CASTNet: A Context-aware, Spatio-temporal Dynamic Motion Prediction Ensemble for Autonomous Driving

TRIER MORTLOCK, The University of California, Irvine, USA
ARNAV VAIBHAV MALAWADE, The University of California, Irvine, USA
KOHEI TSUJIO, The University of California, Irvine, USA
MOHAMMAD ABDULLAH AL FARUQUE, The University of California, Irvine, USA

Autonomous vehicles are cyber-physical systems that combine embedded computing and deep learning with physical systems to perceive the world, predict future states, and safely control the vehicle through changing environments. The ability of an autonomous vehicle to accurately predict the motion of other road users across a wide range of diverse scenarios is critical for both motion planning and safety. However, existing motion prediction methods do not explicitly model contextual information about the environment, which can cause significant variations in performance across diverse driving scenarios. To address this limitation, we propose **CASTNet**: a dynamic, context-aware approach for motion prediction that (i) identifies the current driving context using a spatio-temporal model, (ii) adapts an ensemble of motion prediction models to fit the current context, and (iii) applies novel trajectory fusion methods to combine predictions output by the ensemble. This approach enables CASTNet to improve robustness by minimizing motion prediction error across diverse driving scenarios. CASTNet is highly modular and can be used with various existing image processing backbones and motion predictors. We demonstrate how CASTNet can improve both CNN-based and graph-learning-based motion prediction approaches and conduct ablation studies on the performance, latency, and model size for various ensemble architecture choices. In addition, we propose and evaluate several attention-based spatio-temporal models for context identification and ensemble selection. We also propose a modular trajectory fusion algorithm that effectively filters, clusters, and fuses the predicted trajectories output by the ensemble. On the nuScenes dataset, our approach demonstrates more robust and consistent performance across diverse, real-world driving contexts than state-of-the-art techniques.

CCS Concepts: • **Computer systems organization → Embedded and cyber-physical systems**; • **Computing methodologies → Machine learning**; *Ensemble methods*; **Motion path planning**; *Computer vision*; *Intelligent agents*; **Multi-agent planning;**

Additional Key Words and Phrases: Autonomous vehicles, information fusion, dynamic machine learning architectures, context-aware models, ensemble learning, motion planning

ACM Reference Format:
Trier Mortlock, Arnav Vaibhav Malawade, Kohei Tsujio, and Mohammad Abdullah Al Faruque. 2024. CASTNet: A Context-aware, Spatio-temporal Dynamic Motion Prediction Ensemble for Autonomous Driving. *ACM Trans. Cyber-Phys. Syst.* 8, 2, Article 23 (May 2024), 20 pages. https://doi.org/10.1145/3648622

T. Mortlock and A. V. Malawade contributed equally to this research.

This work was partially supported by the National Science Foundation (NSF) under awards CMMI-1739503 and CCF-2140154.

Authors' address: T. Mortlock, A. V. Malawade, K. Tsujio, and M. A. Al Faruque, The University of California, Irvine, 5440 Engineering Hall, Irvine, CA, 92697; e-mails: tmortloc@uci.edu, malawada@uci.edu, ktsujio@uci.edu, alfaruqu@uci.edu.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM 2378-962X/2024/05-ART23
https://doi.org/10.1145/3648622

1 INTRODUCTION

Autonomous vehicles (AVs) are **cyber-physical systems (CPS)** that rely on a tightly coupled feedback loop of perception, prediction, planning, and control [14, 43]. Like many CPSs, AVs are real-time, energy-constrained systems that require timely decisions that can be influenced by surrounding environmental factors [29]. Figure 1 depicts a high-level illustration of a typical modular autonomous driving pipeline [43]. Ultimately, an AV implements a closed-loop control system that receives sensor and other data from the environment as inputs and determines actuation commands to control the vehicle as outputs. The main stages of operation include: (a) *sensing*—processing of on-board sensor data (e.g., cameras, radars, inertial measurement units) and data about the environment (e.g., maps, directions); (b) *positioning*—localizing the vehicle in its local frame as well as navigating the vehicle with respect to a global frame; (c) *perception*—usually conducted in parallel with positioning tasks, responsible for the detection and tracking of surrounding agents or objects; (d) *prediction*—predicting the motion of other surrounding agents (e.g., vehicles, pedestrians, bicycles); (e) *planning*—determining the desired AV path given the surround objects and their trajectories; and (f) *control*—updating the AV's estimate of its state and outputting the desired actuation commands given a control algorithm.

As Figure 1 highlights, this article focuses on the *motion prediction* stage, which directly influences the AV's path planning and control tasks. Predicting agent (e.g., motorists, cyclists, pedestrians) motion in an environment is a critical spatio-temporal modeling task that requires blending various domains, including sensor fusion, dynamics, mapping, and data-driven modeling. Motion prediction not only influences the planning and control of AVs [15], but also has significant safety benefits: The ability to predict future agent positions accurately can help ensure safe route planning and crash avoidance [28, 30].

Recent works have shown that explicitly modeling contextual information can aid the performance of AVs by informing how agents are likely to behave given external factors [6, 18, 25, 26, 34]. More broadly, authors in Reference [2] argue that extracting contextual information from sensory data in a CPS is critical for enabling increased autonomy levels. Reference [32] highlights the importance of context identification in analyzing the feature space of CPSs regarding out-of-distribution analysis. Furthermore, Reference [27] demonstrates that context-specific modeling can improve AV performance in dynamic environments. Contextual modeling can be key in developing *adaptive* architectures for various CPS applications [12, 16, 37, 42]. Existing methods that lack adaptation suffer in performance during high uncertainty levels in challenging environments [16] and along new driving scenarios [42]. Adaptive deep-learning methods for CPSs have been proposed to leverage context identification at runtime [12, 37]. Inspired by these works, we define context as *information derived from sensory data about the current state of the system and the surrounding environment that influences the system's goals or tasks*. In the scope of this article, the *system* is the autonomous vehicle, and the *task* under study is motion prediction.

1.1 Motivation

In this subsection, we provide insights and qualitative results on the limitations existing motion predictors face in challenging driving environments and the benefits of our proposed approach. The main categories of current motion predictors are physics-based approaches and data-driven machine learning approaches. Physics-based motion prediction models struggle to predict beyond a few seconds in the future, as they often fail to anticipate driver actions or influences from external factors (e.g., other vehicles, road constraints) [23]. Machine learning approaches for AV motion prediction have proven more effective [7, 31, 35, 36]. However, existing motion prediction methods do not explicitly account for driving context in their methodologies and can fail in some contexts,

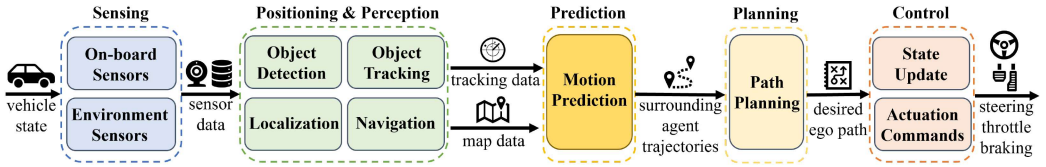

Fig. 1. A typical modular autonomous driving pipeline. Prediction takes in tracking and map data along with sensor data to produce trajectories of the detected road actors. As shown, motion prediction is critical for both the downstream tasks of path planning and control.

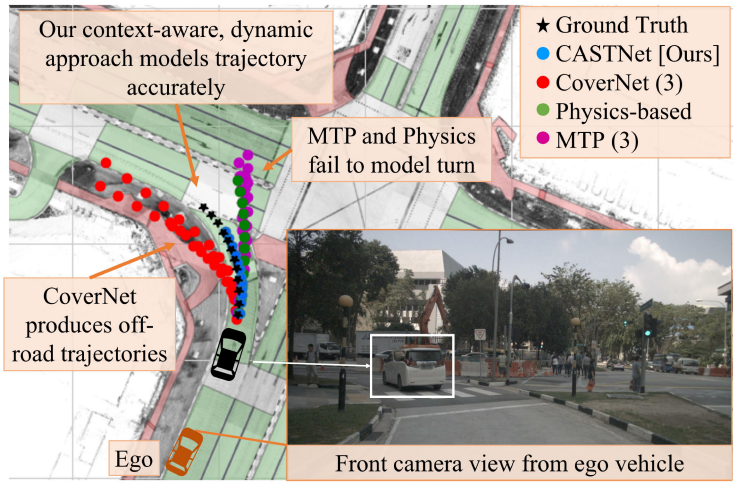

Fig. 2. An example of the limitations of existing motion predictors in a complex driving scenario from nuScenes [4]. In this scene, the goal is to predict the future trajectory of the white van (ground truth shown in black on the map). The results from existing motion predictors are shown in their respective colors and their points of failure are highlighted in the text boxes. Our approach, CASTNet—shown in blue—achieves the best performance in terms of minimum distance from ground truth.

as demonstrated in Figure 2. In this example, the AV (ego vehicle) is attempting to predict the motion of the white van located in front of it for the next 12 timesteps (6 seconds at 2 Hz). The ground truth of the vehicle is shown in black stars, and the predicted trajectories of different motion predictors are shown in their respective colors (with CoverNet [31] and MTP [7], each producing three possible trajectories). As shown in the example, this challenging setting involves construction, numerous pedestrians, and a narrow turn lane in an urban driving setting. Since the two machine learning–based methods (CoverNet and MTP) and the physics-based model do not explicitly model context, these models cannot account for these factors and fail to predict the white van's true trajectory accurately; CoverNet predicts an unlikely off-road trajectory, while MTP and the physics-based model seem to ignore the turn lane and predict that the van will drive straight into oncoming traffic. Overall, vehicle motion prediction presents the following key research challenges:

(1) Predicting motion paths over long-term horizons.
(2) Accounting for spatio-temporal dependencies in the environment.
(3) Utilizing contextual information from sensory data to inform motion predictions.
(4) Maintaining robust prediction performance in challenging scenarios.

