2024 IEEE 63rd Conference on Decision and Control (CDC)
December 16-19, 2024. MiCo, Milan, Italy

Autonomous Driving With Perception Uncertainties: Deep-Ensemble Based Adaptive Cruise Control

Xiao Li*, H. Eric Tseng†, Anouck Girard*, Ilya Kolmanovsky*

Abstract— **Autonomous driving relies on perception systems to understand the environment and guide decision-making. Advanced perception systems often use black-box Deep Neural Networks (DNNs) for human-like comprehension, but their unpredictability and lack of interpretability can limit their use in safety-critical settings. This paper introduces an Ensemble of DNN regressors (Deep Ensemble) to provide predictions with quantified uncertainties. In the context of Adaptive Cruise Control (ACC), the Deep Ensemble estimates the distance headway to the lead vehicle from RGB images, allowing the downstream controller to account for uncertainty. An adaptive cruise controller is developed using Stochastic Model Predictive Control (MPC) with chance constraints to ensure probabilistic safety. Our ACC algorithm is evaluated using a high-fidelity traffic simulator and a real-world traffic dataset, demonstrating effective speed tracking and safe car following. The approach is also tested in out-of-distribution scenarios.**

I. Introduction

Autonomous driving algorithms are typically pipelines of individual modules: the perception module gathers environmental information, and the decision-making module uses this information to make maneuver decisions. Advances in Deep Neural Networks (DNNs) have enabled human-like perception capabilities, effectively extracting environmental information. For example, research has been focusing on integrating DNN-based perception functions, such as localization [1] and mapping [2], into autonomous driving systems. However, a major drawback of DNN-based perception is its lack of interpretability, and the ability of DNNs to generalize can be limited by the coverage of the training data.

Although Neural Networks are universal function approximators [3], they have approximation errors. Their performance becomes even more unpredictable with Out-Of-Distribution (OOD) observations, which follow a different statistical distribution than the training data [4]. This uncertainty in perception can also affect downstream decision-making. In this paper, we consider Adaptive Cruise Control (ACC) leveraging camera sensors. The controller needs to track driver-set speed while maintaining a safe distance from the lead vehicle. In such safety-critical scenarios, accounting for perception uncertainty is crucial for decision-making and control design to ensure system-level safety.

*Xiao Li, Anouck Girard, and Ilya Kolmanovsky are with the Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI 48109, USA. {hsiaoli, anouck, ilya}@umich.edu

†H. Eric Tseng, Retired Senior Technical Leader, Ford Research and Advanced Engineering, Chief Technologist, Excelled Tracer LLC. hongtei.tseng@gmail.com

This research was supported by the University of Michigan / Ford Motor Company Alliance program, and by the National Science Foundation under Awards CMMI-1904394 and ECCS-1931738.

Various methods have been developed to quantify DNN uncertainties. Bayesian Neural Networks [5] have been investigated to represent the uncertainties in DNN predictions via probabilistic modeling of the DNN parameters. To reduce the computational burden, the Monte Carlo Dropout technique has been explored for Bayesian NNs [6]. To improve robustness against adversarial attacks, ensemble methods that use a diverse set of DNNs for a single task have been proposed [7]. Other approaches, such as Laplace Approximation [8], have also been introduced to quantify DNN uncertainties.

In addition, control co-designs have also been studied, under the assumption of bounded DNN errors, to track trajectories [9] and ensure system-level safety [10]. However, these approaches are limited to in-distribution settings [9]. In contrast, this work uses Deep Ensembles [7] for quantifying uncertainty in DNN-based perception, noted for their effectiveness in OOD scenarios. We investigate the downstream control co-design that accounts for perception uncertainty, specifically applied to ACC using camera sensors. The proposed algorithms offer the following properties:

- The Deep Ensemble employs a heterogeneous set of DNNs, that both generate distance headway estimation and quantify the estimation uncertainties, from RGB images of the lead vehicle.
- Leveraging the Deep Ensemble results, the Stochastic MPC is utilized for ACC, guaranteeing probabilistic safety through the integration of chance constraints.
- The proposed ACC algorithm achieves good performance in car-following and speed-tracking tasks, ensuring safety in both in-distribution and OOD scenarios, as verified using a high-fidelity simulator.

This paper is organized as follows: In Sec. II, we introduce the ACC problem. We also outline the assumptions made regarding vehicle kinematics and the ACC design objectives. In Sec. III, we present our Deep Ensemble development that estimates the distance headway which informs the subsequent Stochastic MPC to control the acceleration of the ego vehicle. In Sec. IV, we demonstrate the Deep Ensemble's ability to provide estimations and quantify estimation uncertainties. Furthermore, we validate the proposed adaptive cruise controller, using a high-fidelity simulator and a real-world traffic dataset. Finally, Sec. V provides conclusions.

II. Problem Formulation

In this paper, we focus on control design with visual perception (i.e., cameras) in the loop. As shown in Fig. 1, the ego vehicle observes the lead vehicle in the front via camera sensors, installed to the left and right of the ego vehicle's front window. Using the RGB images from the cameras, the ego vehicle estimates its distance headway to the lead vehicle and, subsequently, commands its acceleration and braking to keep a safe distance headway and track a desired speed.

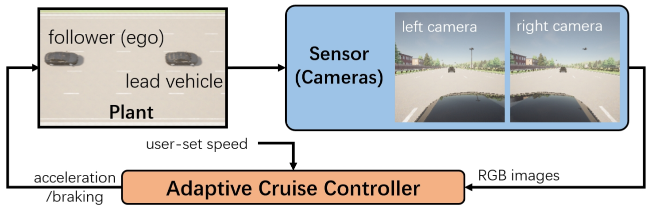

Fig. 1: A schematic diagram of the Adaptive Cruise Control (ACC) scenario: The follower (ego vehicle) maintains a safe distance from the lead vehicle using camera sensors.

