information

MDPI

Article

Uncertainty-Driven Data Aggregation for Imitation Learning in Autonomous Vehicles

Changquan Wang [1,2] and Yun Wang [1,2,*]

[1] Institute of Microelectronics of the Chinese Academy of Sciences, Beijing 100029, China; wangchangquan@ime.ac.cn
[2] University of Chinese Academy of Sciences, Beijing 101408, China
* Correspondence: wangyun@ime.ac.cn

Abstract: Imitation learning has shown promise for autonomous driving, but suffers from covariate shift, where the policy performs poorly in unseen environments. DAgger is a popular approach that addresses this by leveraging expert demonstrations. However, DAgger's frequent visits to sub-optimal states can lead to several challenges. This paper proposes a novel DAgger framework that integrates Bayesian uncertainty estimation via mean field variational inference (MFVI) to address this issue. MFVI provides better-calibrated uncertainty estimates compared to prior methods. During training, the framework identifies both uncertain and critical states, querying the expert only for these states. This targeted data collection reduces the burden on the expert and improves data efficiency. Evaluations on the CARLA simulator demonstrate that our approach outperforms existing methods, highlighting the effectiveness of Bayesian uncertainty estimation and targeted data aggregation for imitation learning in autonomous driving.

Keywords: DAgger; Bayesian uncertainty estimation; Bayesian network; autonomous driving; critical scenes

Citation: Wang, C.; Wang, Y. Uncertainty-Driven Data Aggregation for Imitation Learning in Autonomous Vehicles. *Information* **2024**, *15*, 336. https://doi.org/10.3390/info15060336

Academic Editors: Antonio Comi, Jianbo Li and Junjie Pang

Received: 20 April 2024
Revised: 27 May 2024
Accepted: 3 June 2024
Published: 6 June 2024

1. Introduction

In recent years, end-to-end imitation learning has demonstrated significant potential for autonomous driving by learning a driving policy that directly maps sensory observations to control commands [1–5]. However, a primary challenge in imitation learning is the issue of covariate shift [6,7], which refers to the variation in the state distribution encountered by the policy. As shown in Figure 1a, policies quickly accumulate errors, leading to poor performance in new environments, a phenomenon known as the compounding error problem. Therefore, researchers have proposed data aggregation techniques for training robust policies. One such technique is DAgger [8], which iteratively improves a policy by combining the learner's experience with expert corrections. It is compared with imitation learning, as shown in Figure 1a. This approach has demonstrated its effectiveness in various robotic tasks. Several DAgger variants have been proposed: Q-DAgger [9] accelerates the learning process by incorporating a Q-function to estimate the value of each state–action pair, guiding the algorithm to focus on more informative samples. AggreVaTe [10] introduces a reinforcement learning approach that learns a value function to minimize the cost incurred by the expert, reducing the burden on the expert during training. AggreVaTeD [11] extends AggreVaTe by employing a deep neural network to represent the value function, enabling the algorithm to handle high-dimensional state spaces and improve performance. MinDAgger [12] proposes an asynchronous variant of DAgger that allows for parallel data collection and aggregation, significantly reducing the sample complexity and training time. These DAgger variants aim to enhance the efficiency and practicality of imitation learning by addressing key challenges such as sample complexity [12], expert cost minimization [10], and policy performance optimization [9]. In the field of autonomous driving, one notable approach is DARB (DAgger with replay buffer) [13], which focuses

on critical states and incorporates a replay buffer, thereby achieving significant improvements in generalization performance for autonomous driving scenarios. Additionally, The Roach [14] algorithm propose a novel reinforcement learning agent as an "automated expert" for autonomous driving, outperforming existing methods and enabling effective DAgger to achieve expert-level performance.

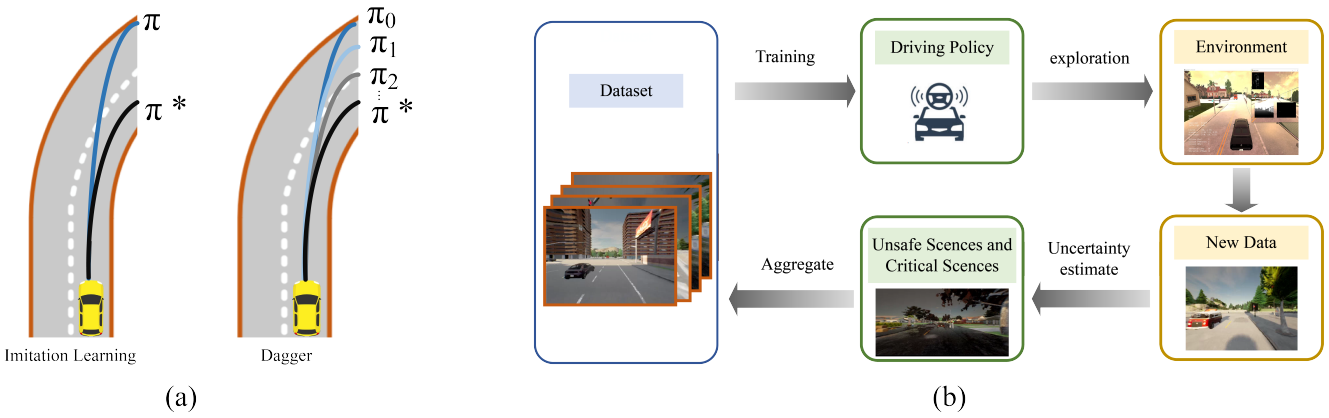


Figure 1. (**a**) Imitation learning, an off-policy method, propagates errors early in the trajectory. In contrast, DAgger is an on-policy approach that integrates the expert's and the agent's control, requiring the expert to label unfavorable states visited by the agent's policy. The agent leverages this data iteratively to refine its own policy through multiple training iterations, progressively approaching the expert's policy. (**b**) We propose a modified version of DAgger with uncertainty estimate and critical scenes for improved driving in dense urban scenarios. In contrast to traditional DAgger algorithms, we employ a deep Bayesian uncertainty estimation method to determine whether a scene is extracted for retraining.

However, the DAgger algorithm's propensity to frequently visit suboptimal states can lead to several challenges, including increased expert burden, reduced data efficiency, degraded policy performance, and instability in the learning process. To address these issues, researchers have proposed various improvements to the algorithm. SafeDAgger introduces a safety policy that selectively queries expert labels, thereby reducing the reliance on experts and enhancing the efficiency and safety of data aggregation [15]. In contrast, DART injects noise during the data collection phase to generate perturbed trajectories that simulate the errors of the learning policy [16]. This data augmentation technique improves the robustness of the learning policy in suboptimal states. Both approaches alleviate the problem of frequent visitation to suboptimal states in the DAgger algorithm from different perspectives, ultimately enhancing the performance and efficiency of imitation learning. These algorithmic advancements demonstrate the ongoing efforts in the research community to address the limitations of the original DAgger algorithm and push the boundaries of imitation learning for real-world applications.

Furthermore, a growing number of researchers are utilizing Bayesian learning methodologies to tackle this problem. Bayesian approaches to the DAgger algorithm, such as EnsembleDAgger [17], DropoutDAgger [18], address the critical issue of quantifying uncertainty in the decision-making process of imitation learning. These Bayesian variants of DAgger aim to improve upon the original algorithm by providing a more nuanced understanding of the learning agent's confidence in its actions. EnsembleDAgger [17] leverages a collection of diverse models to form a committee that votes on the best action, integrating model variance as an implicit measure of uncertainty. DropoutDAgger [18] utilizes Monte Carlo dropout (MC-dropout) [19] within neural networks to approximate Bayesian inference, thus offering a practical way to estimate uncertainty by capturing the predictive variance in the network's output. BAgger [20], or Bayesian aggregation, incorporates explicit Bayesian methods to model the uncertainty of the policy's actions, often through probabilistic modeling techniques. By integrating these Bayesian concepts,

these algorithms enable the learning agent to make more informed decisions about when to defer to the expert, thereby potentially reducing the number of suboptimal actions taken and improving the overall performance of the system.

Closely related to our work, UAIL (uncertainty aware imitation learning) [21] proposed a DAgger-based learning algorithm that utilizes MC-dropout to estimate uncertainty in autonomous driving actions. This algorithm leverages uncertainty estimates during on-policy data collection, allowing the learning agent to switch control at uncertain states and collect data targeted at corrective behaviors at the boundary of optimal and suboptimal states. However, MC-dropout often exhibits inaccurate predictions and high computational costs on large-scale datasets [22,23]. Furthermore, UAIL's implementation of DAgger lacks critical states and replay buffer techniques [13], which limits its overall effectiveness.

Imitation learning has shown promise for autonomous driving but suffers from covariate shift, where the policy performs poorly in unseen environments. DAgger is a popular approach that addresses this by leveraging expert demonstrations. However, DAgger's frequent visits to suboptimal states can lead to several challenges, such as increased expert burden, reduced data efficiency, degraded policy performance, and instability in the learning process.