To address these research challenges, we propose **CASTNet**—a context-aware dynamic ensemble approach for AV motion prediction. CASTNet, shown in blue in Figure 2, can predict the closest trajectory to the ground truth by modeling the context of the surrounding environment, adapting the ensemble architecture accordingly, and fusing submodel outputs to produce a more accurate, context-aware trajectory.

1.2 Novel Contributions

CASTNet employs a modular architecture that dynamically adapts to enable robust motion prediction performance across diverse driving scenarios. By using a novel attention-based spatio-temporal context identification module to identify the set of driving contexts present in each input, CASTNet can select the most effective motion predictors to execute for each input, significantly improving performance across scenarios compared to state-of-the-art methods, which use static architectures and do not explicitly model context. CASTNet is a general approach that can be used with various ensemble backbones and context-identification models to convert traditional static architecture models to context-aware motion predictors. In this work, we present the following novel contributions:

(1) We propose CASTNet, a context-aware dynamic approach for AV motion prediction, that outperforms state-of-the-art methods on diverse, real-world driving data.
(2) We propose several novel techniques for spatio-temporal context identification that enhance situational understanding during AV motion prediction.
(3) We illustrate how CASTNet improves robustness across driving contexts while providing the first analysis of how an adaptive AV motion prediction framework can leverage context to improve performance.
(4) We demonstrate CASTNet's modularity by performing ablation studies with different architectural configurations and motion prediction models.
(5) We propose a trajectory fusion algorithm to fuse the outputs of an ensemble of motion prediction models and evaluate the performance of several fusion methods.

1.3 Article Organization

The remainder of the article is organized as follows: Section 2 discusses related works including background on AV motion prediction, context-aware methods in autonomous systems, and multi-modal trajectory fusion approaches; Section 3 introduces the motion prediction problem formulation along with detailed descriptions of our proposed methodology; Section 4 presents our experimental setup, results, and discusses key findings and future research directions; and finally, Section 5 concludes the article.

2 RELATED WORK

2.1 Motion Prediction

Several recent works have proposed methods for AV motion prediction. **Multimodal Trajectory Prediction (MTP)** [7] uses a **convolutional neural network (CNN)** as a backbone feature extractor to model raster map data. These features are then combined with agent state information (i.e., position, speed, heading, acceleration, and heading change rate) and passed through a **multilayer perceptron (MLP)** to predict multiple possible trajectories for each agent. CoverNet [31] uses a similar approach to MTP, with the addition of a trajectory set generator that maps model outputs to either a fixed or dynamic set of feasible trajectories. The problem is thus translated to classification over a trajectory set. Trajectron++ [35] represents the agents in the scene and their relations to one another as a spatio-temporal graph, incorporating agent dynamics, map data, and

scene information. A recurrent model for motion prediction along with Gaussian mixtures then models this representation. Other approaches that use mixture of experts methods have shown benefits in AV motion prediction, for example, Reference [20], which leverages a mixture of experts approach to select predictors based on uncertainties. However, this analysis focuses only on ego vehicle motion prediction.

Other recent motion prediction works include PLOP [3] (uses conditional imitation learning), PRECOG [33] (conditions forecasts on agent goals), and GOHOME [13] (utilizes a global heatmap probability distribution). Reference [22] combines learning-, physics-, and maneuver-based methods and weighs their outputs as a function of their respective uncertainties in each driving scene, enabling better predictions in unseen environments. These methods successfully incorporate environmental information for the prediction task; however, they employ static architectures and do not explicitly model context, so these methods can fail when faced with complex real-world driving scenarios.

2.2 Context-aware Methods in Autonomous Cyber-physical Systems

Methods exploring the use of context are essential for establishing increased levels of autonomy in cyber-physical systems. Authors in Reference [2] propose a specific class of self-aware cyber-physical systems that focus on extracting contextual information from sensory data. They argue that leveraging distinctions in sensing can lead to more contextually meaningful modeling and, ultimately, more autonomy in the design of CPS. The analysis of CPS performance on out-of-distribution data is another challenge that context-aware methods can improve. Authors in Reference [32] utilize context identification at runtime to analyze out-of-distribution features to enhance the safety of CPS. Furthermore, context-aware methods are commonly used to develop adaptive architectures for CPS [12, 16, 37, 42]. Authors in Reference [12] apply meta-adaptation strategies for CPS to improve performance across changing environments. Likewise, in Reference [37], adaptive deep learning is applied to improve CPS localization.

Context-aware methods have also shown benefits for various AV-specific problems. Reference [21] uses contextual features such as curb distance and traffic lights to improve pedestrian trajectory prediction. MultiPath++ [39], an extension of MultiPath [5], shows improved motion forecasting results by integrating contextual information. Context-based methods using transformers [24] and graph neural networks [6] have also proven effective for AV trajectory prediction; however, they employ static architectures that can fail to adapt to challenging scenarios. Context-aware dynamic sensor fusion architectures have been shown to improve object detection robustness and energy efficiency in AV perception systems [25, 26]. Furthermore, authors in Reference [27] apply context-specific modeling to provide safe velocity regulation of AVs in dynamic, unseen environments.

Context also informs AV motion prediction frameworks by providing critical information about the current scene. Specifically, road type [9, 41] and density [8, 40] have been shown to influence motion predictors. In Reference [41], authors define a semantic model of a driving scene that includes road types to develop an improved trajectory prediction model. Reference [9] improves motion prediction by encoding map elements based on the scene, including driving paths, crosswalks, lane and road boundaries, intersections, driveways, and parking lots. Reference [40] classifies car-following behaviors using traffic density and road type as contextual information.

2.3 Multi-modal Trajectory Fusion

Trajectory prediction methods often output multiple possible trajectories—or *modes*—per agent. These outputs can then be ranked and selected using confidence scores or fused to produce newly refined trajectories. This fusion process, called multi-modal trajectory fusion, involves combining

multiple modes to enhance and refine predicted trajectories so the fused modes are more accurate than previous individual modes [1]. There are various methods to choose subsets of modes to fuse, such as selection algorithms inspired by non-maximum suppression in object detection [44] or optimization-based algorithms [13]. However, these methods have been focused on the domain of estimation and tracking and not motion prediction. One standard method of trajectory fusion applied to estimation and tracking is the **Kalman filter (*KF*)**. Kalman filters assume each mode is uncorrelated and include a prediction step incorporating a temporal motion model and an update step using available measurements. The fused mode is ultimately estimated by the filter by making predictions using the known dynamics and noise model and correcting these predictions via updates from the measurements (in our case, the different modes).

3 METHODOLOGY

3.1 Problem Formulation

Following the typical AV modular pipeline described in Figure 1, the first objective of the AV system is to perceive and model sensor data from the environment via the *positioning* and *perception* modules. These modules take sensor data D from the AV (e.g., camera, lidar) as input and (i) detect and track objects and (ii) localize objects, spatial landmarks, and the AV itself relative to stored high-definition map data. The object detection and tracking components produce a list of agents in the scene and corresponding tracking information (e.g., position, velocity, heading) denoted T. At the same time, the localization and navigation components use stored high-definition map data denoting the different lanes, directions of travel, and static obstacles in the physical environment to correlate object tracks with their relative map locations. Next, these outputs are processed by the *prediction* module to predict future object trajectories.