We use the following discrete-time model to represent the vehicle kinematics,

$$\begin{aligned} x_{k+1} &= x_k + v_k \Delta t + \frac{1}{2} a_k \Delta t^2, \\ v_{k+1} &= v_k + a_k \Delta t, \end{aligned} \tag{1}$$

where x_k, v_k, and a_k are the longitudinal position, velocity, and acceleration at time instance t_k, respectively; $\Delta t > 0$ is the time in second elapsed between discrete time instances t_k and t_{k+1}. Since we focus on car-following development, we only consider the longitudinal kinematics in Eq. (1) while neglecting the lateral ones. Namely, we assume both ego and lead vehicles follow this dynamics model, do longitudinal acceleration or braking maneuvers, and keep the current lane. In the sequel, we use variables with superscripts $x_k^{(l)}$, $v_k^{(l)}$ to represent the states of the lead vehicle while those without (e.g., x_k, v_k, and a_k) denote the ego vehicle states. We use the following models to represent the camera measurements,

$$I_{k,l} = q_l(x_k, x_k^{(l)}),\ I_{k,r} = q_r(x_k, x_k^{(l)}), \tag{2}$$

where the measurements $I_{k,l}, I_{k,r} \in \mathbb{R}^{3\times 224\times 224}$ are RGB images of 3 color channels and size 224×224 acquired from the left and right cameras, respectively.

In this work, we consider the development of an adaptive cruise controller that adopts the following form

$$a_k = K(I_{k,l}, I_{k,r}, v_s), \tag{3}$$

where v_s is the driver-set ACC speed. The controller K computes the acceleration/deceleration command for the ego vehicle based on a pair of RGB images while incorporating the following control objectives and constraints:

- safety: keep an adequate distance headway d_k, defined as the vehicle bumper-to-bumper distance, to the lead vehicle to prevent potential collisions.
- fuel economy: minimize the accumulated acceleration effort $\sum_{i=0}^{N-1} |a_{k+i}|$ over a horizon of length N.
- driving comfort: minimize the rate of change in the acceleration trajectory $(a_{k+i})_{i=0}^{N-1}$.
- speed tracking: track the driver-set speed v_s.
- speed and acceleration limits: the speed v_k and the acceleration a_k within the interval $[v_{\min}, v_{\max}]$ and $[a_{\min}, a_{\max}]$, respectively.

This problem is challenging due to the high dimensionality of the image space, which can induce unpredictable behavior of the controller and, subsequently, raise safety concerns.

III. Method

We propose a modularized ACC development approach to enhance the safety guarantees. As shown in Fig. 2b, a Deep Ensemble estimates the distance headway d_k as a Gaussian distribution and the variance quantifies the estimation uncertainty in Sec. III-A. Subsequently, a Stochastic MPC is utilized to optimize the acceleration trajectory to realize the aforementioned design objectives while ensuring probabilistic safety in Sec. III-B.

A. Deep Neural Network Ensemble

We implement a Deep Ensemble to estimate the distance headway d_k ($d_k \in \mathbb{R}$, $d_k \geq 0$) to the lead vehicle given a pair of RGB images $I_{k,l}$, $I_{k,r}$ from on-board cameras. In this regression problem, the estimates admit the following form,

$$d_k = p(I_{k,l}, I_{k,r}) + e(I_{k,l}, I_{k,r}), \tag{4}$$

where $p(I_{k,l}, I_{k,r})$ is the distance headway estimate and $e(I_{k,l}, I_{k,r})$ is the estimation error that depends on the current image observations. Typical approaches in the literature focus on learning an accurate mapping $p(I_{k,l}, I_{k,r})$ that minimizes $|d_k - p(I_{k,l}, I_{k,r})|$ without characterizing the behavior of the error $e(I_{k,l}, I_{k,r})$. The high dimensionality of image space necessitates complex Convolutional Neural Networks (CNNs) as image encoders, resulting in more unpredictable error dynamics $e(I_{k,l}, I_{k,r})$. Therefore, our work focuses on further modeling and characterizing this error.

Inspired by [11], we assume the error $e(I_{k,l}, I_{k,r})$ is zero-mean Gaussian. We develop DNNs that can simultaneously generate an estimate of $p(I_{k,l}, I_{k,r})$ and quantify the estimation uncertainties by predicting the variance of the error $e(I_{k,l}, I_{k,r})$. As shown in Fig. 2a, the individual ith DNN comprises two blocks: the CNN backbone takes two images $I_{k,l}, I_{k,r}$ and embeds them into vectors $z_{k,l}, z_{k,r} \in \mathbb{R}^{1280}$ using two identical CNN image encoders with shared parameters (i.e., $z_{k,l} = f_{\text{cnn}}(I_{k,l}|\Theta_i)$, $z_{k,r} = f_{\text{cnn}}(I_{k,r}|\Theta_i)$, and Θ_i is the shared parameters in the CNN image encoders); the subsequent Multi-Layer Perceptron (MLP) computes two outputs p_i, σ_i^2 from input vectors $z_{k,l},\ z_{k,r}$ according to,

$$\begin{aligned} z_0 = [z_{k,l}^T \quad z_{k,r}^T]^T, \quad z_1 = \sigma_{\text{ReLU}}\left(W_{i,1} z_0 + b_{i,1}\right), \\ z_2 = W_{i,2} z_1 + b_{i,2}, \quad [p_i \quad \sigma_i^2]^T = z_2, \end{aligned} \tag{5}$$

where $W_{i,1} \in \mathbb{R}^{2560\times 512}, W_{i,2} \in \mathbb{R}^{512\times 2}$ and $b_{i,1} \in \mathbb{R}^{512}, b_{i,2} \in \mathbb{R}^2$ are the network parameters; $\sigma_{\text{ReLU}}(z) = \max\{0, z\}$ is an element-wise ReLU activation function. The ith DNN produces estimate $p_i(I_{k,l}, I_{k,r})$ of the actual distance headway d_k. It also computes an estimate of the variance $\sigma_i^2(I_{k,l}, I_{k,r})$ of the error $e(I_{k,l}, I_{k,r})$.