To address these limitations, we propose a novel DAgger framework that integrates Bayesian uncertainty estimation via MFVI [24,25]. The architecture of the method is shown in Figure 1b. MFVI provides better-calibrated uncertainty estimates compared to prior methods, such as MC-dropout used in UAIL. The proposed algorithm consists of several key components: an uncertainty estimation module, a critical state identification process, and an expert querying mechanism. During training, the framework first uses MFVI to estimate the uncertainty of the agent's predictions. It then identifies both uncertain and critical states—those states where the policy's performance is most uncertain or potentially suboptimal. The expert is queried only for these specific states, enabling a targeted data collection strategy. This approach not only reduces the burden on the expert but also improves data efficiency by focusing on the most informative samples. Evaluations on the CARLA simulator demonstrate that our approach outperforms existing methods, highlighting the effectiveness of Bayesian uncertainty estimation and targeted data aggregation for imitation learning in autonomous driving.

The key contributions of our work are as follows:

- We introduce a novel DAgger framework that synergistically integrates Bayesian uncertainty estimation via MFVI and critical state identification for improved imitation learning in autonomous driving.
- We demonstrate that MFVI provides better-calibrated uncertainty estimates compared to MC-dropout, leading to more effective data collection and improved driving performance.
- We conduct extensive evaluations on the CARLA simulator [26] using the NoCrash [27] and CARLA Leaderboard benchmarks [28], showing that our approach outperforms existing methods, such as CILRS (conditional imitation learning with ResNet and speed prediction) [27], DARB, and UAIL.

Our work advances imitation learning for autonomous driving by demonstrating the effectiveness of integrating Bayesian uncertainty estimation and targeted data aggregation. This makes it a direction with valuable insights for future work in this domain.

2. Background

2.1. Imitation Learning

Imitation learning [29], also known as learning from demonstrations, is a form of machine learning in which an agent learns a policy π from observed state–action pairs (s_i, a_i), where the actions a_i are provided by an expert policy π_E. The goal of the agent is to learn a policy that can imitate the expert policy, either by minimizing the difference between their actions or by maximizing the likelihood of actions taken by the expert. The learned policy can then be used to make decisions in new, unseen environments.

Formally, let $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, respectively. The expert policy $\pi_E : \mathcal{S} \rightarrow \mathcal{A}$ maps states to actions. The agent's goal is to learn a policy $\pi_\theta : \mathcal{S} \rightarrow \mathcal{A}$, parameterized by θ, that mimics the expert's behavior. Given a dataset of expert demonstrations $\mathcal{D} = (s_i, a_i)_{i=1}^N$, where $s_i \in \mathcal{S}$ and $a_i \in \mathcal{A}$, the objective of imitation learning can be formulated as:

$$\min_\theta \mathbb{E}_{s \sim d_{\pi_E}} [L(\pi_\theta(s), \pi_E(s))] \tag{1}$$

where d_{π_E} is the state distribution induced by the expert policy, and L is a loss function that measures the dissimilarity between the actions taken by the learned policy and the expert policy.

2.2. DAgger

DAgger stands for "Dataset Aggregation" and is a combination of supervised and reinforcement learning. It involves collecting a dataset of expert demonstrations and using this data to train a supervised learning model. The model is then used to perform the task, and the expert provides feedback to the model by correcting its errors. This feedback is then used to update the model, making it more accurate and better able to perform the task.

The key idea behind DAgger is to iteratively collect additional data from the states visited by the learned policy and have the expert provide labels for those states. By aggregating the new data with the existing dataset and retraining the policy, DAgger helps mitigate the distribution shift problem and improves the policy's performance. The overall workflow of the Dagger algorithm is outlined in Algorithm 1.

Algorithm 1 DAgger

Collect initial dataset D_0 using expert policy π^*
Initialize dataset $D \leftarrow D_0$
Train initial policy $\hat{\pi}_0 = \text{argmin}_\pi \mathcal{L}(\pi, \pi^*, D_0)$
for $i = 1\ N$ **do**
  Generate on-policy dataset D_i using $\hat{\pi}_{i-1}$
  Ask the expert π^* for labels for D_i to get D_i'
  Aggregate datasets: $D \leftarrow D \cup D_i'$
  Train policy $\hat{\pi}_i = \text{argmin}_\pi \mathcal{L}(\pi, \pi^*, D)$
end for
return $\hat{\pi}_N$

In this formulation, $\mathcal{L}(\pi, \pi^*, D)$ represents the loss function that measures the difference between the actions predicted by the policy π and the expert's actions π on the dataset D. The specific choice of loss function depends on the problem domain and the type of actions being predicted (e.g., mean squared error for continuous actions, cross-entropy loss for discrete actions).

By iteratively collecting data from the learned policy, getting expert labels, and retraining the policy, DAgger effectively combines supervised learning and reinforcement learning to improve the policy's performance and generalization ability. The algorithm has been successfully applied in various domains, including robotics, autonomous driving, and game playing.

2.3. Bayesian Neural Networks and Inference

Deep Bayesian algorithms are a class of methods that combine Bayesian inference with deep learning to probabilistically model the weights of deep neural networks, thereby enabling the estimation of model uncertainty.

Let $f_\theta(x)$ denote a deep neural network, where θ represents the network weights and x is the input. In the Bayesian framework, we treat the weights θ as random variables and model them with a prior distribution $p(\theta)$. Given a training dataset $\mathcal{D} = (x_i, y_i)_{i=1}^N$, our goal is to compute the posterior distribution $p(\theta|\mathcal{D})$. According to Bayes' theorem:

$$p(\theta \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \theta)p(\theta)}{p(\mathcal{D})} \propto p(\mathcal{D} \mid \theta)p(\theta) \tag{2}$$

However, for deep neural networks, the posterior distribution $p(\theta|\mathcal{D})$ is often intractable. Therefore, we typically resort to approximate inference methods, such as variational inference (VI) [24] or Markov chain Monte Carlo (MCMC) [30], to approximate the posterior distribution.

Mean field variational inference (MFVI) [31]: MFVI approximates the true posterior distribution $p(\theta|\mathcal{D})$ with a simpler variational distribution $q_\phi(\theta)$, parameterized by ϕ. The objective is to minimize the Kullback–Leibler (KL) divergence between $q_\phi(\theta)$ and $p(\theta|\mathcal{D})$, which is equivalent to maximizing the evidence lower bound (ELBO). The optimal variational parameters ϕ^* are obtained through optimization techniques such as gradient descent. Figure 2c illustrates the optimization process.

Monte Carlo dropout (MC-dropout) [19]: MC-dropout is a practical approach that interprets dropout regularization as an approximate Bayesian inference method. By enabling dropout during both training and testing phases, MC-dropout approximates the posterior distribution over the weights. Multiple forward passes with different dropout masks allow for the estimation of predictive uncertainty.

Deep Bayesian learning plays a pivotal role in enhancing the safety and reliability of autonomous vehicles by providing a probabilistic framework that can effectively handle uncertainty in perception [32–34], prediction [35], and decision-making tasks [36–38]. The integration of Bayesian inference with deep learning models allows for the quantification and propagation of uncertainty through various layers of the system, which is crucial for developing robust AVs that can operate in dynamic and unpredictable environments.

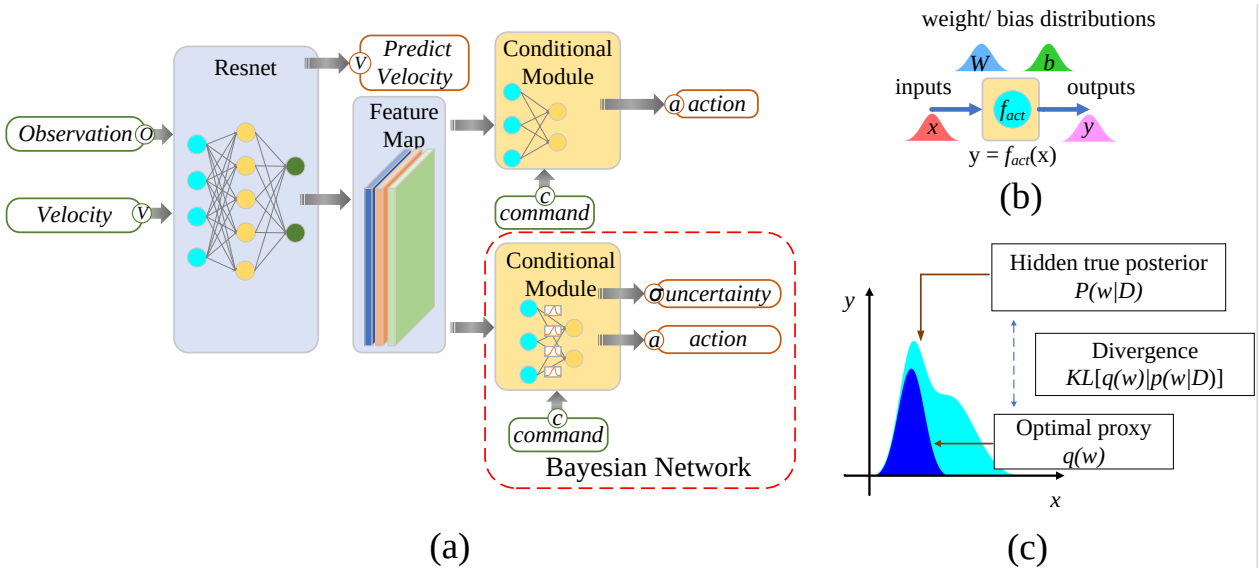


Figure 2. (**a**) The schematic representation of our imitation learning network and deep Bayesian network. They have the same latent space. In our imitation learning network, we employ the CILRS framework. Initially, an input image undergoes processing by a ResNet perception module, resulting in a latent space representation. Subsequently, two prediction heads are employed: one for controls and the other for speed. In our deep Bayesian network, the latent space serves as input for predicting the uncertainty associated with controls. This network comprises interconnected Bayesian neurons organized in layers, each potentially varying in activation functions, weight distributions, and bias distributions. (**b**) A Bayesian neuron, the mathematical operation involves an activation function, a distribution of weights w, and a distribution of biases b specific to that neuron. When processing inputs x, the network samples one instance of weights and biases from their respective distributions and applies the activation function accordingly. This approach allows for uncertainty estimation and robust modeling in neural networks. (**c**) The variational inference for analyzing the optimal posterior distribution $p(\theta|\mathcal{D})$ by estimating a relatively simpler distribution $q(\theta)$.