As in related works [7, 10, 31], we combine the tracking information T and map information provided by the *perception* module to construct a bird's-eye view of the scene at each timestep containing both current and past location information about each visible agent, resulting in a raster map M. Following Reference [4], we encode two seconds of historical tracking information T into M to predict future agent trajectories. We denote $s^a_{i:j}$ as the discrete-time trajectory of an agent a from frame i to j where $i < j$. Our objective is to predict the trajectory of each agent a in a scene by modeling the following function:

$$s^a_{i:j} = \phi(M_a, T_a, D), \tag{1}$$

for all a in M where M_a is M transformed to be in the reference frame of a, and T_a is the tracking information corresponding to a. We extend this core formulation to a dynamic model architecture as follows: Here, we assume ϕ is composed of a general purpose feature extraction model β that produces an initial set of features $\mathbf{F}$, as well as several independent motion prediction sub-models $\phi_1, \phi_2, \ldots, \phi_n$ that process $\mathbf{F}$.

$$\mathbf{F} = \beta(M_a) \tag{2}$$

We propose that ϕ contains a context-identification component π to identify the scene's current context ω and dynamically configure which sub-models are executed for each input. We denote this selected set of n submodels as ϕ^*.

$$\omega = \pi(D, T_a) \tag{3}$$

$$\phi^* = \phi[\omega] = \{\phi_1, \phi_2, \ldots \phi_n\} \tag{4}$$

Then, we process $\mathbf{F}$ and T_a with each submodel in ϕ^*, where each submodel outputs a set of output trajectories, denoted as $s_1, s_2, \ldots, s_n$.

$$s_1, s_2, \ldots s_n = \phi^*(\mathbf{F}, T_a) \tag{5}$$

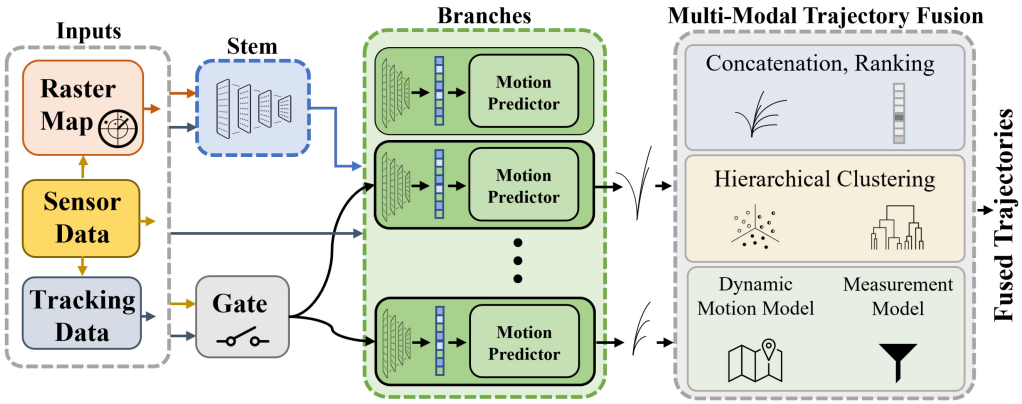

Fig. 3. CASTNet Architecture. Our approach processes sensor, tracking, and map data as inputs fed into the shared feature extractor, denoted as the *stem*. These features are then passed to the *branches* representing an ensemble of motion predictor models. The *gate* is trained on sensor and tracking data to perform the branch selection process. Finally, our multi-modal trajectory fusion algorithm processes the predicted trajectories from the branches and produces the final fused set of trajectories.

We also define a mechanism Ψ for combining the trajectories s_n predicted by the individual sub-models to produce a final decision. Ψ can be a simple aggregation method or a multi-modal fusion approach.

$$s^a_{i:j} = \Psi(s_1, s_2, \ldots s_n) \tag{6}$$

The resulting trajectory predictions are then passed to the *planning* module to generate a feasible, safe motion plan for the ego car. This plan is processed by the *control* module to execute a series of actuator commands to complete the vehicle motion plan.

3.2 CASTNet Architecture

The CASTNet architecture is shown in Figure 3 and is a multi-branch tree-structured model. As such, it implements a shared feature extractor, denoted the *stem*, followed by a set of context-specialized sub-models, or *branches*. The end-to-end workflow of our model is described in Algorithm 1 and consists of the following steps: First, map data M and tracking data T are passed to the *stem* model to extract features $\mathbf{F}$. At the same time, sensor data D and tracking data T are passed to the gate model to identify the current driving context ω. This context is then used to select the set of branches ϕ^* to process $\mathbf{F}$. Then, the selected branches each output a set of predicted trajectories $\mathbf{S}$ for each agent. Finally, the trajectory fusion block fuses the outputs of the different branches to produce a more refined set of predicted trajectories, each compared to the ground truth trajectory for the agent at that timestep $s^a_{i:j}$. CASTNet is a general approach that can extend any motion prediction model to a multi-branch dynamic architecture to improve performance across diverse contexts. We evaluate two variants of our model: (i) a CoverNet [31]/MTP [7] CNN-based architecture that uses raster map data as input, where the stem is a ResNet-18 CNN model and the branches are MLP-based motion predictors, and (ii) a Trajectron++ [35] architecture that uses scene-graph representations as input for a variational graph autoencoder, where the encoder is used as the stem and each branch implements a decoder. In the subsequent sections, we describe each component of our architecture.

3.3 Stem Model

The *stem* model, β, is a shared feature extraction model that learns from all of the driving contexts in the training set. For the CoverNet and MTP variants of CASTNet, the stem uses a ResNet-18

ALGORITHM 1: CASTNet Algorithm

Input: Raster Map (M_a) and Tracking Data (T_a) for agent a,
Sensor Data (D),
Gate Selection Method ($select \in \{top_\kappa, binary\}$)
Output: Fused Motion Predictions $\mathbf{S}^f = \{s_1^f, s_2^f, \ldots, s_{10}^f\}$ for agent a

Initialize branch output vector $\mathbf{S}$
$\mathbf{F} \leftarrow \beta(M_a, T_a)$ `// process raster map using stem`
$\omega \leftarrow \pi(D, T_a)$ `// gate identifies current contexts from sensor data`
if $select = top_\kappa$ **then**
  $\omega' \leftarrow top_\kappa(\omega)$ `// select top-k highest confidence contexts`
else if $select = binary$ **then**
  $\omega' \leftarrow \omega[sigmoid(\omega) > 0.5]$ `// select contexts with confidence score > 0.5`
$\phi^* \leftarrow \phi[\omega']$ `// select branch models to execute`
for $branch$ *in* ϕ^* **do**
  $\mathbf{S}[branch] \leftarrow branch(\mathbf{F})$ `// generate trajectory predictions`
$\mathbf{S}^f \leftarrow$ **Algorithm2**$(\mathbf{S})$ `// multi-modal fusion over branch predictions`

[17] CNN model, similar to the CNN backbones used in References [7, 31]. Trajectron++ uses a graph autoencoder architecture, so the Trajectron++ variants of CASTNet use the encoder model as the stem. For the CNN-based stems, we implement a configurable split point that determines which layers of the model are included in the stem and which layers are added to each branch. In this manner, we can tune the proportion of layers used for shared feature extraction (those in the stem) or context-specific modeling (those in the branches). Having a larger stem can improve the quality of shared features and generalization across contexts. However, having larger branches increases the capacity of the branch models to specialize in their contexts and reduces the amount of feature extraction done by the stem. We provide an ablation study of this stem-branch split point in Section 4.6. After feature extraction, each of the selected branches uses the stem's outputs for context-specific motion prediction.

3.4 Context-aware Gating Model

The gating model π aims to identify the current driving context using sensor and state information. Based on this context identified by the gate model, the appropriate motion prediction model can be selected from the ensemble for that input sample. Since our branches are trained on specific subsets of the dataset and multiple contexts can exist in any given input, we train the gate model as a multi-label classifier to identify all matching contexts for a given input.

We propose several gate model architectures, detailed in Figure 4, and compare the performance of different architectures in ablation studies in Section 4.5. Each gate model processes two prior seconds of front-facing camera images from D and agent tracking information T_a. We capture as much contextual information from the scene as possible by using the front-view image directly as our input instead raster maps that are limited in their ability to model high-level visual context cues such as pedestrian density, road obstructions, road markings, and so on. The CNN Gate spatially models the input images using three 2D convolution layers followed by a max-pooling layer, producing an output feature vector for each image. We then sum these features across the two seconds of history. The Attention gate is implemented similarly to the CNN gate, with an added self-attention layer before the max-pooling layer to attend over the image features. The CNN-LSTM and Attention-LSTM gates are the same as the previous gates but use a two-layer

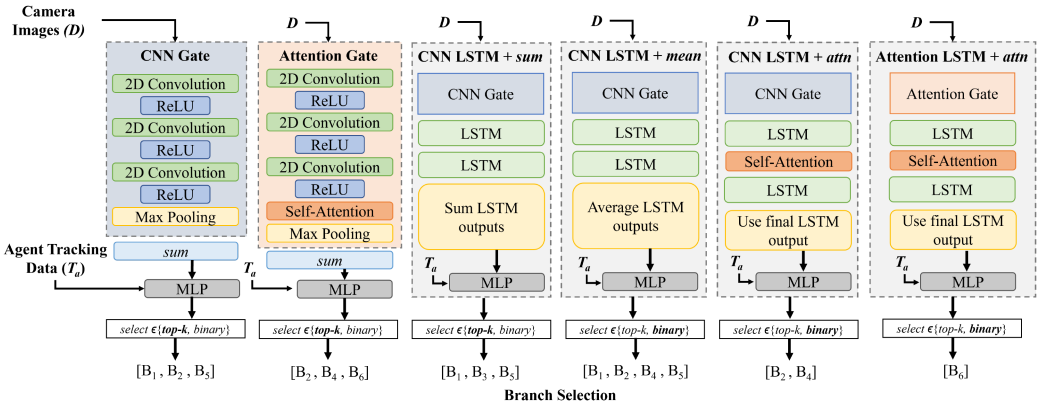

Fig. 4. Various architectural configurations we developed for the gating models. Each gate model takes camera images (D) and agent tracking data T_a as input and outputs the set of branch models to execute. The left three gates are shown with top-κ($k = 3$) gate selection, while the right three gates are shown with *binary* gate selection. The right four gates show the differences between various temporal pooling and spatial modeling options.

long short-term memory (LSTM) network to temporally model the image feature vectors instead of the summation operation used by the CNN Gate and Attention Gate. After the LSTM, we apply a pooling operation to aggregate the temporal features across the input images. We evaluate three temporal pooling operations: *sum* (summation across the outputs), *mean* (averaging across the outputs), and *attn*. The *attn* pooling operation applies self-attention over the first LSTM layer's output for each image, then passes the result to the second LSTM. The last output from the second LSTM is then returned.