Given a training trajectory $\mathcal{D} = (d_k, I_{k,l}, I_{k,r})_{k=1}^M$, the parameter Θ of the ith DNN, i.e., $\Theta =$

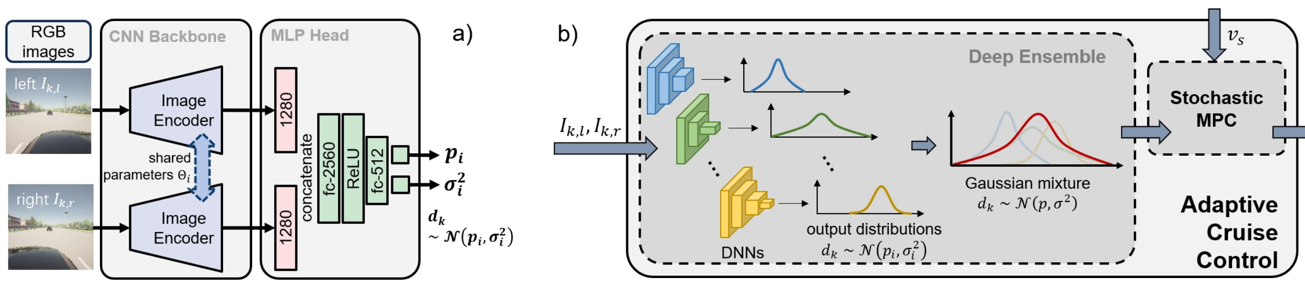

Fig. 2: Schematic diagrams of adaptive cruise controller design: (a) Each DNN, differing in the CNN architecture of the image encoder, estimates the distribution of the distance headway from input RGB images. (b) An ensemble of DNNs, with heterogeneous CNN architectures, collectively estimates the distance headway as a Gaussian mixture. A Stochastic MPC then uses the estimated headway mean and variance to compute the acceleration/deceleration command for the ego vehicle.

$\{\Theta_i, W_{i,1}, W_{i,2}, b_{i,1}, b_{i,2}\}$, is optimized using the following proposition:

Proposition 1. *Given a training trajectory $\mathcal{D} = (d_k, I_{k,l}, I_{k,r})_{k=1}^M$, assuming each data point $(d_k, I_{k,l}, I_{k,r}) \in D$ is independently collected, and the error is zero-mean Gaussian, i.e., $e(I_{k,l}, I_{k,r}) \sim \mathcal{N}\left(0, \sigma_i^2(I_{k,l}, I_{k,r})\right)$, the optimal parameter is attained according to the following likelihood maximization,*

$$\Theta^* = \underset{\Theta}{\operatorname{argmax}}\, \mathbb{P}\left(\mathcal{D}|\Theta\right), \tag{6}$$

and it is equivalent to the following optimization,

$$\begin{aligned}\Theta^* = \underset{\Theta}{\operatorname{argmin}}\, \mathcal{L}\left(\mathcal{D}|\Theta\right) = \underset{\Theta}{\operatorname{argmin}} \textstyle\sum_{k=1}^M \\ \left[\log \sigma_i^2(I_{k,l}, I_{k,r}|\Theta) + \frac{(d_k - p_i(I_{k,l}, I_{k,r}|\Theta))^2}{\sigma_i^2(I_{k,l}, I_{k,r}|\Theta)}\right].\end{aligned} \tag{7}$$

In the case of a large dataset $\mathcal{D}$, we note that an iterative training algorithm based on Monte Carlo Sampling can be applied, i.e., batch Stochastic Gradient Descent (SGD), where a mini-batch dataset $\mathcal{D}' \subset \mathcal{D}$ is sampled to update the parameter Θ^* according to Eq. (7) at each iteration. The proof is presented as follows:

Proof. The likelihood in (7) can be rewritten according to

$$\begin{aligned}&\mathbb{P}\left(\mathcal{D}|\Theta\right) = \textstyle\prod_{k=1}^M \mathbb{P}\left(d_k | I_{k,l}, I_{k,r}, \Theta\right) \\ &= \textstyle\prod_{k=1}^M \frac{1}{\sigma_i(I_{k,l}, I_{k,r}|\Theta)\sqrt{2\pi}} \exp -\frac{1}{2}\left(\frac{d_k - p_i(I_{k,l}, I_{k,r}|\Theta)}{\sigma_i(I_{k,l}, I_{k,r}|\Theta)}\right)^2\end{aligned}$$

where the first equality is derived from the independence assumption, and the second equality is due to the zero mean Gaussian assumption of $e(I_{k,l}, I_{k,r})$. Then, the maximization in Eq. (6) is equivalent to the following minimization, $\underset{\Theta}{\operatorname{argmin}}\left(-\log \mathbb{P}\left(\mathcal{D}|\Theta\right)\right)$, where this minimization of the negative log-likelihood is equivalent to that in Eq. (7). □

Furthermore, we adopt the idea of Deep Ensemble [7] to improve the robustness of the distance headway estimation in OOD scenarios. The Deep Ensemble comprises n different DNNs of various CNN architectures as the image encoders (see Fig. 2). Individually, the ith DNN in the Deep Ensemble is trained to generate predictions $p_i(I_{k,l}, I_{k,r})$, $\sigma_i^2(I_{k,l}, I_{k,r})$, where the actual distance headway d_k follows a Gaussian distribution $\mathcal{N}\big(p_i(I_{k,l}, I_{k,r}), \sigma_i^2(I_{k,l}, I_{k,r})\big)$ according to assumptions in Proposition 1. Collectively, n DNNs in the Deep Ensemble form a Gaussian mixture, and produce the final distance headway estimates according to,

$$\begin{aligned}p_k &= \frac{1}{n}\sum_{i=1}^n p_i(I_{k,l}, I_{k,r}), \\ \sigma_k^2 &= \frac{1}{n}\sum_{i=1}^n \left(\sigma_i^2(I_{k,l}, I_{k,r}) + p_i^2(I_{k,l}, I_{k,r})\right) - p_k^2,\end{aligned} \tag{8}$$

where here and in the sequel we drop the dependence of p_k, σ_k^2 on $(I_{k,l}, I_{k,r})$ to simplify the notations. Eventually, the actual distance headway follows a Gaussian distribution derived from the Gaussian mixture, i.e., $d_k \sim \mathcal{N}(p_k, \sigma_k^2)$.