3. Method

3.1. Imitation Learning for Autonomous Vehicles

Our proposed framework employs CILRS [27] as the imitation learning algorithm. CILRS is an end-to-end architecture specifically designed for autonomous driving. It extends the traditional behavior cloning approach by incorporating conditional imitation learning (CIL) [1], enabling it to effectively handle complex urban driving scenarios that demand nuanced navigational decisions.

As show in Figure 2a CILRS employs a CIL framework, which extends imitation learning cloning by incorporating high-level navigational commands to disambiguate actions in complex driving scenarios. The network structure consists of an input module that processes sensory observations, including single RGB images and ego vehicle speed, to capture the dynamic driving environment. This input is then fed into a deep neural network that simultaneously learns perception and control tasks. The architecture includes convolutional layers for feature extraction from the image data, followed by fully connected layers that integrate the high-level commands with the visual features to predict the vehicle's control parameters. This design allows the network to learn end-to-end policies capable of handling intricate driving maneuvers.

Consider a dataset of observation–action pairs $D(o_i, a_i)_{i=1}^{N}$, collected from expert policy. The goal of imitation learning is to learn a policy $\pi(o_t) : O \rightarrow A$, which maps observations o_t to actions a_t at at every time step and imitates the behavior of an expert action:

$$\text{IL} : \min_{\pi} \sum_{i} [(a_t, \pi(o_i))] \tag{3}$$

The policy is optimized by minimizing a distance L between the predicted action and the expert action, In our autonomous driving model, the output of the policy is a three-dimensional continuous action vector (steer, throttle, and brake). Specifically, we use a mean squared error (MSE) loss for training, which measures the average squared difference between the predicted actions and the expert actions. By minimizing this loss, the policy is trained to produce actions that closely match the expert's actions, thus effectively imitating the expert's behavior.

CIL add some high-level command c that can convey the planning module intent at test time. This command can guide the car's driving direction, thus avoiding multimodality behavior. For example, when the car is at the intersection, whether the car should turn left or right, or go straight. If tell the car a planning command c helps resolving the ambiguity. The training dataset becomes $D\{(o_i, c_i, a_i)\}_{i=1}^{N}$, High-level commands include three: turn left, turn right, and go straight. The command-conditional imitation learning objective can be written as

$$\text{CIL} : \min_{\pi} \sum_{i} [(a_t, \pi(o_i, c_i))] \tag{4}$$

Imitation learning assumes that the data distribution is independently identically distributed, and imitation cannot accurately replicate the expert action. And because autonomous driving is a sequential decision-making problem, each errors will gradually accumulate. This causes the probability of accident to increase over time while the car is running. Another problem is that it is easy for the car to make mistakes if it encounters states that are not present in the expert data distribution. This problems can be solved using iterative on-policy algorithms such as DAgger, which we discuss next.

3.2. Mean Field Variational Inference

We constructed a deep Bayesian network to evaluate the uncertainty in end-to-end autonomous driving systems. As show in Figure 2a, the network's input is the feature output layer from the CILRS backbone network, while the output comprises the control actions along with their associated uncertainties. Mirroring the latter half of the CILRS architecture, each driving command module corresponds to a deep Bayesian network.

The network structure consists of six fully connected layers. The prior parameters for the Bayesian neural network (BNN) are initialized with a mean of 0.0 and a standard deviation of 1.0.

MFVI operates on the principle of approximating the complex posterior distribution $p(w \mid \mathcal{D})$ of the model parameters w, given the dataset $\mathcal{D}$, with a simpler variational distribution $q_\lambda(w)$. This variational distribution is typically factorized across the dimensions of w to ease computations:

$$q_\lambda(w) = \prod_{i=1}^{D} q_{\lambda_i}(w_i) \tag{5}$$

where D represents the dimensionality of w, and $\lambda = \{\lambda_1, \ldots, \lambda_D\}$ are the variational parameters we aim to optimize.

Within the framework of MFVI, the optimization objective is to approximate the intractable true posterior distribution $p(w \mid \mathcal{D})$ with a variational distribution $q_\lambda(w)$ that is more computationally feasible to handle. This approximation is achieved by minimizing the Kullback–Leibler (KL) divergence from $q_\lambda(w)$ to $p(w)$, which is a measure of the loss of information when $q_\lambda(w)$ is used to approximate $p(w)$. The KL divergence is given by the integral

$$\mathrm{KL}[q_\lambda(w) \| p(w)] = \int q_\lambda(w) \log \frac{q_\lambda(w)}{p(w)} dw \tag{6}$$

which quantifies the expected value of the logarithmic difference between $q_\lambda(w)$ and $p(w)$, calculated with respect to $q_\lambda(w)$.

The evidence lower bound (ELBO), which is formulated as:

$$\mathcal{L}(\lambda) = \mathbb{E}_{q_\lambda(w)}[\log p(\mathcal{D} \mid w)] - \mathrm{KL}[q_\lambda(w) \| p(w)] \tag{7}$$

here, the first term is the expected log-likelihood of the data given the parameters, indicating how well the model fits the data. The second term is the KL divergence, which provides regularization.

The loss function to be minimized is then defined as the negative ELBO:

$$\mathcal{J}(\lambda) = -\mathcal{L}(\lambda) \tag{8}$$

During inference, we employ Monte Carlo sampling to approximate the predictive distribution. Forward passes are performed through the network T times, each time sampling different weights $w \sim p(w|D)$ from the posterior distribution. The model output a^t is computed for each sample t as $a^t = f(x, w^t)$, where f is the neural network mapping inputs x to outputs y given weights w^t. The outputs from the T samples are averaged to obtain the final prediction:

$$a_{\text{pred}} = \frac{1}{T} \sum_{t=1}^{T} a^{(t)} \tag{9}$$

by sampling multiple times with varying weights, we derive a robust prediction encompassing the full posterior $p(a|x, D)$.

Predictive uncertainty is quantified by calculating the predictive entropy of the output distribution over control actions a. The predictive entropy is then calculated as

$$\sigma(a|x, D) = -\int p(a|x, D) \log p(a|x, D) da \tag{10}$$

where a is the output (driving action), x is the input , and D is the training data. Predictive uncertainty captures the inherent noise and variability in the data.

Model uncertainty is quantified using mutual information. Mutual information captures the dependence between the output a and the model parameters ω by measuring the reduction in uncertainty of a when ω is known:

$$\sigma(a, \omega|x, D) = H(a|x, D) - \mathbb{E}_{p(\omega|D)}[H(a|x, \omega)] \tag{11}$$

A higher mutual information indicates the model is more uncertain about its own predictions.

Uncertainty quantification in BNNs provides valuable insights into the reliability of the model's predictions. By distinguishing between data uncertainty and model uncertainty, we can make safer and more informed decisions in the context of autonomous driving. Model uncertainty, which can be mitigated by incorporating sufficient data, is often quantified through stochastic forward passes using dropout. However, this approach is computationally expensive and not suitable for real-time applications. In this paper, we primarily focus on data uncertainty, which arises from the inherent noise or ambiguity present in the input data.

We leverage the data uncertainty present in the prediction of the driving policy to identify safe states. Consequently, the set of critical states, denoted as $\mathcal{S}_{\text{unsafe}}$, can be determined as a result of this process.

$$\mathcal{S}_{\text{unsafe}} = \{s \in \mathcal{S} \mid \sigma(\text{s}) > \tau\} \tag{12}$$

The selection of the safety threshold parameter τ is crucial in achieving a harmonious trade-off between ensuring safety and optimizing query efficiency.

3.3. Critical Scenes

The DAgger algorithm appends the entire generated on policy trajectory to the training dataset for the current iteration. However, not all states present the same utility for the driving policy. Specifically, states that correspond to failure cases have maximum utility for accelerate the convergence of DAgger algorithm. On this basis, according to the characteristics of automatic driving tasks, the definition of key frames is introduced, so that the training strategy can be more focused on the driving task.

Scene-based: In the context of dense urban driving, tasks such as navigating intersections are deemed more critical than traveling straight on an empty road, as most collisions occur at intersections and during turning maneuvers. Additionally, collisions are prone to happen when pedestrians unexpectedly cross the road. Therefore, we define the following key scenarios: (1) intersection, (2) a turning scenario, and (3) pedestrians on the road. Consequently, we prioritize sampling these scenarios, where scenarios (1) and (2) align with the definitions provided by DAgger.