After spatio-temporal features have been extracted from the image data, the agent tracking information is concatenated to the features (shown in Figure 3), and the result is passed through an MLP layer, producing the gate output as a logit score for each possible context. We then use one of the following gate selection techniques to identify which branches to execute: *top-*κ (selecting the κ highest-scoring contexts) or *binary* (applying a sigmoid and selecting all contexts with a confidence score > 0.5, or the highest confidence one if none are > 0.5).

3.5 Motion Predictor Ensemble

Each branch $(\phi_1, \phi_2, \ldots, \phi_n)$ contains a motion prediction model (e.g., MTP, CoverNet, Trajectron++), and when CASTNet is deployed, each branch within the ensemble is built with the same model type. The branches are specialized in a specific context by training each branch with the scenes from the dataset labeled as that matching context. This approach differs from training a motion prediction model across the entire dataset, where it may learn to generalize well but perform poorly in specific scenarios. Since the stem is a shared feature extraction backbone that contains the first part of the motion prediction model, we only include the layers from the model that are not in the stem in each branch (e.g., the MLP layers of MTP and CoverNet, the decoder layers in Trajectron++). Varying the backbone split point will adjust the proportion of layers in the shared stem versus each branch, providing insight into the balance between generalized feature extraction performed in the stem and scenario specialization performed in the branches.

As mentioned above, we train each branch with a different subset of the dataset to enable branches to specialize in distinct driving contexts. We define three location-specific contexts, {*city streets, suburban, parking lot*}, and three density-specific contexts, {*high vehicle density, high*

ALGORITHM 2: Trajectory Fusion Algorithm

Input: Branch Motion Predictions: $\mathbf{S} = s_1, s_2, \ldots, s_n$,
Number of Clusters: C,
Time in Future: T,
Fusion Method: $fusion \in \{CI, ITF, KF, Averaging\}$
Output: Fused Motion Predictions $\mathbf{S}^f = \{s_1^f, s_2^f, \ldots, s_{10}^f\}$

$\mathbf{S} \leftarrow mapfilter(\mathbf{S})$ // discard off-road trajectories, s
$\mathbf{S} \leftarrow concat(s_1, s_2, \ldots s_n)$ // remove repeat trajectories
$\Gamma \leftarrow ranking(\mathbf{S})$ // rank trajectories based on confidence scores
$\Delta \leftarrow dist(\mathbf{S})$ // compute matrix of distances between trajectories
$\xi \leftarrow cluster(\Delta, \Gamma, C)$ // cluster trajectories based on similarity
$\mathbf{S}^f, \mathbf{D}, \mathbf{M}$ // initialize fused modes, dynamics and measurement models
for c *in* C **do**
  for t *in* T **do**
    $\mathbf{S}^f[c][t] = fusion(\xi[c][t], \mathbf{D}, \mathbf{M})$
return $\mathbf{S}^f$ // fused mode predictions

pedestrian density, low density}, where samples can have multiple context labels assigned. The set of possible contexts can be redefined, depending on the problem and application; we chose this set of contexts for AV motion prediction, because each of these contexts has a different distribution of motion paths, presenting unique challenges (e.g., high pedestrian density and city street scenarios have more complex agent motion patterns than suburban or low-density driving). Additionally, the selection of location-specific [9, 41] and density-specific [8, 40] contexts is supported in the current literature. Each branch selected by the gate model outputs a set of modes that are then collectively fused by the trajectory fusion component.

3.6 Fusion of Multi-modal Motion Predictions

For each agent in a scene, each active branch outputs multiple possible modes with varying confidence levels. The goal of Ψ, the trajectory fusion function, is to produce a set of fused modes that are more accurate than the individual modes. The functionality of Ψ is described in Algorithm 2 and consists of the following steps: First, if map data is available, then we can perform *map filtering*, filtering out trajectories that exit the drivable road area. Then, all branch outputs are concatenated, filtered for repeat trajectories, and ranked according to the confidence of each mode. Next, a distance matrix of the average displacement error between modes is computed and passed through a hierarchical agglomerative clustering method that recursively groups modes together, starting with the closest pair of nodes until an entire tree structure of clustered modes is created. Finally, the modes in each cluster are combined according to a fusion method to produce a fused mode prediction for that cluster. These fused modes are subsequently ranked by their confidence and define the final set of output predictions.

We evaluate our architecture with two fusion methods: **Kalman filtering (*KF*)** and *Averaging*—calculating the point-wise average across all modes per cluster for each timestep. *KF* is performed as follows using the general form of the discretized linear dynamics of a system with state $\mathbf{s}_{\mathbf{KF}}^{\mathbf{f}}$ (x,y position of the target vehicle) and measurements $\mathbf{z}$ (branch motion predictions to fuse) at timestep t, given as:

$$\mathbf{s}_{\mathbf{KF}}^{\mathbf{f}}(t) = \mathbf{A}\mathbf{s}_{\mathbf{KF}}^{\mathbf{f}}(t-1) + \mathbf{v}(t), \tag{7}$$

$$\mathbf{z}(t) = \mathbf{H}\mathbf{s}_{\mathbf{KF}}^{\mathbf{f}}(t) + \mathbf{w}(t), \tag{8}$$

where $\mathbf{A}$ is the state transition matrix; $\mathbf{v}$ is the process noise vector, which is modeled as zero-mean, normally distributed random variable with covariance, $\mathbf{Q}$; $\mathbf{H}$ is the measurement matrix relating the state to the measurements; and $\mathbf{w}$ is the measurement noise vector, which also is zero-mean with a normal distribution and covariance, $\mathbf{R}$. During the prediction step of *KF*, the state estimate, $\mathbf{s}^{\mathbf{f}}_{\mathbf{KF}}$, and its estimation error covariance matrix, $\mathbf{P}(k)$, are propagated forward through the dynamics model with the added process noise. The prediction equations are as follows:

$$\mathbf{s}^{\mathbf{f}}_{\mathbf{KF}}(t|t-1) = \mathbf{A}\,\mathbf{s}^{\mathbf{f}}_{\mathbf{KF}}(t-1|t-1), \tag{9}$$

$$\mathbf{P}(t|t-1) = \mathbf{A}\,\mathbf{P}(t-1|t-1)\,\mathbf{A}^{\top} + \mathbf{Q}(t-1), \tag{10}$$

where the notation $(t+1|t)$, indicates the next timestep given the current timestep. Next, during the update step, measurements ($\mathbf{z}$), in the form of the selected branch motion predictions ($\mathbf{s}_i$), are sequentially used to update the state estimate and its covariance. The measurement update equations are as follows:

$$\mathbf{s}^{\mathbf{f}}_{\mathbf{KF}}(t|t) = \mathbf{s}^{\mathbf{f}}_{\mathbf{KF}}(t|t-1) - \mathbf{K}(t)[\mathbf{H}(\mathbf{s}^{\mathbf{f}}_{\mathbf{KF}}(t|t-1) - \mathbf{z}(t)], \tag{11}$$

$$\mathbf{P}(t|t) = \mathbf{P}(t|t-1) - \mathbf{K}(t)\mathbf{H}\mathbf{P}(t|t-1), \tag{12}$$

$$\mathbf{K}(t) = \mathbf{P}(t|t-1)\mathbf{H}^{\top}[\mathbf{H}\mathbf{P}(t|t-1)\mathbf{H}^{\top} + \mathbf{R}(t)]^{-1}, \tag{13}$$

with $\mathbf{K}$ representing the Kalman gain. The prediction and update step are iterated for the available number of timesteps T to produce an estimate of the state, $\mathbf{s}^{\mathbf{f}}_{\mathbf{KF}}$, and its associated estimation error covariance, $\mathbf{P}$, representing the uncertainty involved with the state estimate.