B. Adaptive Cruise Control

At current time t_k, we assume that previous acceleration a_{k-1} and the distance headway estimates $p_{k-1}, \sigma_{k-1}^2, p_k, \sigma_k^2$ generated from the Deep Ensemble are known. Note that the actual distance headway, i.e., d_{k-1} and d_k, is unknown to the algorithm, but the following results hold, $d_k \sim \mathcal{N}(p_k, \sigma_k^2)$, $d_{k-1} \sim \mathcal{N}(p_{k-1}, \sigma_{k-1}^2)$. Then, we can predict the distributions of future distance headway for a variable acceleration trajectory using the following proposition:

Proposition 2. *Given a_{k-1}, an acceleration trajectory $(a_{k+i})_{i=0}^{N-1}$ of length N, and distribution parameters p_{k-1}, σ_{k-1}^2, p_k, σ_k^2, such that the unknown distance headway obeys $d_k \sim \mathcal{N}(p_k, \sigma_k^2)$, $d_{k-1} \sim \mathcal{N}(p_{k-1}, \sigma_{k-1}^2)$, and if the lead vehicle has a constant speed, then, the variables d_{k+i}, Δv_{k+i}, $i = 0, \ldots, N$ are Gaussian distributed,*

$$\begin{aligned}d_{k+i} &\sim \mathcal{N}(p_{k+i}, \sigma_{k+i}^2), \\ \Delta v_{k+i} &\sim \mathcal{N}(p'_{k+i}, \sigma'^2_{k+i}), \quad i = 0, \ldots, N,\end{aligned} \tag{9}$$

where $\Delta v_{k+i} = v_{k+i}^{(l)} - v_{k+i}$ is the speed difference between the lead and ego vehicles. Furthermore, the distribution parameters p_{k+i}, σ_{k+i}^2 and $p'_{k+i}, \sigma'^2_{k+i}$ can be recursively derived using the following results,

$$\begin{aligned}p'_k = \tfrac{1}{\Delta t}(p_k - p_{k-1}) - \tfrac{1}{2}a_{k-1}\Delta t,\ \sigma'^2_k = \tfrac{1}{\Delta t^2}(\sigma_k^2 + \sigma_{k-1}^2), \\ p_{k+i+1} = p_{k+i} + p'_{k+i}\Delta t - \tfrac{1}{2}a_{k+i}\Delta t^2, \\ \sigma_{k+i+1}^2 = \sigma_{k+i}^2 + \Delta t^2 \sigma'^2_{k+i},\ p'_{k+i+1} = p'_{k+i} - a_{k+i}\Delta t, \\ \sigma'^2_{k+i+1} = \tfrac{2}{\Delta t^2}\sigma_{k+i}^2 + \sigma'^2_{k+i},\ i = 0, \ldots, N-1,\end{aligned}$$

where the distribution means p_{k+i}, p'_{k+i} linearly depend on the variables $(a_{k+i})_{i=0}^{N-1}$, and variances $\sigma^2_{k+i}, \sigma'^2_{k+i}$ are constants, for all $i = 0, \dots, N$.

Proof. Assuming the lead vehicle maintains a constant speed, the proposition above can be derived from the following equalities, $\Delta v_k = \frac{1}{\Delta t}(d_k - d_{k-1}) - \frac{1}{2}a_{k-1}\Delta t$, $d_{k+i+1} = d_{k+i} + \Delta v_{k+i}\Delta t - \frac{1}{2}a_{k+i}\Delta t^2$, $\Delta v_{k+i+1} = \frac{1}{\Delta t}(d_{k+i+1} - d_{k+i}) - \frac{1}{2}a_{k+i}\Delta t$, $i = 0, \dots, N-1$. Based on Propostion 2, we establish the prediction of future distance headway d_{k+i} and speed difference Δv_{k+i} as Gaussian distributions with the means being the linear functions of the acceleration trajectory $(a_{k+i})_{i=0}^{N-1}$ and constant variances. □

Hence, treating $(a_{k+i})_{i=0}^{N-1}$ as decision variables, we formulate a Stochastic MPC problem that predicts the distributions of the future distance headway and speed difference for different $(a_{k+i})_{i=0}^{N-1}$ and optimizes $(a_{k+i})_{i=0}^{N-1}$ while incorporating the objectives in Sec. II according to,

$$\underset{\substack{a_{k+i-1}, v_{k+i}, p_{k+i},\\ p'_{k+i},\ i=1,\dots,N}}{\operatorname{argmin}} \mathbb{E}\left[\sum_{i=0}^{N-1} r_1 a^2_{k+i} + r_2(a_{k+i} - a_{k+i-1})^2 + \sum_{i=1}^{N} q_1(v_{k+i} - v_s)^2 + q_2 \Delta v^2_{k+i}\right] \quad (10a)$$

subject to:

$$\mathbb{P}\left(d_{k+i} \geq d_s + T_s v_{k+i}\right) \geq 1 - \epsilon_i, \quad (10b)$$
$$d_{k+i} \sim \mathcal{N}(p_{k+i}, \sigma^2_{k+i}),\ \Delta v_{k+i} \sim \mathcal{N}(p'_{k+i}, \sigma'^2_{k+i}), \quad (10c)$$
$$p_{k+i} = p_{k+i-1} + p'_{k+i-1}\Delta t - \frac{1}{2}a_{k+i-1}\Delta t^2, \quad (10d)$$
$$p'_{k+i} = p'_{k+i-1} - a_{k+i-1}\Delta t, \quad (10e)$$
$$v_{\min} \leq v_{k+i} \leq v_{\max},\ a_{\min} \leq a_{k+i-1} \leq a_{\max}, \quad (10f)$$
$$v_{k+i} = v_{k+i-1} + a_{k+i-1}\Delta t,\ i = 1, \dots, N, \quad (10g)$$