Safe time-based: The level of risk is mainly reflected by the interaction between the autonomous vehicle and objects in the scenario, which can be naturally described by the distance—a small distance means the risk of collision is high. This intuition can also be converted to other metrics such as time to collision (TTC) [39,40]. The TTC of a vehicle–driver combination i at instant t with respect to a leading vehicle $_{i-1}$ can be calculated with

$$TTC_i = \frac{x_{i-1}(t) - x_i(t) - l_i}{\widehat{x}_i(t) - \hat{x}_{i-1}(t)}, \forall \widehat{x}_i(t) > \hat{x}_{i-1}(t) \tag{13}$$

In this formula, $x_{i-1}(t)$ represents the position of the leading vehicle (vehicle$_{i-1}$) at time t. Similarly, $x_i(t)$ denotes the position of the following vehicle (vehicle$_i$) at the same time. The term l_i specifies the length of the following vehicle (vehicle$_i$). Additionally, $\widehat{x}_i(t)$ indicates the speed (or velocity) of the following vehicle (vehicle$_i$) at time t, while $\hat{x}_{i-1}(t)$ represents the speed of the leading vehicle (vehicle$_{i-1}$) at the same instant.

The TTC represents the time remaining before a collision occurs if both vehicles continue at their current speeds. The numerator, $x_{i-1}(t) - x_i(t) - l_i$, represents the gap

distance between the following vehicle and the leading vehicle, adjusted for the length of the following vehicle. The denominator, $\hat{x}_i(t) - \hat{x}_{i-1}(t)$, represents the relative speed between the two vehicles. The condition $\hat{x}_i(t) > \hat{x}_{i-1}(t)$ ensures that the following vehicle is moving faster than the leading vehicle, which is necessary for a potential collision to occur.

According to international test practices and industry research [39,41], the TTC threshold for household vehicles is set to 2.4 s. Below this value, vehicles are prone to collision. This threshold is based on extensive experimental data and real-world testing, ensuring enough reaction time for drivers or automated systems to take evasive action and avoid accidents. Specifically, the 2.4 s threshold accounts for average human reaction times combined with the mechanical response times of vehicles, providing a safe margin to prevent collisions.

We can obtain the safety factor of the vehicle according to the TTC value, which is defined as follows:

$$\mathcal{S}_c = \{s_c \in \mathcal{S} \mid TTC_i < 2.4\} \tag{14}$$

Overall, scene-based and safe time-based sampling play a significant role in enhancing the focus and effectiveness of the training strategy within the context of autonomous driving tasks.

3.4. DAgger with Uncertainty Estimates and Critical States

DAgger is an iterative training algorithm that collects on-policy data at each iteration under the current policy and trains the next policy under the aggregate of all collected data. This iteration continues until the supervised cost on a validation set stops improving.

Our proposed DAgger algorithm synergistically combines uncertainty estimation with the identification of critical states, with the detailed procedure depicted in Algorithm 2. In any given scenario s, for every action a executed under the agent policy $\hat{\pi}$, we commence by utilizing a Bayesian neural network to conduct an uncertainty prediction. Should the resulting uncertainty surpass a predefined threshold τ, we defer control to the expert policy π^*, thereby preventing the traversal of a sequence of suboptimal states. The algorithm leverages uncertainty estimation to orchestrate the data collection process, focusing on acquiring data that are most instrumental in enhancing model performance. specifically, correction behaviors at the boundary between optimal and suboptimal states.The selection of the policy at each sampling instance is shown in Equation (15), where VI is the uncertainty prediction network. Concurrently, for each scenario, we extract critical frames using both scene-based and safe time-based methods.

$$\pi = \begin{cases} \pi^*, VI(s) > \tau \\ \hat{\pi}, \text{ otherwise} \end{cases} \tag{15}$$

The determination of the threshold τ is a significant element within our computational framework. The threshold is computed iteratively by aggregating the uncertainty estimates, represented by $\sigma(a \mid s)$, for all scenarios within a given iteration to obtain their mean μ and standard deviation σ^*. We then define the threshold τ as $\tau = \mu + \sigma^*$.

The theoretical justification for adopting the threshold $\tau = \mu + \sigma^*$ is rooted in the properties of the normal distribution. In a normal distribution, approximately 68% of the data lies within one standard deviation of the mean. Thus, by setting the threshold to $\mu + \sigma^*$, our algorithm targets scenarios that exhibit a level of uncertainty above what is typical for 68% of the scenarios observed. This method ensures that the scenarios selected for sampling present a higher degree of uncertainty, which is pivotal for the algorithm to identify and learn from more challenging and less certain states.

We also employed the replay buffer method proposed by the authors. The fixed size of the replay buffer meticulously controls the blend of expert and on-policy data, enabling the learning algorithm to successively refine and augment the policy's capabilities, especially in its areas of deficiency, culminating in a more adept and flexible driving policy.

Algorithm 2 DAgger with Uncertainty Estimate and Critical States

Collect D_0 using expert policy π^*
$\hat{\pi}_0 = \text{argmin}_\pi \mathcal{L}(\pi, \pi^*, D_0)$
Initialize variational inference network $\text{VI}_0(D_0)$
Initialize uncertainty threshold τ_0
Initialize replay buffer $D \leftarrow D_0$
Let $m = |D_0|$
for $i = 1$ to N **do**
  Generate on-policy datasets D_i using $\pi(\pi = \pi^*, VI(s) > \tau; \pi = \hat{\pi}, otherwise)$
  sampling critical states s_c from D_i
  sampling unsafe states :

$$\mathcal{S}_\text{u} \leftarrow \{s \in D_i \mid \text{VI}_{i-1}(\text{s}) > \tau_{i-1}\}$$

  Get $D_i' \leftarrow \{s \in s_c \cup s_u \mid (s, \pi^*(s))\}$ of visited states by expert $\hat{\pi}^*$
  Combine datasets: $D \leftarrow D \cup D_i'$
  while $|D| > m$ **do**
    Sample $(s, \pi^*(s))$ randomly from $D \cap D_0$
    $D \leftarrow D - \{(s, \pi^*(s))\}$
  end while
  Train $\hat{\pi}_i = \text{argmin}_\pi \mathcal{L}(\pi, \pi^*, D)$ with policy initialized from $\hat{\pi}_{i-1}$
  Training $\text{VI}_i(D)$.
  Update τ_i.
end for
return $\hat{\pi}_N$

4. Experiments and Evaluation

4.1. Environment

We utilize the CARLA simulator as the environment for training and evaluation of our autonomous driving models. The NoCrash benchmark serves as our primary assessment tool. Each NoCrash scenario defines specific training conditions, including the virtual towns (Town 1 for training and Town 2 for testing) and weather settings, for data collection [26,27]. Subsequently, the agent's performance is evaluated in novel towns and weather conditions not encountered during training.

The NoCrash benchmark examines the agent's ability to generalize from Town 1, meticulously designed as a training environment, to Town 2, designed as a testing environment. Figure 3 shows maps of these towns and representative views. Town 1 features an intricate network of 2.9 km of drivable roads, meticulously designed to replicate real-world urban landscapes, featuring diverse urban structures (buildings, vegetation), traffic signage, and infrastructure. The inclusion of intricate intersections, pedestrian crossings, and a variety of weather and lighting conditions makes Town 1 ideal for training robust ADS models. The extensive and varied layout challenges these models with diverse scenarios, fostering the development of robust perception, planning, and control modules.

Town 2, a scaled-down version of Town 1, presents a more compact environment (1.4 km of drivable roads) with an equally detailed level of fidelity. This environment features a distinct urban layout with unique textures and 3D models, including a mix of single-lane and multi-lane roads, as well as more complex junctions compared to Town 1. Town 2 serves primarily as a testing ground, offering an unseen environment to assess the adaptability and robustness of models trained in Town 1. By encountering novel road layouts, building structures, and environmental textures, this approach verifies the generalization capabilities of the models, ensuring they can handle variations within a similar urban style.

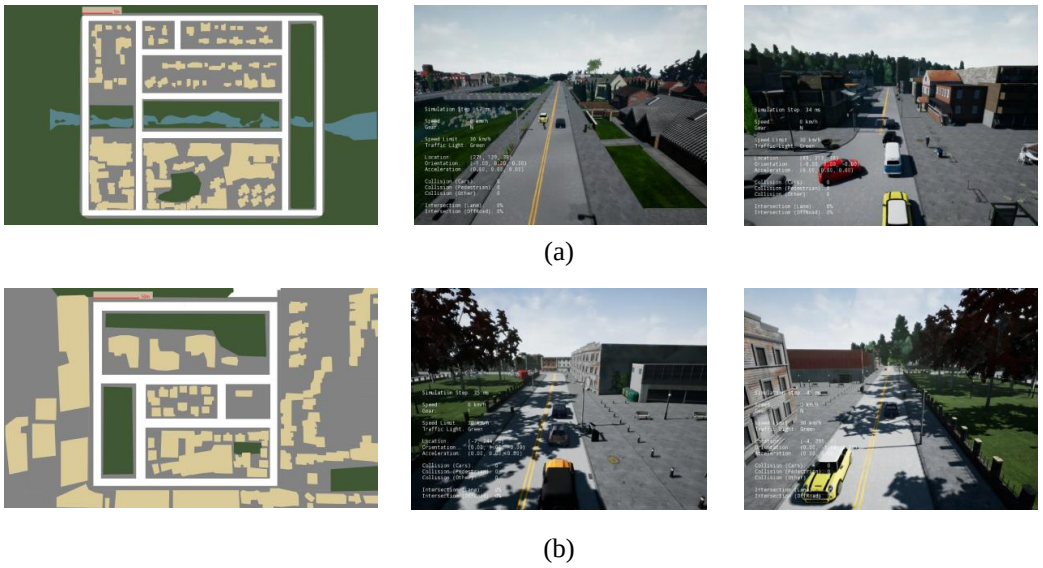


Figure 3. The two CARLA towns. (**a**) The map of Town 1 and two example scenes. (**b**) The map of Town 2 and two example scenes.