The measurement noise covariance matrix was set at $\mathbf{R} = [5, 0; 0, 5]$. Additionally, for *KF*, a 2D constant-velocity dynamics model for $\mathbf{A}$ and $\mathbf{H}$ was utilized with a process noise covariance matrix $\mathbf{Q} = [12.5, 2.5e\text{-}4; 2.5e\text{-}4, 12.5]$. The initial estimate was set to the last known position of the target agent, with an initial covariance matrix $\mathbf{P} = [1, 0; 0, 1]$. Tuning for the multi-modal fusion parameters for the Kalman filter ($\mathbf{Q}$,$\mathbf{R}$), including the number of modes and clusters, was conducted over the training dataset. We additionally implemented an approach that tuned the Kalman filter based on the confidence values returned by our deep-learning motion prediction models; however, the performance of these methods was worse than the statically defined noise values, as the deep-learning models' confidence predictions were not representative of their true accuracy to the ground truth trajectories.

4 EXPERIMENTS

4.1 Experimental Setup

We evaluate CASTNet on the nuScenes dataset [4] and use the training/validation sets and metrics defined by the nuScenes prediction challenge. Since the nuScenes test set annotations are not provided, we present results using the nuScenes validation set. For each agent a in a particular sample, two seconds of historical data are given (M_a, T_a, and D). Each model then predicts the agent's future location over the next 6 seconds at 2 Hz, yielding 12 (x,y) predictions.

The distribution of the different contexts across all samples in the dataset is as follows: 40% for *city streets*, 31% for *suburban*, 5% for *parking lot*, 31% for *high vehicle density*, 15% for *high pedestrian density*, 28% for *low density*. We note that a single sample can span multiple contexts, which accounts for the overlap between context subsets.

We compare CASTNet against several state-of-the-art motion prediction techniques including MTP [7], CoverNet [31], and Trajectron++ (T++) [35], introduced in Section 2. We evaluate MTP configured with either one or three output modes. We configured CoverNet with 64 (ϵ=8), 415 (ϵ=4), or 2,206 output modes (ϵ=2), corresponding to the fixed trajectory sets defined in their paper.

The evaluated MTP and CoverNet baselines use a ResNet-18 [17] backbone. We evaluate Trajectron++ with dynamics information (dyn) and dynamics and map information (dyn, map). We also compare CASTNet against one physics-based baseline that makes trajectory predictions assuming a **constant agent velocity and heading (CVH)**. The CVH model assumes the acceleration and heading of each agent are constant and then computes the future trajectory given the past kinematics data, time window horizon of prediction, and frequency of measurements. We used a pre-trained Trajectron++ model that was trained on nuScenes. We trained MTP and CoverNet following the training procedures stated in the respective papers.

For the MTP and CoverNet variants of CASTNet, training from scratch took longer but yielded better performance than initializing the model with pre-trained ImageNet weights and training on nuScenes for additional epochs as was done in References [7] and [31]. As such, we trained the stem/branch models simultaneously from scratch on the nuScenes dataset for 400 epochs using a learning rate of 1e-7, the Adam optimizer, and a batch size of 16. On a single GPU, training took ∼120 hours. Training the MTP and CoverNet baselines took ∼100 hours, as we found that performance could be improved by extending training to 150 epochs. Since nuScenes does not annotate the high-level context of each scene, we used three human annotators to label the matching contexts for each scene in nuScenes. The stem was trained with data from all contexts, while the branches were trained as described in Section 3.5. For the Trajectron++ variants of CASTNet, we use the encoder layers as the stem and the decoder layers as each branch. With these variants, we initialized the model from pre-trained Trajectron++ weights. The best performance was achieved by freezing the pre-trained Trajectron++ weights in the stem and refining the branch weights with additional training for 10 epochs, which took ∼40 hours. *CASTNet-CN* and *CASTNet-T* denote CASTNet with CoverNet stem/branches and Trajectron++ stem/branches, respectively. All CASTNet results shown are with the AttentionLSTM gate and *attn* pooling, except Table 5. We used a learning rate of 2e-4, the Adam optimizer, and a batch size of 64 to train our gate models for 100 epochs. We trained and evaluated all models on a Linux server with an Intel Xeon E5-2630 CPU and an Nvidia Titan Xp GPU.

We use the following metrics commonly used to score motion prediction [7, 31, 35] to evaluate each approach.

- ***Minimum Average Displacement Error over k (minADE$_k$)*** meters: Computed by taking the average of the point-wise L2 distances between the predicted trajectory and the ground truth for the k most likely predictions before selecting the minimum value.
- ***Miss Rate at 2 Meters over k (MR2$_k$)*** meters: This metric reports how often the k most likely predictions have a point-wise Euclidean distance greater than 2 meters from the ground truth trajectory.
- ***Final Displacement Error over k (FDE$_k$)*** meters: L2 distance between the final position of each agent and the predicted final position of the agent for the k most likely prediction before averaging across all agents.

4.2 nuScenes Evaluation

We evaluate the performance improvements gained by CASTNet-CN compared to the baseline CoverNet (ϵ=4) on the nuScenes validation set in Table 1. In this section, CASTNet is evaluated with Attention-LSTM gating and two mode fusion variations: **Kalman filtering** (***KF***) and **Averaging** (***avg.***). CASTNet-CN improves upon the CoverNet baseline across all metrics for all the variations of gate selections and late fusion. We bold the results for each gate selection method for the best-performing late fusion method. As shown in Table 1, *KF* outperforms *avg.* on all metrics except the $\gamma = top-3$ gating method, where *avg.* marginally beats *KF* for minADE$_1$, minADE$_5$, FDE$_1$. The

Table 1. Performance Improvement when Using CASTNet-CN (ϵ=4) over CoverNet (ϵ=4) on the nuScenes Dataset

Model (Configuration)	minADE_k			MR2_k		FDE_k		
	k=1	k=5	k=10	k=5	k=10	k=1	k=5	k=10
CoverNet (ϵ=4) [31]	7.556	3.855	2.963	0.972	0.962	16.732	8.186	5.966
CASTNet-CN (α, avg.)	6.149	2.266	1.787	0.890	0.794	12.385	4.146	2.980
CASTNet-CN (α, *KF*)	**6.107**	**2.262**	**1.782**	**0.883**	**0.789**	**12.339**	**4.123**	**2.953**
CASTNet-CN (γ, avg.)	**6.151**	**2.294**	1.802	0.901	0.799	**12.325**	4.180	3.002
CASTNet-CN (γ, *KF*)	6.152	**2.294**	**1.798**	**0.896**	**0.795**	12.326	**4.163**	**2.980**
CASTNet-CN (σ, avg.)	6.143	2.277	1.789	0.895	0.797	12.356	4.154	2.980
CASTNet-CN (σ, *KF*)	**6.104**	**2.273**	**1.786**	**0.888**	**0.792**	**12.314**	**4.132**	**2.955**

Gate Selection Legend: α=*top-1*, γ=*top-3*, σ=*binary*

Table 2. Performance Improvement when Using CASTNet-T over T++ on the nuScenes Dataset

Model (Configuration)	minADE_k			MR2_k		FDE_k		
	k=1	k=5	k=10	k=5	k=10	k=1	k=5	k=10
T++ (dyn, map) [35]	4.264	2.319	1.887	**0.614**	**0.615**	10.063	5.086	3.952
CASTNet-T (α, avg.)	3.418	**1.597**	1.334	0.683	0.671	8.092	3.689	2.944
CASTNet-T (α, *KF*)	**3.393**	1.598	**1.333**	0.676	0.666	**8.043**	**3.684**	**2.938**

Gate Selection Legend: α=*top-1*

underlined results indicate the best performance across all variations, as the CASTNet-CN ($\alpha = top-1$, *KF*) performs the best across the majority of metrics, achieving a **41.3%** lower minADE_5 than the CoverNet baseline. This performance improvement showcases the ability of our modular, model-agnostic approach to improve existing motion prediction methods. Binary gate selection achieves lower minADE_1 and FDE_1 than *top-1* and *top-3* gate selection, but *top-1* performs best across all metrics. Finally, the FDE results for CASTNet-CN show significant improvement upon the baseline, nearly decreasing the error by 50% for FDE_5 and FDE_{10}. The high FDE_1 results indicate that future work should focus on improving trajectory predictions at longer time horizons.

The results for T++ and the CASTNet-T variants are shown in Table 2. Based on CASTNet-CN's performance for the gate selection method, we tested our CASTNet-T models with Attention-LSTM gating using *top-1* selection and evaluated the same two mode fusion variations (*KF* and *avg.*). T++ uses slightly different inputs (additional information regarding pedestrians within the scene) and evaluation samples (due to T++ only requiring 1 second of history vs. 2 seconds, allowing for more viable agents in a scene). The CASTNet-T variants outperform the baseline T++ model on all metrics except for MR2. The MR2 is calculated as the proportion of misses over all agents in a scene—so, the fact that T++ achieves better MR2 metrics but worse minADE metrics indicates that T++'s predictions can be more conservative, but overall less accurate than CASTNet-T's predictions. Better metrics for CASTNet-T are achieved when *KF* is used instead of *avg.* for trajectory fusion, except for minADE_5 where *avg.* nominally beats *KF*. Compared to the CASTNet-CN variants, the CASTNet-T variants achieved much better MR2 scores, indicating that CASTNet-T's predictions were relatively close together for each agent.