where N is the prediction horizon; $\epsilon_i \in (0, 1]$, $i = 1, \dots, N$ are tunable positive constants; v_s is the driver-set target speed; d_s, T_s are the adjustable ACC stopping distance, and constant time headway, respectively. Meanwhile, Proposition 2 implies larger variances σ^2_{k+i}, σ'^2_{k+i} for prediction horizon i further in the future. We set the constants ϵ_i to satisfy the following inequality, $\epsilon_1 \leq \cdots \leq \epsilon_N$, such that the chance constraints (10b) are relaxed more further along the horizon. The variables r_1, r_2, q_1, q_2 are tunable weights that balance the minimization of control effort, the reduction of the rate of changes in control, speed tracking, and lead vehicle-following, respectively. Furthermore, we use the chance constraints in Eq. (10b), (10c) to enforce a sufficient distance headway in probability, and the equalities (10d), (10e) propagate the distribution means based on the results from Proposition 2.

Note that the Gaussian-distributed random variables d_{k+i}, Δv_{k+i} and chance constraints render the MPC problem (10) stochastic. To make the problem machine solvable, we transcript the Stochastic MPC into a deterministic one using the following result:

Proposition 3. *Under the condition that $\delta_i = 0$ for all $i = 1, \dots, N$, solving the following **Quadratic Programming** problem recovers the solution of the Stochastic MPC problem (10),*

$$\underset{\substack{a_{k+i-1}, v_{k+i}, p_{k+i},\\ \delta_i, p'_{k+i},\ i=1,\dots,N}}{\operatorname{argmin}} \sum_{i=0}^{N-1} r_1 a^2_{k+i} + r_2(a_{k+i} - a_{k+i-1})^2 + \sum_{i=1}^{N} q_1(v_{k+i} - v_s)^2 + q_2 p'^2_{k+i} + \rho\delta_i \quad (11a)$$

subject to:

$$p_{k+i} \geq d_s + T_s v_{k+i} + \sqrt{2\sigma^2_{k+i}}\,\texttt{erf}^{-1}(1 - 2\epsilon_i) - \delta_i \quad (11b)$$
$$(10d),\ (10e),\ (10f),\ (10g),\ \delta_i \geq 0,\ i = 1, \dots, N, \quad (11c)$$

where $\texttt{erf}^{-1}$ is the inverse image of the Gauss error function, non-negative variables δ_i, $i = 1, \dots, N$ are used to relax the constraints (11b) ensuring recursive feasibility and the weight ρ is adjusted to penalize the violations of soft constraints (11b) due to the introduction of δ_i.

Proof. We consider the condition that $\delta_i = 0$ for all $i = 1, \dots, N$. The only random variables in Eq. (10a) are $\Delta v_{k+i} \sim \mathcal{N}(p'_{k+i}, \sigma'^2_{k+i})$, therefore, the other terms can be moved out of the expectation. Moreover, we can establish the following equality, $\mathbb{E}\left[\Delta v^2_{k+i}\right] = \sigma'^2_{k+i} + \mathbb{E}\left[\Delta v_{k+i}\right]^2 = \sigma'^2_{k+i} + p'^2_{k+i}$, which combined with σ'^2_{k+i} being a constant from the results in Proposition 2 proves that $\operatorname{argmin} \sum_{i=1}^{N} \mathbb{E}\left[q_2 \Delta v^2_{k+i}\right] = \operatorname{argmin} \sum_{i=1}^{N} q_2 p'^2_{k+i}$. Namely, the optimization objectives of (10) and (11) are equivalent. Furthermore, the chance constraints (10b), (10c) are equivalent to Eq. (11b). □

Eventually, provided with the estimated distributions from Deep Ensemble, we formulate a Deterministic MPC that is recursively feasible, predicts the future distance headway distributions, and computes an acceleration trajectory $(a_{k+i})_{i=0}^{N-1}$ ensuring probabilistic safety. We also note that the introduction of slack variables δ_i ensures the recursive feasibility of the optimization problem (11) by relaxing the probabilistic safety guarantee in (10b). A more detailed discussed is available in [12].

IV. Case Studies

Here, we demonstrate the effectiveness of the proposed ACC algorithm. The Deep Ensemble is trained and evaluated using a high-fidelity simulation environment in Sec. IV-A. We showcase the proposed ACC algorithm that integrates the Deep Ensemble with the Stochastic MPC in a simulation example in Sec. IV-B, and report the quantitative results in Sec. IV-C leveraging a high-fidelity simulator and real-world vehicle trajectories. Finally, in comparison with the in-distribution example provided in Sec. IV-B, the performance of the proposed algorithm is demonstrated in OOD scenarios in Sec. IV-D. The demonstration videos are available in `https://xiaolisean.github.io/publication/2024-12-16-CDC2024`.

A. Deep Ensemble for Distance Headway Estimation

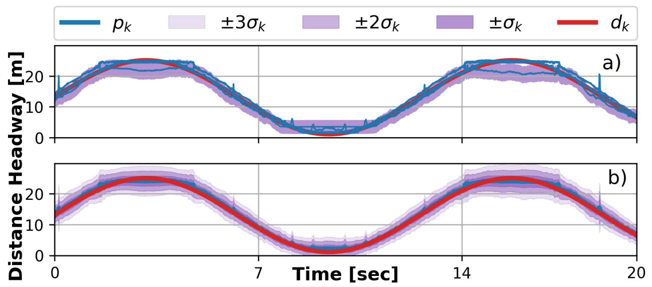

Fig. 3: Distance headway estimates p_k (blue lines) with uncertainty quantification using 1σ, 2σ, and 3σ intervals (purple bands) versus the actual d_k (red lines). (a) The results, $p_i(I_{k,l}, I_{k,r}), \sigma_i^2(I_{k,l}, I_{k,r})$, $i = 1, \ldots, 6$, of each DNN visualized in overlap. (b) Deep Ensemble estimation.