Consistent with the NoCrash benchmark, our driving policy is developed using datasets collected in Town 1 under four distinct weather conditions. The policy's robustness is then assessed across various scenarios: Training, New Weather (NW), New Town (NT), and the combined New Town and Weather (NTW) conditions. The NoCrash benchmark categorizes traffic density into three distinct levels: empty, regular, and dense. These classifications determine the quantity of pedestrians and vehicles populating each map. For the evaluations conducted on the CARLA Leaderboard, the traffic density of each map has been calibrated to correspond with the established busy traffic parameters.

4.2. Uncertainty Estimation

4.2.1. Mean and Standard Deviation of Uncertainty Estimates

The imitation agent and our uncertainty estimation system were tested in a novel environment, Town 2, which is part of the CARLA benchmark. This town features a smaller and more varied layout compared to Town 1 and includes both old and new weather conditions, using a subset of test cases provided in the benchmark. The network outputs three control signals: steering angle, throttle value, and brake. Uncertainties for the control signals were computed independently. The tested uncertainty estimation signals include steer standard error, throttle standard error, and brake standard error. We present several specific examples of uncertainty estimation. Figure 4a demonstrates the effect of different lighting and weather conditions on uncertainty estimation within the same scene. Specifically, it includes three images, each corresponding to different lighting and weather scenarios. These variations significantly affect the uncertainty estimates, highlighting the importance of accurate uncertainty estimation under diverse environmental conditions. Figure 4b presents additional scenarios where the estimated uncertainty exceeds a predefined threshold. This part of the figure emphasizes challenging situations such as changes in road surface illumination, traffic lights, sharp turns, and other novel or complex conditions. The high uncertainty estimates in these scenarios underscore the potential of our proposed DAgger framework to handle complex driving environments effectively.

To evaluate the sensitivity of uncertainty estimation to novel scenarios. We collected four distinct datasets, each comprising 10,000 frames, under varied scenario conditions. under each high-level command, we independently calculated the mean value u and standard deviation σ of the uncertainty for each action. These statistics serve as the main basis for calculating the uncertainty threshold in the subsequent steps. Table 1 presents the corresponding statistical values of action uncertainty obtained from testing in

different environments. The results demonstrate that the uncertainty of the agent's actions increases in new environments, which aligns with the expected behavior of uncertainty estimation principles.

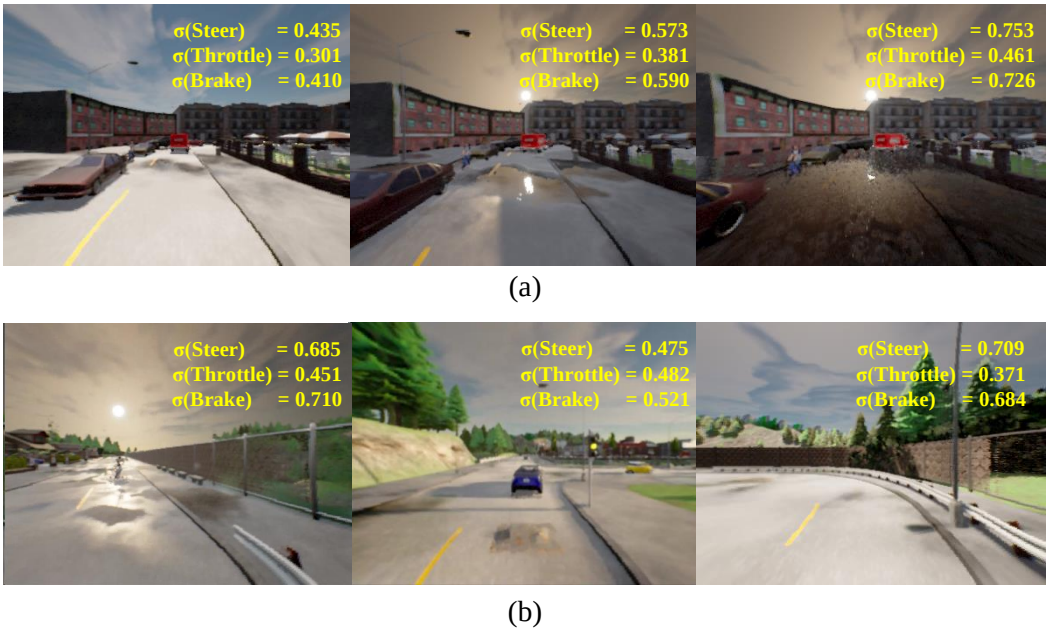


Figure 4. (**a**) The three images illustrate the influence of varying lighting and weather conditions on the estimated uncertainty within the same scene. (**b**) Additional examples where the uncertainty of the estimate exceeds the threshold. For instance, this can occur when the agent encounters changes in road surface illumination, traffic lights, sharp turns, or other novel or complex situations.

Table 1. Mean and standard deviation of uncertainty estimates in different scenarios.

Action	Conditions	Uncertainty Value (u,σ)			
		Follow	**Left**	**Right**	**Straight**
Steer	Train	(0.597,0.169)	(0.782,0.197)	(0.742,0.187)	(0.627,0.156)
	NW	(0.624,0.169)	(0.774,0.212)	(0.792,0.241)	(0.674,0.201)
	NT	(0.627,0.187)	(0.775,0.272)	(0.772,0.236)	(0.672,0.244)
	NWT	(0.644,0.193)	(0.795,0.247)	(0.812,0.263)	(0.694,0.274)
Throttle	Train	(0.352,0.262)	(0.447,0.214)	(0.478,0.274)	(0.382,0.198)
	NW	(0.372,0.257)	(0.467,0.244)	(0.493,0.179)	(0.342,0.231)
	NT	(0.379,0.264)	(0.455,0.264)	(0.497,0.146)	(0.368,0.245)
	NWT	(0.424,0.287)	(0.461,0.234)	(0.498,0.174)	(0.391,0.227)
Brake	Train	(0.742,0.145)	(0.421,0.124)	(0.447,0.126)	(0.712,0.110)
	NW	(0.765,0.121)	(0.415,0.121)	(0.490,0.132)	(0.722,0.152)
	NT	(0.810,0.114)	(0.427,0.112)	(0.493,0.146)	(0.730,0.154)
	NWT	(0.801,0.132)	(0.433,0.164)	(0.489,0.263)	(0.732,0.167)

During the iterative training process of DAgger, we analyzed the changes in u and σ after each training loop. As show in Table 2, the number of iterations increases, the parameters of the deep Bayesian network gradually stabilize, leading to more precise uncertainty estimations. Consequently, u and σ tend to decrease with the increasing number of loops, reflecting the growing confidence in action selection through imitation learning. This variation necessitates dynamic adjustments when selecting the uncertainty threshold τ to adapt to the changes in each iteration.

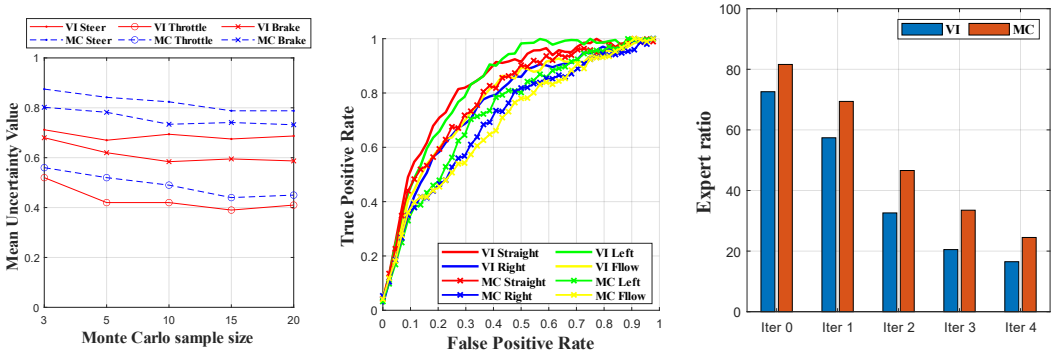
Table 2. Mean and standard deviation of uncertainty estimates in Iter.

