Pradhan, Y. Chai, B. Sapp, C. R. Qi, Y. Zhou, Z. Yang, A. Chouard, P. Sun, J. Ngiam, V. Vasudevan, A. McCauley, J. Shlens, and D. Anguelov, "Large Scale Interactive Motion Forecasting for Autonomous Driving: The Waymo Open Motion Dataset," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 9710–9719.

[28] Q. Ge, S. E. Li, Q. Sun, and S. Zheng, "Numerically Stable Dynamic Bicycle Model for Discrete-Time Control," *arXiv preprint arXiv:2011.09612*, 2020.

[29] K. P. Murphy, "Dynamic Bayesian Networks: Representation, Inference and Learning," Ph.D. dissertation, University of California, Berkeley, 2002.

[30] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006. [Online]. Available: http://www.gaussianprocess.org/gpml

[31] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[32] L. Liu, Z. Gong, and B. Sun, "Trajectory Outlier Detection: A KNN Approach," in *International Conference on Artificial Intelligence (ICAI)*, 2011, p. 135–141.

[33] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[34] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[35] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The Graph Neural Network Model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[36] Z. Zhou, L. Ye, J. Wang, K. Wu, and K. Lu, "HiVT: Hierarchical Vector Transformer for Multi-Agent Motion Prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 8823–8833.

[37] A. Cui, S. Casas, K. Wong, S. Suo, and R. Urtasun, "GoRela: Go Relative for Viewpoint-Invariant Motion Forecasting," *arXiv preprint arXiv:2211.02545*, 2022.

[38] J. L. Elman, "Finding Structure in Time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[39] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[40] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the Properties of Neural Machine Translation: Encoder–Decoder Approaches," in *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, 2014, pp. 103–111.

[41] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human Trajectory Prediction in Crowded Spaces," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 961–971.

[42] J. Ngiam, B. Caine, V. Vasudevan, Z. Zhang, H.-T. L. Chiang, J. Ling, R. Roelofs, A. Bewley, C. Liu, A. Venugopal, D. Weiss, B. Sapp, Z. Chen, and J. Shlens, "Scene Transformer: A unified architecture for predicting multiple agent trajectories," 2022. [Online]. Available: https://arxiv.org/abs/2106.08417

[43] N. Nayakanti, R. Al-Rfou, A. Zhou, K. Goel, K. S. Refaat, and B. Sapp, "Wayformer: Motion Forecasting via Simple & Efficient Attention Networks," 2022. [Online]. Available: https://arxiv.org/abs/2207.05844

[44] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2014, pp. 2672–2680.

[45] J. Gu, C. Sun, and H. Zhao, "DenseTNT: End-to-End Trajectory Prediction from Dense Goal Sets," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[46] S. Shi, L. Jiang, D. Dai, and B. Schiele, "MTR++: Multi-Agent Motion Prediction with Symmetric Scene Modeling and Guided Intention Querying," 2024. [Online]. Available: https://arxiv.org/abs/2306.17770

[47] W. Kool, H. van Hoof, and M. Welling, "Stochastic Beams and Where To Find Them: The Gumbel-Top-k Trick for Sampling Sequences Without Replacement," in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, vol. 97, 2019, pp. 3499–3508.