In the Deep Ensemble, we integrate 6 DNNs, i.e., $n = 6$, with each of them employing a different CNN architecture as the image encoder. We utilize the following CNNs due to their outstanding performance as image encoders in solving image classification problems: ResNet50 [13], GoogleNet [14], AlexNet [15], MobileNetV2 [16], EfficientNet [17], and VGG16 [18]. We also note that other network architectures can be employed, provided the architectures within the Deep Ensemble are diverse. The goal is to train individual DNNs to predict the distribution parameters $p_i(I_{k,l}, I_{k,r})$, $\sigma_i^2(I_{k,l}, I_{k,r})$ given the corresponding RGB images $I_{k,l}, I_{k,r}$, such that $d_k \sim \mathcal{N}\big(p_i(I_{k,l}, I_{k,r}), \sigma_i^2(I_{k,l}, I_{k,r})\big)$.

We use the Carla simulator [19] to collect datasets and test our developments. We collect a dataset $D = (d_k, I_{k,l}, I_{k,r})_{k=1}^{20706}$ of 20706 data triplets. The data points are collected in the map `Town06` in Carla. To simplify the exposition of the approach, we fix the model of the lead vehicle to `vehicle.lincoln.mkz_2020` (2020 Lincoln MKZ Sedan) and set the weather to `ClearNoon` (good lighting conditions, no rain, and no objects casting shadow). The distance headway in the dataset ranges from 1 to 25 m, where data points with $d_k \geq 25$ m are neglected due to cameras' resolution limitations. We note that scenarios where $d_k \geq 25$ m do not affect the ACC algorithm's performance in ensuring safety in the near term.

We use Python with Pytorch [20] to train the DNN. The training process is set to 100 epochs using Batch SGD with momentum and the loss function (7) defined among the mini-batch dataset. The mini-batch sizes are set to 60, 105, 500, 65, 60, and 75 (maximum batch size capability of a Nvidia GeForce RTX 4080 GPU with 16 GB memory) for ResNet50, GoogleNet, AlexNet, MobileNetV2, EfficientNet, and VGG16, respectively. We set the learning rate and momentum to 0.001 and 0.9, respectively. Each time we train a different DNN, we randomly select 80% data points for training and 20% data points for validation, while the DNN initial parameters Θ are also randomly initialized. All images input to the DNN are normalized using the following function in the `torchvision` package,

```
transforms.Normalize
```

$$([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]).$$

Meanwhile, to ensure numerical stability with the logarithm in the loss function (7), we enforce the positiveness of the output variance $\sigma_i^2(I_{k,l}, I_{k,r})$ without significantly altering its value, using the following assignment,

$$\sigma_i^2(I_{k,l}, I_{k,r}) \leftarrow \epsilon + \log\left(1 + \exp \sigma_i^2(I_{k,l}, I_{k,r})\right),$$

where a small $\epsilon > 0$ is chosen, e.g., $\epsilon = 10^{-6}$.

To evaluate the performance of each DNN and the Deep Ensemble, we separately collect a testing trajectory $(d_k, I_{k,l}, I_{k,r})_{k=1}^{M'}$ of 20 seconds, and the results are reported in Fig. 3. The 1σ band in the Deep Ensemble results, i.e., $p_k \pm \sigma_k$, contains the actual distance headway d_k which demonstrates the effectiveness of our method in both providing accurate estimates and quantifying the estimation uncertainties. Moreover, we also note that results from individual DNNs differ from each other significantly when the distance headway is large. This is due to the resolution limitation of the RGB images, where the lead vehicle vanishes as a black pixel when the distance headway is larger than 20 m. However, the Deep Ensemble can reflect this uncertainty using a larger variance in the estimation results.

B. Adaptive Cruise Control in Carla Simulation

Using the Carla simulator [19], we create car-following scenarios where the ego vehicle, controlled by our algorithm, follows a lead vehicle. In the sequel, we leverage a naturalistic traffic trajectory dataset, named High-D dataset [21], to configure realistic car-following scenarios. The High-D dataset records real-world vehicle trajectories in German freeways. The statistics visualized in Fig. 4 are obtained from data of 110,500 vehicles driven over 44,500 kilometers.

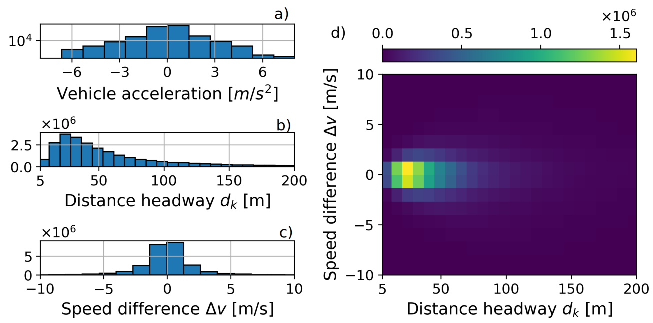

Fig. 4: Histogram of vehicle driving statistics in High-D dataset: (a) longitudinal acceleration/deceleration (y-axis in log scale); (b) distance headway and (c) speed difference between lead and follower vehicles; (d) 2D histogram combining statistics in (b) and (c).