Action	Iter	Uncertainty Value (u,σ)			
		Follow	**Left**	**Right**	**Straight**
Steer	Iter 0	(0.597,0.169)	(0.782,0.197)	(0.742,0.187)	(0.627,0.156)
	Iter 1	(0.554,0.154)	(0.724,0.223)	(0.702,0.217)	(0.594,0.201)
	Iter 2	(0.490,0.156)	(0.695,0.190)	(0.684,0.245)	(0.523,0.188)
	Iter 3	(0.467,0.142)	(0.674,0.247)	(0.652,0.193)	(0.497,0.174)
	Iter 4	(0.468,0.141)	(0.664,0.247)	(0.654,0.189)	(0.497,0.187)
Throttle	Iter 0	(0.352,0.262)	(0.447,0.214)	(0.478,0.274)	(0.382,0.198)
	Iter 1	(0.334,0.257)	(0.425,0.241)	(0.452,0.187)	(0.337,0.214)
	Iter 2	(0.319,0.214)	(0.401,0.197)	(0.421,0.146)	(0.318,0.241)
	Iter 3	(0.302,0.155)	(0.394,0.210)	(0.402,0.148)	(0.304,0.187)
	Iter 4	(0.289,0.164)	(0.374,0.232)	(0.432,0.165)	(0.324,0.196)
Brake	Iter 0	(0.742,0.145)	(0.421,0.124)	(0.447,0.126)	(0.712,0.110)
	Iter 1	(0.720,0.137)	(0.396,0.210)	(0.402,0.210)	(0.682,0.201)
	Iter 2	(0.702,0.156)	(0.368,0.192)	(0.394,0.176)	(0.662,0.175)
	Iter 3	(0.675,0.135)	(0.334,0.126)	(0.380,0.193)	(0.630,0.142)
	Iter 4	(0.660,0.136)	(0.326,0.107)	(0.362,0.142)	(0.631,0.137)

4.2.2. Monte Carlo Sample Size

Deep Bayesian algorithms require multiple forward inferences to quantify the uncertainty of the data; this called Monte Carlo sampling. There exists a trade-off between the sample size and the accuracy of uncertainty computation. Generally, more sample size can achieve better accuracy but at the cost of higher computational complexity. However, the computed uncertainty value tends to stabilize as the number of sample size increases. Therefore, it is crucial to select an appropriate number of sample size to balance accuracy and computational complexity.

We evaluated the changes in the mean of all uncertainty in the training set. As shown in Figure 5a, we analyzed the relationship between the sample size and the mean of uncertainty. We also compared our results with the MC-dropout algorithm used in UAIL. The figure demonstrates that the mean of our MFVI algorithm stabilizes after size = 10, while MC-dropout still exhibits significant fluctuations. This indicates that MC-dropout requires more computation to obtain reliable uncertainty estimates. This is theoretically justified because MFVI approximates the posterior distribution by maximizing the evidence lower bound, which can be computed analytically via the log-likelihood term and the KL divergence term, leading to reduced computational complexity.



(a) (b) (c)

Figure 5. A comparison of variational inference and Monte Carlo dropout on uncertainty prediction, based on our test set. (**a**) The mean uncertainty of actions varies with changes in sample size. (**b**) ROC curves for predicting infractions in test environment, the total uncertainty under different commands. (**c**) The average ratio of consultations by the expert policy. Algorithms eventually transfer control to the novice policy.

4.2.3. Infraction Prediction

We evaluated the predictive performance of our proposed candidate uncertainty functions for infraction detection using receiver operating characteristic (ROC) curves. Since there is no public dataset for uncertainty evaluation, we followed the UAIL approach and considered the following events as infractions: collisions, crossing into the opposite lane, and driving onto the curb. Due to the branched structure of the network, ROC curves for different commands are plotted in Figure 5b. In this experiment, we utilize the joint uncertainty as described by UAIL, where the total uncertainty is defined as $U_{\text{total}} = U_{\text{streer}} + \alpha U_{\text{throttle}}, (\alpha = 0.6)$. The different shapes of the ROC curves indicate that different threshold values should be used to achieve similar true-positive ratios across different commands. As can be seen from the figure, the Area Under the curve (AUC) value for the straight command is low. This is because straight driving is relatively simple and does not involve complex steering operations, resulting in lower uncertainty values and reduced sensitivity to dangerous scenarios.

Compared to MC-dropout, our algorithm achieved better AUC values for all commands. This demonstrates that our algorithm can better estimate the uncertainty of the scene. Theoretically, MC-dropout has been shown to be more accurate for small data sets [19]. However, variational inference outperforms MC-dropout in large-scale data sets [22] and when the data distribution is shifted [42]. Since our autonomous driving task involves large-scale data with frequent distribution changes, MFVI is more suitable for uncertainty estimation in the autonomous driving scenario.

4.2.4. Expert Ratio

The performance of uncertainty prediction is also reflected in the ratio of expert queries. As shown in Figure 5c, it can be observed that the algorithm initially relies heavily on expert queries. However, as the number of training iterations increases, the number of expert queries decreases. We can also see that variational inference can reduce the expert ratio faster than MC-dropout. This indicates that the variational inference algorithm can better select uncertain data and help the imitation learning model to train better.

4.3. Driving Performance

4.3.1. Metrics

We present our results using two key performance indicators: the success rate, as proposed by the NoCrash benchmark, and the driving score, an innovative metric introduced by the CARLA leaderboard. The success rate quantifies the proportion of routes completed without incurring any collisions or impasses. The driving score, on the other hand, is calculated as the product of two factors: route completion, which is the percentage of the route distance successfully navigated; and the infraction penalty, which is a cumulative discount applied for each infraction incurred during the drive. For instance, if an agent were to run two red lights during a single route, with a penalty coefficient of 0.7 for each infraction, the overall infraction penalty would be ($0.7 \times 0.7 = 0.49$). The driving score offers a more nuanced assessment than the success rate, accounting for a broader spectrum of infractions, making it particularly advantageous for evaluating performance over longer routes.

4.3.2. Baseline and Alternatives

To analyze the driving performance, we compare our method against several algorithms: CILRS [27], DARB [13], UAIL [21], DA_UE (DAgger with uncertainty estimate using Variational Inference), DA_CS (DAgger with critical scenes), DA_UE+CS (our proposed, DAgger with uncertainty estimate and critical scenes), and UAIL+ (UAIL incorporates advanced DAgger architecture).

CILRS is a seminal end-to-end imitation learning algorithm in the autonomous driving domain, widely used as a control network, including in our own system. DARB represents an extension of the original DAgger algorithm that leverages critical state and replay buffer technology. Notably, DARB incorporates extensive data preprocessing tailored to

autonomous driving tasks. This algorithm has demonstrated promising performance on the NoCrash benchmark, establishing itself as a valuable baseline for research on DAgger algorithms in the context of autonomous driving.

UAIL also employs the DAgger framework and utilizes Monte Carlo dropout for uncertainty estimation. However, UAIL suffers from poor performance due to its outdated DAgger implementation. To compare the impact of different deep Bayesian algorithms on performance, we ported the UAIL algorithm to DARB to obtain UAIL+. Therefore, the only difference between UAIL+ and our approach lies in the choice of the deep Bayesian algorithm.

Due to DARB's use of an advanced DAgger architecture, all of our proposed algorithms are developed based on DARB. Our proposed DA_UE builds upon DARB by using uncertainty estimation via MFVI to extract key frames, providing a more robust approach to handling uncertain and replacing traditional methods. DA_CS is an extension of DARB that focuses on incorporating our defined critical scenes during the training process, thereby improving the policy's ability to respond to high-risk scenarios. DA_UE+CS, our complete algorithm, combines the features of DA_UE and DA_CS, leveraging uncertainty estimation to extract key frames and identifying critical safety scenes, aiming to utilize the strengths of both approaches for enhanced performance.

We compare our approach against the aforementioned algorithms for performance evaluation. For our infraction analysis, we focus on DARB since it significantly outperforms other approaches and serves as a strong baseline.

4.3.3. Performance and Ablation Study

The performance of the models on the NoCrash benchmark at each DAgger iteration is shown in Figure 6. As expected, all models improved after iterative training with the DAgger mechanism. While the backbone networks of all models were the same, based on CILRS, the algorithms differed mainly in data selection and training methods. Our algorithm, which combines MFVI with safe critical states, achieved good results in all scenarios. Our autopilot can achieve a driving score of 30% and a success rate of 40%.

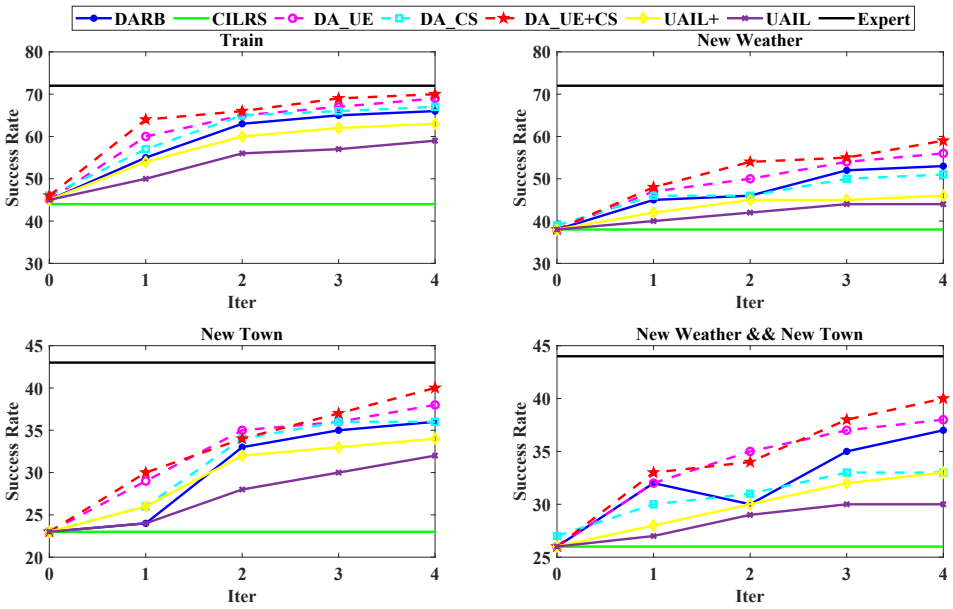


Figure 6. Success rate of different methods across conditions. experimental environments: NW (New Weather), NT (New Town), NTW (New Town and Weather). Algorithms used in our experiments: DARB [13], CILRS [27], DA_UE (DAgger with uncertainty estimate use variational inference), DA_CS(DAgger with critical scenes), DA_UE+CS (our proposed, DAgger with uncertainty estimate and critical scenes), UAIL [21], UAIL+ (UAIL with critical scenes).