We further evaluate baseline motion predictors on the nuScenes validation set in Table 3. Compared to the CVH, MTP, and CoverNet baselines, CASTNet-CN (α, *KF*, ϵ=4) achieves the best minADE_{10}, minADE_5, MR2_{10}, FDE_5, and FDE_{10} scores. MTP (modes=1) and CVH only predict one mode, so their scores across k remain constant. Although this allows for more fine-tuned results on k=1 with good performance, they are often less robust in predicting challenging trajectories,

Table 3. Comparison of Motion Prediction Results on the nuScenes Dataset

Model (Configuration)	minADE_k			MR2_k		FDE_k		
	k=1	k=5	k=10	k=5	k=10	k=1	k=5	k=10
Physics-based (CVH)	4.613	4.613	4.613	0.912	0.912	11.21	11.21	11.21
MTP (modes=1) [7]	3.912	3.912	3.912	0.945	0.945	9.271	9.271	9.271
MTP (modes=3) [7]	4.236	3.492	3.492	0.880	0.880	9.911	7.479	7.479
CoverNet (ϵ=8) [31]	6.438	2.628	2.272	0.922	0.918	13.44	5.086	3.898
CoverNet (ϵ=4) [31]	7.556	3.855	2.963	0.972	0.962	16.73	8.186	5.966
CoverNet (ϵ=2) [31]	12.96	5.545	3.801	0.896	0.816	23.33	11.11	7.473
CASTNet-CN (α, *KF*)	6.107	**2.262**	**1.782**	0.883	**0.789**	12.34	**4.123**	**2.953**

Gate Selection Legend: α=*top-1*

leading to worse performance as k increases and potential safety concerns with only one prediction available. MTP (modes=1) performed best on the k=1 metrics, likely because the model specialized for this task due to its single output mode. CoverNet (ϵ=8) outperforms the other CoverNets, because the smaller output reduces model complexity and improves convergence. MTP (modes=3) achieves higher k=5,10 metrics than MTP with one mode. Although CASTNet does not achieve the best results for minADE_1, it significantly improves upon both minADE_5 and minADE_{10}. Overall, the results from Tables 1, 2, and 3 indicate that integrating motion predictors with CASTNet can improve their performance significantly and enable them to outperform state-of-the-art motion prediction models across most metrics.

4.3 Inference Latency and Model Size Analysis

In Table 4, we compare our approach's inference latency and model size with the baseline methods. This section evaluates our CASTNet models with *avg.* trajectory mode fusion. As shown in Table 4, our CASTNet-CN variants have a larger model size but a minimal latency increase compared to the baseline CoverNet (ϵ=4) model. The larger model size can be attributed to the multi-branch architecture of the model, while the slight latency increase results from the addition of the gating model. Top-1 (α) gate selection results in the lowest CASTNet-CN inference latency. These trends repeat for the CASTNet-T models compared to the T++ baseline. T++ incurs significantly higher latency than MTP and CoverNet but uses far fewer parameters. This difference is likely attributable to T++'s use of scene-graph modeling and heavy reliance on temporal models, while MTP and CoverNet use CNNs and MLPs to model image data. Our approach is computationally expensive at the training stage, as it employs an ensemble method; however, during inference time, it achieves latency close to the baseline models used within its ensemble, making it on par with current approaches in terms of resources necessary while deployed.

4.4 Scenario-specific Evaluation

Figure 5 presents each approach's scenario-specific minADE_{10} for six different driving contexts. In the subsection, we show results from our CASTNet variants using *KF* trajectory mode fusion. The results show that CASTNet-T (dyn) outperforms CVH, MTP, CoverNet, and T++ across all scenarios. The limitations of CVH, MTP, and CoverNet are most visible in the *high ped. density* and *high veh. density* contexts, as their error increases significantly while our approach remains stable. We also find that both CASTNet-CN and CASTNet-T perform more consistently across scenarios than T++, achieving a $\text{minADE}_{10} < 2$ in all contexts, while T++ varies from 1.6–2.26, performing worst in *low density* and *suburban* contexts. Notably, CASTNet-T (dyn) outperforms T++ (dyn, map) in every context evaluated, even though T++ (dyn, map) benefits from additional map information.

Table 4. Inference Latency Comparison between Baseline Models and CASTNet Variants

Model (Configuration)	Latency (ms)	Num. Params.
MTP (modes=1) [7]	10.50	13M
MTP (modes=3) [7]	10.80	14M
CoverNet (ϵ=8) [31]	9.97	14M
CoverNet (ϵ=4) [31]	10.03	15M
CoverNet (ϵ=2) [31]	10.63	22M
CASTNet-CN (α, ϵ=4)	11.10	34M
CASTNet-CN (γ, ϵ=4)	12.98	34M
CASTNet-CN (σ, ϵ=4)	12.24	34M
T++ (dyn) [35]	86.60	0.3M
CASTNet-T (α, dyn)	90.98	1.2M

Gate Selection Legend: α=*top-1*, γ=*top-3*, σ=*binary*

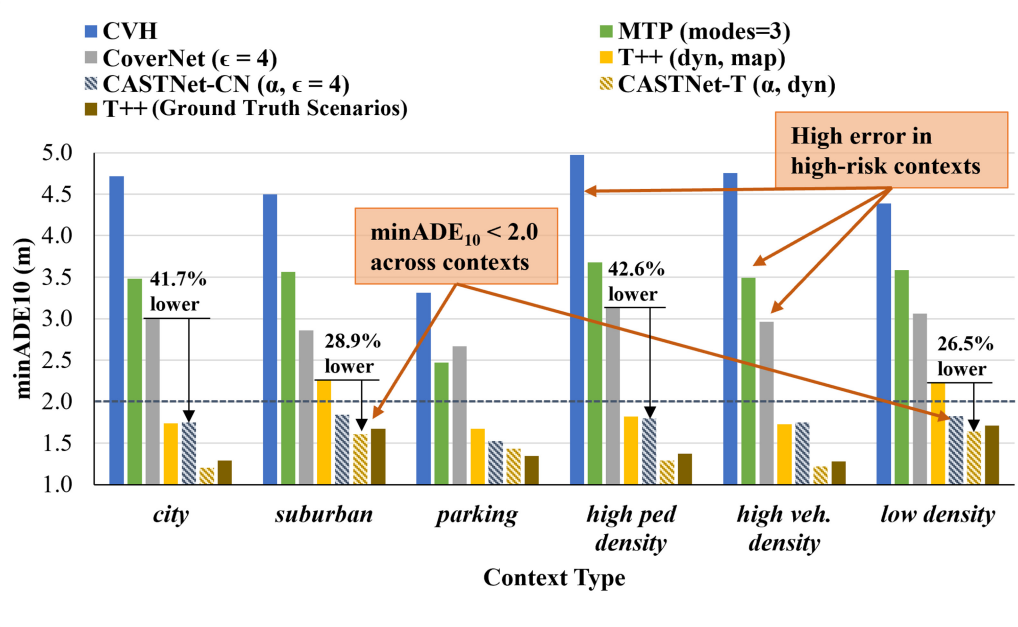

Fig. 5. Motion prediction results for each approach in different driving contexts of the nuScenes dataset.

We also show the performance of each branch within CASTNet-T on their respective ground truth scenarios, denoted as T++ (Ground Truth Scenarios). Interestingly, CASTNet-T outperforms these specialized branches across all scenarios except for *parking*, which can likely be attributed to the benefits of using a shared stem with a learned gating model and varying the branch selection for each input sample. These results indicate that CASTNet is more robust to variability across driving scenarios and highlights the benefits of contextual modeling.

4.5 Gating Evaluation

We evaluate the performance of our gate models in Table 5 in terms of minADE_5 score and MR2_5 score when used in our CASTNet-CN (α, ϵ=4) model with *KF* fusion. We also show each gate's

Table 5. Evaluation of Our Proposed Gating Models

Gate Model		Temp. Pooling	Acc.	Coverage Error	minADE_5	MR2_5
Spatial	Temporal					
CNN	None	*sum*	13.85%	3.478	2.287	0.886
Attn.	None	*sum*	9.79%	3.537	2.347	0.889
CNN	LSTM	*sum*	13.60%	3.354	2.296	0.885
CNN	LSTM	*mean*	7.64%	3.302	2.274	**0.882**
CNN	LSTM	*attn*	**22.17%**	**3.171**	2.279	0.883
Attn.	LSTM	*sum*	**19.59%**	3.332	2.284	0.884
Attn.	LSTM	*mean*	15.44%	3.299	**2.274**	0.883
Attn.	LSTM	*attn*	18.08%	**3.248**	**2.262**	**0.883**

multi-label classification accuracy. For this metric, the prediction for a sample is only deemed correct if the predicted labels match the ground truth. As such, this metric does not support partially correct predictions, which are typical for multi-label classifiers. To address this gap, we include coverage error [38], which measures how many branches must be selected until all the ground-truth positive labels are covered; a lower coverage error indicates a better multi-label classifier. A perfect gate model would achieve a coverage error of 1.937 on this dataset.