The lower speed limit is set to $v_{\min} = 0$ m/s while the upper speed limit of the dataset is $v_{\max} = 34$ m/s. As shown in Fig. 4, the majority of longitudinal accelerations and decelerations of High-D vehicles are within the range of $[a_{\min}, a_{\max}] = [-6, 6]$ m/s^2. The Stochastic MPC has a prediction horizon of 3 sec, i.e., $N = 3$ and $\Delta t = 1$ sec. Furthermore, the Stochastic MPC operates in an asynchronous updating scheme and recomputes a_k every 0.5 sec. Other parameters are set using the following values: $d_s = 15$ m, $T_s = 0$ sec, $[r_1, r_2, q_1, q_2, \rho] = [1, 5, 5, 1, 50]$, and $\epsilon_{1,2,3} = 0.2,\ 0.4,\ 0.6$, respectively. Provided with distributions $d_{k+i} \sim \mathcal{N}(p_{k+i}, \sigma^2_{k+i})$, $\Delta v_{k+i} \sim \mathcal{N}(p'_{k+i}, \sigma'^2_{k+i})$, from the Deep Ensemble, the MPC problem (11) is solved using PyDrake [22].

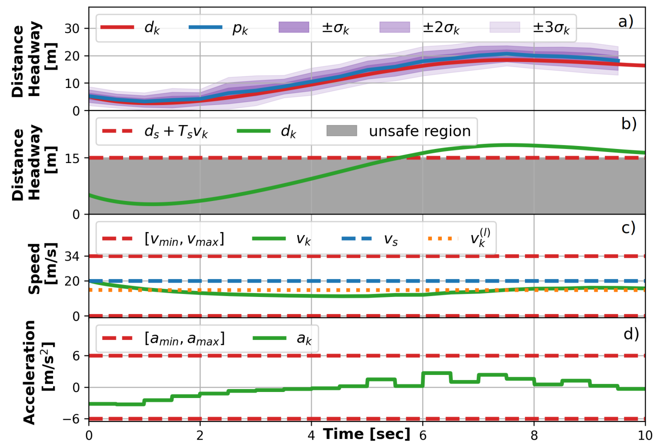

Fig. 5: ACC simulation example in which the ego vehicle follows the lead vehicle given its speed smaller than $v_s = 20$ m/s: (a) distance headway estimation from the Deep Ensemble; (b) ACC algorithm regulating the ego out of the unsafe region; (c) speed trajectories of the lead and ego vehicles; (d) acceleration commands.

When the lead vehicle's speed is below $v_s = 20$ m/s, our algorithm performs car-following as both promoted by the term p'^2_{k+i} in (11a) and enforced by the constraints in (11b). As shown in Fig. 5b, our ACC algorithm effectively decelerates the ego vehicle to maintain a safe distance headway. Subsequently, the introduction of the term $(v_{k+i} - v_s)^2$ in (11a) prompts the algorithm to command the ego vehicle to accelerate and track the driver-set speed v_s. The control effort remains moderate, and the rate of change in acceleration is small due to the inclusion of terms a^2_{k+i} and $(a_{k+i} - a_{k+i-1})^2$ in (11a) (see Fig. 5c). More animation examples are available in https://xiaolisean.github.io/publication/2024-12-16-CDC2024.

C. Adaptive Cruise Control with Real-World Trajectories

We inherit parameters and ACC configurations from the previous section and use the High-D dataset [21] to quantitatively evaluate the performance of our algorithm. We construct 56 car-following test cases using the Carla simulator [19]. In those cases, the speed trajectories $(v_k)^K_{k=1}$ of the lead vehicles are sampled from the High-D dataset (see Fig. 6) with standard deviations larger than 4 m/s. We remove speed trajectories where the lead vehicles take fewer acceleration/braking actions to be able to test our algorithm in more challenging but realistic cases. In the sequel, we use $(v_k)_k$ to denote $(v_k)^K_{k=1}$ given the length of the speed trajectory K is variable.

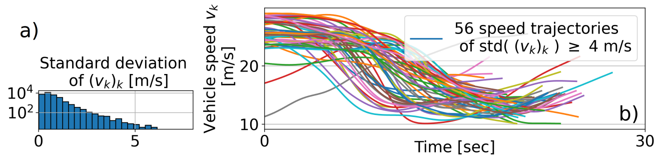

Fig. 6: Sampling of lead vehicles' speed trajectories from the High-D dataset: (a) histogram of the standard deviation $\texttt{std}\big((v_k)_k\big)$ of the High-D vehicle speed trajectory $(v_k)_k$ (y-axis in log scale); (b) 56 speed trajectories $(v_k)_k$ are chosen with their standard deviation larger than 4 m/s.

Moreover, as shown in Fig. 4, the majority of the follower vehicles in the High-D dataset have a distance headway larger than 5 m and a relative speed difference within an interval of $[-5, 5]$ m/s. Hence, we initialize the follower vehicle with an initial distance headway $d_k = 5$ m and an initial speed 5 m/s larger than the lead vehicle, i.e., $\Delta v_k = -5$ m/s. These initial conditions yield the initial distance headway which is unsafe; this allows us to examine the ability of the algorithm to handle emergency conditions. Finally, in each test case, the ACC target speed v_s is set to be the average speed of the sampled lead vehicle's speed trajectory $(v_k)_k$. Subsequently, the speed of the lead vehicle fluctuates near v_s, and we can test both the speed-tracking and car-following functionalities in one test case.

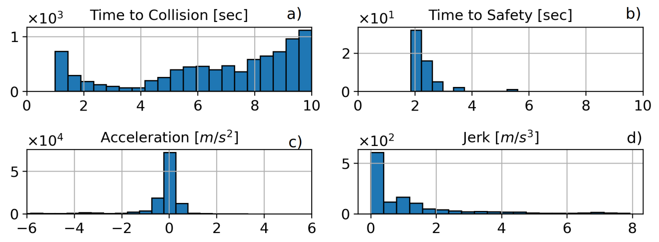

Fig. 7: Statistics from the 56 test cases where each data point corresponds to a frame in the simulation where the frame rate is 100 Hz: (a) Time to Collision (ToC) is calculated as the time required for the follower and lead vehicles to collide assuming they travel at the current speed. (Infinite ToC values when the lead vehicle is faster than the follower are neglected) (b) Time to Safety is defined as the time elapsed from $t = 0$ to the time instance t_k when $d_k \geq d_s + T_s v_k$. (c) Acceleration commands from the ACC algorithm. (d) Jerk.