We examined whether on-policy data help to improve our driving performance. We mainly used two techniques to improve the agent's performance: critical state and uncertainty estimation. As shown in the figure, both methods improved performance. However, the effects and principles of these two methods are different.

Critical scenes slightly improved performance, thereby affirming that the sampled critical states contain useful information that facilitates improved driving behavior. However, on the new weather condition, the performance of critical scenes starts to decline. This indicates that the scenario selection method of critical scenes does not include weather changes, so it is not good at handling such scenarios.

Uncertainty estimation significantly improved performance in all scenarios. This indicates that calculating the uncertainty of scenarios and selecting suboptimal data does improve the performance of the model. This is consistent with previous research.

4.3.4. Comparison against DARB

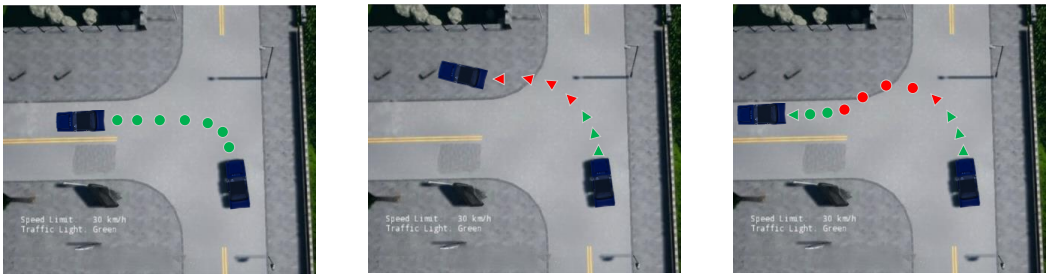
DARB is a notable approach in imitation learning for autonomous driving, focusing on aggregating data from critical and high-uncertainty states. This method employs a replay buffer mechanism to prioritize these states, aiming to enhance the generalization performance of the driving policy. However, it is essentially still the traditional DAgger algorithm, which requires frequent access to the expert policy when collecting trajectories and may also collect a lot of suboptimal data. Our proposed DAgger framework with MFVI shares a similar goal but offers distinct advantages over DARB.

We employ an uncertainty-based prediction method combined with active learning, which differs significantly from traditional DAgger algorithms such as DARB in the way training trajectory is collected. As shown in Figure 7, we illustrate the differences in data collection methods through specific experimental examples. The DARB method, particularly during the early stages of training when the performance of the learned policy is suboptimal, tends to collect a large amount of poor and unnecessary suboptimal data. Figure 7b demonstrates such an example, where the vehicle driven by the learned policy ends up on the sidewalk. Learning how to drive on the sidewalk is not useful, and once in this suboptimal state, even the expert policy has difficulty recovering, let alone providing guidance to the learned policy. In our experiments, when scenarios such as driving onto the sidewalk or colliding with a lamppost occur, the expert policy often fails to recover. Collecting such trajectory data results in poor guidance and labeling by the expert policy, which degrades the quality of the training data and subsequently the training effectiveness of the learned policy.

The uncertainty-based prediction method combined with active learning helps to some extent in avoiding the collection of suboptimal data. As shown in Figure 7c, when the uncertainty of the learned policy exceeds the threshold, control is immediately handed over to the expert, who gradually recovers to a normal state before returning control to the learned policy. This approach prevents the continuous collection of suboptimal data, thereby improving the quality of the training data and the effectiveness of the learned policy's training.

To validate the effectiveness of the uncertainty-based prediction method for dataset collection, we modified the data collection approach of DARB to incorporate uncertainty prediction and active learning, naming the algorithm DA_UE. As shown in Figure 6, DA_UE's success rate surpasses that of DARB from the initial training stages and eventually approaches expert-level performance. Furthermore, DA_UE demonstrates superior generalization on the test set, particularly under NT and NW conditions where DARB's generalization capability significantly declines, yet DA_UE maintains strong performance. Using the Carla leaderboard testing methodology, as shown in Table 3, DA_UE achieved a driving score of 27.9, representing a 32% improvement over DARB. Evaluating the individual test components, we also observe enhanced safety with DA_UE. This indicates that leveraging uncertainty prediction combined with active learning for data collection effectively avoids the accumulation of suboptimal data, thereby improving the quality and

diversity of the dataset annotations and significantly enhancing the performance of the learned policy.



(a) (b) (c)

Figure 7. Trajectory data for different control methods. In the diagrams, circles represent expert trajectories, while triangles indicate trajectories generated by the learned policy. Green signifies that uncertainty remains below the threshold, indicating a safe state. Conversely, red signifies that uncertainty exceeds the threshold, indicating an unsafe state. (**a**) The vehicle is entirely driven by the expert π^*. (**b**) The vehicle is driven entirely by the learned policy $\hat{\pi}$, reflecting the traditional DAgger algorithm, such as DARB, for collecting training trajectories. This approach tends to collect a significant amount of suboptimal data. (**c**) The uncertainty-based prediction method, where the expert policy π^* takes over driving when the learned policy's uncertainty exceeds the threshold, thereby avoiding the collection of excessive suboptimal data.

Table 3. Driving performance and infraction analysis of IL agents on NoCrash-busy.

	DS (%, ↑)	RC (%, ↑)	IS (↑)	CO (↓)	CP (↓)	CV (↓)	RI (↓)	AB (↓)
Unit	%, ↑	%, ↑	%, ↑	/Km, ↓	/Km, ↓	/Km, ↓	/Km, ↓	/Km, ↓
DARB	21.12 ± 2	27.8 ± 5	76 ± 3	1.26 ± 0.21	1.92 ± 0.32	0.96 ± 0.11	1.32 ± 0.04	3.26 ± 0.14
CILRS	5.37 ± 2	14.4 ± 2	55 ± 1	2.35 ± 0.25	2.69 ± 0.25	1.48 ± 0.18	1.62 ± 0.08	4.28 ± 0.20
DA_UE	27.9 ± 3	34.8 ± 3	79 ± 1	1.01 ± 0.26	1.62 ± 0.23	0.76 ± 0.12	1.56 ± 0.04	3.02 ± 0.18
DA_CS	18.9 ± 2	26.4 ± 3	71 ± 2	1.36 ± 0.14	1.82 ± 0.34	1.20 ± 0.17	1.29 ± 0.08	3.21 ± 0.21
DA_UE+CS	30.1 ± 2	36.7 ± 4	82 ± 4	0.96 ± 0.17	1.42 ± 0.19	0.72 ± 0.13	0.94 ± 0.05	2.84 ± 0.18
UAIL+	9.77 ± 2	17.4 ± 2	63 ± 1	2.11 ± 0.36	2.42 ± 0.21	1.35 ± 0.17	1.45 ± 0.14	3.27 ± 0.16
UAIL	7.37 ± 2	16.4 ± 2	59 ± 1	2.26 ± 0.26	2.79 ± 0.22	1.43 ± 0.18	1.42 ± 0.08	4.01 ± 0.20

New Town and New Weather. Mean and standard deviation over 3 evaluation seeds. DS: driving score. RC: route completion. IS: infraction score. CP: collisions with pedestrians. CV: collisions with vehicles. CO: collisions with others. RI: red light infraction. AB: agent blocked.

4.3.5. Comparison against UAIL and UAIL+

As shown in Figure 6, UAIL also improved the performance of the model and had some generalization ability to new scenarios. UAIL also used uncertainty prediction, but it used the traditional DAgger structure without data preprocessing or replay buffer, so the final performance improvement was limited. UAIL+ used the MC-dropout algorithm to perform uncertainty prediction in our algorithm structure. The performance was indeed better than UAIL, which also indicates that the improved techniques proposed by researchers in the DAgger algorithm do improve performance. However, the performance of UAIL+ was still not as good as that of DA_UE. As we analyzed before, in the autonomous driving scenario, the MC-dropout method is not as good as the MFVI method for uncertainty prediction, which ultimately leads to poor driving performance of the agent. This also shows that choosing different deep Bayesian algorithms has an impact on the performance of Bayesian DAgger-like algorithms.

4.3.6. Infraction Analysis

Table 3 shows that DA_UE underperforms DARB in the traffic light scenario. Through analyzing the specific data, we found that the Bayesian network is unstable for predicting traffic scenes with traffic lights and cannot make good uncertainty predictions. This indicates that the network's perception of small targets is not good, which leads to insufficient training data extraction for the Bayesian network in this aspect. However, the subsequent critical scenses makes up for this defect and improves the training samples for traffic lights. Finally, the overall performance of the DA_UE+CS algorithm exceeds that of DARB, demonstrating the effectiveness of the proposed method.