The results show that using a temporal model slightly improves all evaluated metrics. Interestingly, using attention in the spatial model only benefits when a temporal model is included. Regarding temporal pooling, *attn* pooling outperforms both *sum* and *mean* pooling, and *mean* pooling is slightly better than *sum*. Overall, we found that the AttentionLSTM gate with *attn* pooling results in the lowest minADE_5. The CNN LSTM gate with *attn* pooling achieved the highest accuracy and lowest coverage error, meaning it could better predict each sample's exact set of matching labels.

4.6 Architecture Ablation Study

To illustrate CASTNet's modularity, we explore the performance of different architectural configurations in Table 6. We evaluate both CoverNet (ϵ=8) and MTP (modes=16) branch configurations and two backbone split points (we note that 16 modes were used here to allow for an appropriate evaluation across k=10). The results shown are with an AttentionLSTM gate with *attn* pooling, *top-1* gate selection, and *KF* mode fusion. Since ResNet-18 consists of four CNN blocks, *Block-1* indicates that the first block of the ResNet-18 backbone is located in the stem, and each branch contains the remaining three blocks and the motion predictor's MLP layers. *Stem* indicates that all four blocks are in the stem, and each branch contains only the motion predictor's MLP layers. Table 6 shows that splitting at *Block-1* leads to significantly larger model size and diminished performance. This result suggests using a larger shared feature extractor is more effective than larger context-specific feature extractors. The results also show a narrower gap between the performance of *Block-1* versus *Stem* with CoverNet branches than with MTP branches. Overall, these results are significant, as they demonstrate that using *smaller* network sizes can *improve* performance—inspiring further research into searching for optimal split point locations.

4.7 Discussion

Our results demonstrate that CASTNet can convert existing motion prediction models into dynamic, context-aware architectures, improving robustness across driving scenarios. Our method for splitting and gating the model does not require significant algorithmic changes; thus, real-world implementations can be developed relatively quickly. Although we trained and tested using six explicitly defined contexts informed by related works, this approach may limit the model in

Table 6. Ablation Study on Different Architectural Configurations of Our CASTNet Model

CASTNet Branch Type	Split Point	Num. Params.	minADE_5	minADE_{10}
CoverNet ($\epsilon = 8$)	*Block-1*	81M	3.189	2.367
CoverNet ($\epsilon = 8$)	*Stem*	24M	**3.131**	**2.229**
MTP (modes=16)	*Block-1*	89M	3.644	3.107
MTP (modes=16)	*Stem*	34M	**3.228**	**2.636**

more complex settings. Further ablation studies could be performed on scenarios with overlapping contexts. In our approach, due to the limitations of the dataset, we used human annotators to label the contexts present; however, this raises concerns about scalability and cost. An interesting future direction would be to explore learned methods for defining contexts and splitting data for branch specialization instead of heuristically defining scenarios.

While we showed the efficacy of CNN-based gating strategies, using methods that integrate knowledge or stem features could improve performance. It would be insightful to evaluate how more complex gating and fusion approaches improve the generalization and accuracy of CASTNet in future works. Similarly, future works could explore different mode fusion strategies. Our Kalman filter's dynamics models for different classes of objects (e.g., cars, pedestrians, bicycles) within the filter could further improve results if object classifications are performed accurately.

The CASTNet architecture can be particularly beneficial in scenarios requiring high levels of adaptation. However, one limitation of CASTNet is its reliance on sensing data, which may not always be guaranteed in contested sensing environments that may cause sensor failures. An interesting area of future work revolves around enabling further levels of adaptation by training gate models with different subsets of sensor data (e.g., camera data only, lidar data only, camera and lidar data) to improve the robustness to a variety of sensing conditions. Additionally, we recognize that our results and analysis are constrained to the nuScenes dataset and that using further datasets and simulation platforms can allow for further studies on the effects of parameters, such as several different scenarios and the amount of prior data used by the models. Although nuScenes is currently the most popular motion prediction dataset, CASTNet's performance could be evaluated on newer datasets like Lyft Level 5 [19] and Waymo Open Motion Dataset [11] once trained motion prediction model weights for these datasets become publicly available.

5 CONCLUSION

AV motion prediction is a challenging multi-domain problem that can vary in complexity, depending on the driving context. In this work, we proposed CASTNet, a dynamic context-aware motion prediction approach for AVs. CASTNet is a highly modular approach that employs an ensemble of motion predictors and a deep learning–based context identification module that helps to improve the robustness of prediction performance in challenging environments. This work demonstrated that CASTNet outperforms state-of-the-art motion predictors across driving contexts on a real-world driving dataset. Additionally, we showed that spatio-temporal attention could be used to model the driving context from vehicle sensor data effectively. We conducted extensive ablation studies on our proposed architecture including evaluation of architectural configurations and context identification models. We also proposed several strategies for performing multi-modal trajectory fusion and illustrated the modularity of our approach by evaluating multiple different motion prediction backbones within CASTNet. Since real-world driving consists of a wide range of diverse driving scenarios, CASTNet is likely to perform better in a real AV than existing methods, due to its ability to adapt the architecture to maximize performance in each driving context. The

key research findings demonstrated in this work are beneficial to providing increased levels of autonomy in CPS, and the techniques demonstrated can be applied to other CPS that utilize sensors to perceive and predict actions of other entities in the surrounding environment.

ACKNOWLEDGMENTS

The authors would like to thank Harsimrat Kaeley, Anurag Karra, and Sebastien Lajeunesse-Degroot for the help in experimentation. Any opinions, findings, conclusions, or recommendations expressed in this article are those of the authors and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] Yaakov Bar-Shalom, X. Rong Li, and Thiagalingam Kirubarajan. 2004. *Estimation with Applications to Tracking and Navigation: Theory Algorithms and Software*. John Wiley & Sons.

[2] Kirstie Bellman, Christopher Landauer, Nikil Dutt, Lukas Esterle, Andreas Herkersdorf, Axel Jantsch, Nima TaheriNejad, Peter R. Lewis, Marco Platzner, and Kalle Tammemäe. 2020. Self-aware cyber-physical systems. *ACM Trans. Cyber-phys. Syst.* 4, 4 (2020), 1–26.

[3] Thibault Buhet, Emilie Wirbel, Andrei Bursuc, and Xavier Perrotton. 2021. PLOP: Probabilistic polynomial objects trajectory prediction for autonomous driving. In *Conference on Robot Learning*. PMLR, 329–338.

[4] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. 2020. nuScenes: A multimodal dataset for autonomous driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'20)*. 11621–11631.

[5] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. 2019. MultiPath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. In *3rd Conference on Robot Learning (CoRL'19)*.

[6] Bo Chen, Decai Li, and Yuqing He. 2021. Simultaneous prediction of pedestrian trajectory and actions based on context information iterative reasoning. In *IEEE International Conference on Intelligent Robots and Systems (IROS'21)*. 1007–1014. DOI: https://doi.org/10.1109/IROS51168.2021.9636252

[7] Henggang Cui, Vladan Radosavljevic, Fang-Chieh Chou, Tsung-Han Lin, Thi Nguyen, Tzu-Kuo Huang, Jeff Schneider, and Nemanja Djuric. 2019. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *IEEE International Conference on Robotics and Automation (ICRA'19)*. IEEE, 2090–2096.

[8] Nachiket Deo, Akshay Rangesh, and Mohan M. Trivedi. 2018. How would surround vehicles move? A unified framework for maneuver classification and motion prediction. *IEEE Trans. Intell. Vehic.* 3, 2 (2018), 129–140.

[9] Nemanja Djuric, Henggang Cui, Zhaoen Su, Shangxuan Wu, Huahua Wang, Fang-Chieh Chou, Luisa San Martin, Song Feng, Rui Hu, Yang Xu, Alyssa Dayan, Sidney Zhang, Brian C. Becker, Gregory P. Meyer, Carlos Vallespi-Gonzalez, and Carl K. Wellington. 2021. MultiXNet: Multiclass multistage multimodal motion prediction. In *IEEE Intelligent Vehicles Symposium (IV'21)*. IEEE, 435–442.

[10] Nemanja Djuric, Vladan Radosavljevic, Henggang Cui, Thi Nguyen, Fang-Chieh Chou, Tsung-Han Lin, Nitin Singh, and Jeff Schneider. 2020. Uncertainty-aware short-term motion prediction of traffic actors for autonomous driving. In *IEEE/CVF Winter Conference on Applications of Computer Vision*. 2095–2104.