As shown in Fig. 7, our algorithm can ensure a sufficient Time-to-Collsion (ToC) that is larger than 4 seconds at most of the simulation time. We also note that the majority of the time when the ToC ≤ 2 sec is due to the test cases being initialized with a small distance headway and large velocity difference. Meanwhile, our algorithm can also regulate the ego vehicle back to a safe distance headway within 4 seconds (see Fig. 7b) while the acceleration/deceleration effort is moderate and the jerk values are kept smaller than 2 $\mathrm{m/s^3}$ for a comfortable driving experience.

D. ACC in Out-Of-Distribution Scenarios

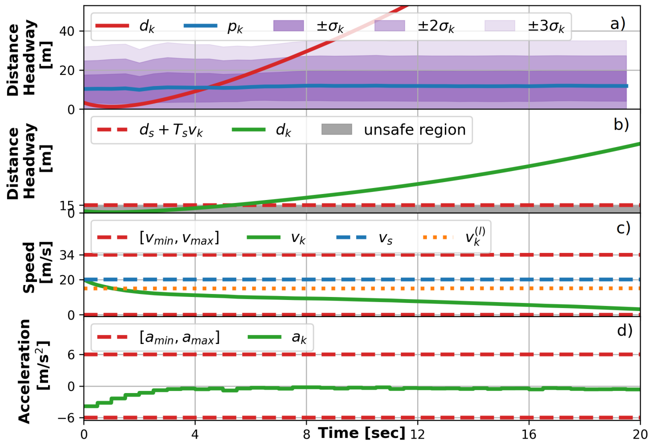

Fig. 8: ACC example in an OOD scenario.

We note that the previous case studies are conducted using the same weather settings and the same lead vehicle model as in the training dataset (see Sec. IV-A). To further explore the capability of the algorithm in OOD scenarios, we change the lead vehicle model from a small 2020 Lincoln MKZ sedan (in black) to a large firetruck (in red). Moreover, we also change the weather from `ClearNoon` to `HardRainSunset` where the lighting condition is worse, objects cast shadows on the road and raindrops block the camera views. The animation is available in `https://xiaolisean.github.io/publication/2024-12-16-CDC2024`. To compare with the in-distribution scenario, we use the same initial distance headway, initial speed difference, and speed profile of the lead vehicle as in the example presented in Fig. 5. The results are reported in Fig. 8. We note that the Deep Ensemble can capture the out-of-distribution and yield predictions with large variance. Then, Stochastic MPC commands the vehicle to take conservative maneuvers and decelerate to a speed lower than the set speed v_s.

V. Conclusion

In this paper, we introduced a Deep Ensemble-based distance headway estimator using RGB images of the lead vehicle. This estimator provides both mean and variance of the headway distance. A Stochastic MPC based controller is then designed to enable adaptive cruise control with probabilistic safety. Using a high-fidelity simulator and real-world traffic dataset, we demonstrated the effectiveness of our proposed approach in speed tracking and car following, ensuring safety in both in-distribution and out-of-distribution scenarios. Future investigations will encompass experiments on a wider range of OOD scenarios and extensions to real-world applications.

References

[1] W. Lu, Y. Zhou, G. Wan, S. Hou, and S. Song, “L3-net: Towards learning based lidar localization for autonomous driving,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6389–6398.

[2] T. Roddick and R. Cipolla, “Predicting semantic map representations from images using pyramid occupancy networks,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 138–11 147.

[3] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[5] R. M. Neal, *Bayesian learning for neural networks*. Springer Science & Business Media, 2012, vol. 118.

[6] Y. Gal and Z. Ghahramani, “Dropout as a bayesian approximation: Representing model uncertainty in deep learning,” in *international conference on machine learning*. PMLR, 2016, pp. 1050–1059.

[7] B. Lakshminarayanan, A. Pritzel, and C. Blundell, “Simple and scalable predictive uncertainty estimation using deep ensembles,” *Advances in neural information processing systems*, vol. 30, 2017.

[8] H. Ritter, A. Botev, and D. Barber, “A scalable laplace approximation for neural networks,” in *6th international conference on learning representations, ICLR 2018-conference track proceedings*, vol. 6. International Conference on Representation Learning, 2018.

[9] S. Dean, N. Matni, B. Recht, and V. Ye, “Robust guarantees for perception-based control,” in *Learning for Dynamics and Control*. PMLR, 2020, pp. 350–360.

[10] X. Li, Y. Li, A. Girard, and I. Kolmanovsky, “System-level safety guard: Safe tracking control through uncertain neural network dynamics models,” *arXiv preprint arXiv:2312.06810*, 2023.

[11] D. A. Nix and A. S. Weigend, “Estimating the mean and variance of the target probability distribution,” in *Proceedings of 1994 ieee international conference on neural networks (ICNN’94)*, vol. 1. IEEE, 1994, pp. 55–60.

[12] B. Kouvaritakis, M. Cannon, S. V. Raković, and Q. Cheng, “Explicit use of probabilistic distributions in linear predictive control,” *Automatica*, vol. 46, no. 10, pp. 1719–1724, 2010.

[13] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[14] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Advances in neural information processing systems*, vol. 25, 2012.

[16] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[17] M. Tan and Q. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.

[18] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[19] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “CARLA: An open urban driving simulator,” in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[20] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, “Pytorch: An imperative style, high-performance deep learning library,” *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[21] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein, “The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems,” in *International Conference on Intelligent Transportation Systems*. IEEE, 2018, pp. 2118–2125.

[22] R. Tedrake and the Drake Development Team, “Drake: Model-based design and verification for robotics,” 2019. [Online]. Available: https://drake.mit.edu