[11] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R. Qi, Yin Zhou, Zoey Yang, Aurelien Chouard, Pei Sun, Jiquan Ngiam, Vijay Vasudevan, Alexander McCauley, Jonathon Shlens, and Dragomir Anguelov. 2021. Large scale interactive motion forecasting for autonomous driving: The Waymo Open Motion Dataset. In *IEEE/CVF International Conference on Computer Vision*. 9710–9719.

[12] Ilias Gerostathopoulos, Tomas Bures, Petr Hnetynka, Adam Hujecek, Frantisek Plasil, and Dominik Skoda. 2017. Strengthening adaptation in cyber-physical systems via meta-adaptation strategies. *ACM Trans. Cyber-phys. Syst.* 1, 3 (2017), 1–25.

[13] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. 2022. GOHOME: Graph-oriented heatmap output for future motion estimation. In *International Conference on Robotics and Automation (ICRA'22)*. IEEE, 9107–9114.

[14] Cosmin Ginerica, Mihai Zaha, Florin Gogianu, Lucian Busoniu, Bogdan Trasnea, and Sorin Grigorescu. 2021. ObserveNet control: A vision-dynamics learning approach to predictive control in autonomous vehicles. *IEEE Robot. Automat. Lett.* 6, 4 (2021), 6915–6922.

[15] Sorin Mihai Grigorescu, Bogdan Trasnea, Liviu Marina, Andrei Vasilcoi, and Tiberiu Cocias. 2019. NeuroTrajectory: A neuroevolutionary approach to local state trajectory learning for autonomous vehicles. *IEEE Robot. Automat. Lett.* 4, 4 (2019), 3441–3448.

[16] Kazumune Hashimoto, Natsuko Tsumagari, and Toshimitsu Ushio. 2022. Collaborative rover-copter path planning and exploration with temporal logic specifications based on Bayesian update under uncertain environments. *ACM Trans. Cyber-phys. Syst.* 6, 2 (2022), 1–24.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'16)*. 770–778.

[18] Matti Henning, Johannes Christian Muller, Fabian Gies, Michael Buchholz, and Klaus Dietmayer. 2022. Situation-aware environment perception using a multi-layer attention map. *IEEE Trans. Intell. Vehic.* 8, 1 (2022).

[19] John Houston, Guido Zuidhof, Luca Bergamini, Yawei Ye, Long Chen, Ashesh Jain, Sammy Omari, Vladimir Iglovikov, and Peter Ondruska. 2021. One thousand and one hours: Self-driving motion prediction dataset. In *Conference on Robot Learning*. PMLR, 409–418.

[20] Xin Huang, Stephen G. McGill, Brian C. Williams, Luke Fletcher, and Guy Rosman. 2019. Uncertainty-aware driver trajectory prediction at urban intersections. In *International Conference on Robotics and Automation (ICRA'19)*. IEEE, 9718–9724.

[21] Nikita Japuria, Golnaz Habibi, and Jonathan How. 2017. CASNSC: A context-based approach for accurate pedestrian motion prediction at intersections. In *Conference and Workshop on Neural Information Processing Systems (NEURIPS'17)*.

[22] Gihoon Kim, Dongchan Kim, Yoonyong Ahn, and Kunsoo Huh. 2021. Hybrid approach for vehicle trajectory prediction using weighted integration of multiple models. *IEEE Access* 9 (2021), 78715–78723.

[23] Stéphanie Lefèvre, Dizan Vasquez, and Christian Laugier. 2014. A survey on motion prediction and risk assessment for intelligent vehicles. *ROBOMECH J.* 1, 1 (2014), 1–14.

[24] Lingyun Luke Li, Bin Yang, Ming Liang, Wenyuan Zeng, Mengye Ren, Sean Segal, and Raquel Urtasun. 2020. End-to-end contextual perception and prediction with interaction transformer. In *IEEE International Conference on Intelligent Robots and Systems (IROS'20)*. IEEE, 5784–5791.

[25] Arnav Vaibhav Malawade, Trier Mortlock, and Mohammad Abdullah Al Faruque. 2022. HydraFusion: Context-aware selective sensor fusion for robust and efficient autonomous vehicle perception. In *ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPS'22)*.

[26] Arnav Vaibhav Malawade, Trier Robert Mortlock, and Mohammad Abdullah Al Faruque. 2022. EcoFusion: Energy-aware adaptive sensor fusion for efficient autonomous vehicle perception. In *Design Automation Conference (DAC'22)*.

[27] Yanbing Mao, Yuliang Gu, Naira Hovakimyan, Lui Sha, and Petros Voulgaris. 2023. SL1-simplex: Safe velocity regulation of self-driving vehicles in dynamic and unforeseen environments. *ACM Trans. Cyber-phys. syst.* 7, 1 (2023), 1–24.

[28] Kaouther Messaoud, Itheri Yahiaoui, Anne Verroust-Blondet, and Fawzi Nashashibi. 2020. Attention based vehicle trajectory prediction. *IEEE Trans. Intell. Vehic.* 6, 1 (2020), 175–185.

[29] Diksha Moolchandani, Kishore Yadav, Geesara Kulathunga, Ilya Afanasyev, Anshul Kumar, Manuel Mazzara, and Smruti Sarangi. 2022. Game theory–based parameter tuning for energy-efficient path planning on modern UAVs. *ACM Trans. Cyber-phys. Syst.* 6, 4 (2022), 1–29.

[30] Alexandra S. Mueller, Jessica B. Cicchino, and David S. Zuby. 2020. What humanlike errors do autonomous vehicles need to avoid to maximize safety? *J. Safety Res.* 75 (2020), 310–318.

[31] Tung Phan-Minh, Elena Corina Grigore, Freddy Boulton, Oscar Beijbom, and Eric Wolff. 2020. CoverNet: Multimodal behavior prediction using trajectory sets. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'20)*. 14074–14083.

[32] Shreyas Ramakrishna, Zahra Rahiminasab, Gabor Karsai, Arvind Easwaran, and Abhishek Dubey. 2022. Efficient out-of-distribution detection using latent space of β-VAE for cyber-physical systems. *ACM Trans. Cyber-phys. Syst.* 6, 2 (2022), 1–34.

[33] Nicholas Rhinehart, Rowan McAllister, Kris Kitani, and Sergey Levine. 2019. PRECOG: Prediction conditioned on goals in visual multi-agent settings. In *IEEE/CVF International Conference on Computer Vision*. 2821–2830.

[34] Markus Roth, Jork Stapel, Riender Happee, and Dariu M. Gavrila. 2021. Driver and pedestrian mutual awareness for path prediction and collision risk estimation. *IEEE Trans. Intell. Vehic.* 7, 4 (2021).

[35] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. 2020. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *European Conference on Computer Vision (ECCV'20)*. Springer, 683–700.

[36] Xiaolin Tang, Kai Yang, Hong Wang, Jiahang Wu, Yechen Qin, Wenhao Yu, and Dongpu Cao. 2022. Prediction-uncertainty-aware decision-making for autonomous vehicles. *IEEE Trans. Intell. Vehic.* 7, 4 (2022), 849–862.

[37] Saideep Tiku, Prathmesh Kale, and Sudeep Pasricha. 2021. QuickLoc: Adaptive deep-learning for fast indoor localization with mobile devices. *ACM Trans. Cyber-phys. Syst.* 5, 4 (2021), 1–30.

[38] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. 2009. Mining multi-label data. In *Data Mining and Knowledge Discovery Handbook*. Springer, 667–685.

[39] Balakrishnan Varadarajan, Ahmed Hefny, Avikalp Srivastava, Khaled S. Refaat, Nigamaa Nayakanti, Andre Cornman, Kan Chen, Bertrand Douillard, Chi Pang Lam, Dragomir Anguelov, and Benjamin Sapp. 2022. Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction. In *International Conference on Robotics and Automation (ICRA'22)*. IEEE, 7814–7821.

[40] Jinghua Wang, Zhao Zhang, Feng Liu, and Guangquan Lu. 2021. Investigating heterogeneous car-following behaviors of different vehicle types, traffic densities and road types. *Transport. Res. Interdiscip. Perspect.* 9 (2021), 100315.

[41] Shaobo Wang, Pan Zhao, Biao Yu, Weixin Huang, and Huawei Liang. 2020. Vehicle trajectory prediction by knowledge-driven LSTM network in urban environments. *J. Advanc. Transport.* 2020, 1 (2020).

[42] Yahan Yang, Ramneet Kaur, Souradeep Dutta, and Insup Lee. 2022. Interpretable detection of distribution shifts in learning enabled cyber-physical systems. In *ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPS'22)*. IEEE, 225–235.

[43] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. 2020. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access* 8 (2020), 58443–58469.

[44] Hang Zhao, Jiyang Gao, Tian Lan, Chen Sun, Ben Sapp, Balakrishnan Varadarajan, Yue Shen, Yi Shen, Yuning Chai, Cordelia Schmid, et al. 2021. TNT: Target-driven trajectory prediction. In *Conference on Robot Learning*. PMLR, 895–904.

Received 28 February 2023; revised 4 January 2024; accepted 4 February 2024