5. Conclusions

This study presents a novel DAgger framework that synergistically integrates Bayesian uncertainty estimation via mean field variational inference and critical state identification to improve imitation learning for autonomous driving. The proposed method efficiently approximates the posterior distribution over the policy's parameters and provides well-calibrated uncertainty estimates. During training, the framework identifies both uncertain and critical states, querying the expert only for these states, thereby reducing the burden on the expert and improving data efficiency.

Evaluations on the CARLA simulator using the NoCrash and CARLA leaderboard benchmarks demonstrate that the proposed method outperforms existing imitation learning approaches, such as CILRS, DARB, and UAIL. The results highlight the effectiveness of integrating Bayesian uncertainty estimation and targeted data aggregation in imitation learning for autonomous driving. The proposed approach achieves a higher success rate and lower infraction rate compared to the traditional DAgger algorithm, indicating that uncertainty prediction methods are more purposeful in the data collection process by focusing on states where the learning agent needs improvement.

In conclusion, this study presents a novel and effective approach to imitation learning for autonomous driving by leveraging Bayesian uncertainty estimation and critical state identification. The proposed method improves driving performance, generalization ability, and data efficiency, making it a promising direction for future research in autonomous driving.

Author Contributions: Conceptualization: C.W. and Y.W.; Methodology: C.W.; Software: C.W.; Validation: C.W. and Y.W.; Formal analysis: C.W.; Investigation: C.W.; Resources: C.W.; Data curation: C.W.; Writing—original draft preparation: C.W.; Writing—review & editing: Y.W.; Visualization: C.W.; Supervision: Y.W.; Project administration: Y.W.; Funding acquisition: Y.W.; All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Key Research and Development Program of China (2021YFB2501403).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data in this study are available from the first author upon request.

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

References

1. Codevilla, F.; Miiller, M.; López, A.; Koltun, V.; Dosovitskiy, A. End-to-end driving via conditional imitation learning. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; pp. 1–9.
2. Bojarski, M.; Del Testa, D.; Dworakowski, D.; Firner, B.; Flepp, B.; Goyal, P.; Jackel, L.D.; Monfort, M.; Muller, U.; Zhang, J.; et al. End to end learning for self-driving cars. *arXiv* **2016**, arXiv:1604.07316.
3. Prakash, A.; Chitta, K.; Geiger, A. Multi-modal fusion transformer for end-to-end autonomous driving. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 19–25 June 2021; pp. 7077–7087.
4. Xiao, Y.; Codevilla, F.; Gurram, A.; Urfalioglu, O.; López, A.M. Multimodal end-to-end autonomous driving. *IEEE Trans. Intell. Transp. Syst.* **2020**, *23*, 537–547. [CrossRef]

5. Yang, Z.; Zhang, Y.; Yu, J.; Cai, J.; Luo, J. End-to-end multi-modal multi-task vehicle control for self-driving cars with visual perceptions. In Proceedings of the 2018 24th International Conference on Pattern Recognition (ICPR), Beijing, China, 20–24 August 2018; pp. 2289–2294.
6. Brantley, K.; Sun, W.; Henaff, M. Disagreement-regularized imitation learning. In Proceedings of the International Conference on Learning Representations, New Orleans, LO, USA, 6–9 May 2019.
7. Rajaraman, N.; Yang, L.; Jiao, J.; Ramchandran, K. Toward the fundamental limits of imitation learning. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 2914–2924.
8. Ross, S.; Gordon, G.; Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, Fort Lauderdale, FL, USA, 11–13 April 2011; pp. 627–635.
9. Mullins, G.E.; Dress, A.G.; Stankiewicz, P.G.; Appler, J.D.; Gupta, S.K. Accelerated testing and evaluation of autonomous vehicles via imitation learning. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; pp. 5636–5642.
10. Ross, S.; Bagnell, J.A. Reinforcement and imitation learning via interactive no-regret learning. *arXiv* **2014**, arXiv:1406.5979.
11. Sun, W.; Venkatraman, A.; Gordon, G.J.; Boots, B.; Bagnell, J.A. Deeply aggrevated: Differentiable imitation learning for sequential prediction. In Proceedings of the International Conference on Machine Learning, PMLR, Sydney, NSW, Australia, 6–11 August 2017; pp. 3309–3318.
12. Monfort, M.; Johnson, M.; Oliva, A.; Hofmann, K. Asynchronous data aggregation for training end to end visual control networks. In Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, Sao Paulo, Brazil, 8–12 May 2017; pp. 530–537.
13. Prakash, A.; Behl, A.; Ohn-Bar, E.; Chitta, K.; Geiger, A. Exploring data aggregation in policy learning for vision-based urban autonomous driving. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, DC, USA, 14–19 June 2020; pp. 11763–11773.
14. Zhang, Z.; Liniger, A.; Dai, D.; Yu, F.; Van Gool, L. End-to-end urban driving by imitating a reinforcement learning coach. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Virtual, 11–17 October 2021; pp. 15222–15232.
15. Zhang, J.; Cho, K. Query-efficient imitation learning for end-to-end autonomous driving. *arXiv* **2016**, arXiv:1605.06450.
16. Laskey, M.; Powers, C.; Joshi, R.; Poursohi, A.; Goldberg, K. Learning robust bed making using deep imitation learning with dart. *arXiv* **2017**, arXiv:1711.02525.
17. Menda, K.; Driggs-Campbell, K.; Kochenderfer, M.J. Ensembledagger: A bayesian approach to safe imitation learning. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 3–8 November 2019; pp. 5041–5048.
18. Menda, K.; Driggs-Campbell, K.; Kochenderfer, M.J. Dropoutdagger: A bayesian approach to safe imitation learning. *arXiv* **2017**, arXiv:1709.06166.
19. Gal, Y.; Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Proceedings of the International Conference on Machine Learning, PMLR, New York, NY, USA, 19–24 June 2016; pp. 1050–1059.
20. Cronrath, C.; Jorge, E.; Moberg, J.; Jirstrand, M.; Lennartson, B. BAgger: A Bayesian algorithm for safe and query-efficient imitation learning. In Proceedings of the Machine Learning in Robot Motion Planning—IROS 2018 Workshop, Madrid, Spain, 5 October 2018.
21. Cui, Y.; Isele, D.; Niekum, S.; Fujimura, K. Uncertainty-aware data aggregation for deep imitation learning. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 761–767.
22. Wilson, A.G.; Izmailov, P. Bayesian deep learning and a probabilistic perspective of generalization. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 4697–4708.
23. Michelmore, R.; Wicker, M.; Laurenti, L.; Cardelli, L.; Gal, Y.; Kwiatkowska, M. Uncertainty quantification with statistical guarantees in end-to-end autonomous driving control. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 7344–7350.
24. Blundell, C.; Cornebise, J.; Kavukcuoglu, K.; Wierstra, D. Weight uncertainty in neural network. In Proceedings of the International Conference on Machine Learning, PMLR, Lille, France, 6–11 July 2015; pp. 1613–1622.
25. Jordan, M.I.; Ghahramani, Z.; Jaakkola, T.S.; Saul, L.K. An introduction to variational methods for graphical models. *Mach. Learn.* **1999**, *37*, 183–233. [CrossRef]
26. Dosovitskiy, A.; Ros, G.; Codevilla, F.; Lopez, A.; Koltun, V. CARLA: An Open Urban Driving Simulator. In Proceedings of the 1st Annual Conference on Robot Learning, Mountain View, CA, USA, 13–15 November 2017; pp. 1–16.
27. Codevilla, F.; Santana, E.; López, A.M.; Gaidon, A. Exploring the limitations of behavior cloning for autonomous driving. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 9329–9338.
28. Leaderboard, C. CARLA Leaderboard. 2024. Available online: https://leaderboard.carla.org/ (accessed on 16 March 2024).
29. Hussein, A.; Gaber, M.M.; Elyan, E.; Jayne, C. Imitation learning: A survey of learning methods. *ACM Comput. Surv. (CSUR)* **2017**, *50*, 1–35. [CrossRef]
30. Jackman, S. Estimation and inference via Bayesian simulation: An introduction to Markov chain Monte Carlo. *Am. J. Political Sci.* **2000**, *44*, 375–404. [CrossRef]

31. Zhang, C.; Bütepage, J.; Kjellström, H.; Mandt, S. Advances in variational inference. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *41*, 2008–2026. [CrossRef] [PubMed]
32. Ravindran, R.; Santora, M.J.; Jamali, M.M. Camera, lidar, and radar sensor fusion based on bayesian neural network (clr-bnn). *IEEE Sens. J.* **2022**, *22*, 6964–6974. [CrossRef]
33. Verstraete, D.; Droguett, E.; Modarres, M. A deep adversarial approach based on multi-sensor fusion for semi-supervised remaining useful life prognostics. *Sensors* **2019**, *20*, 176. [CrossRef]
34. Wang, X.; Wang, X.; Mao, S.; Zhang, J.; Periaswamy, S.C.; Patton, J. Indoor radio map construction and localization with deep Gaussian processes. *IEEE Internet Things J.* **2020**, *7*, 11238–11249. [CrossRef]
35. Jiang, Y.; Zhu, B.; Yang, S.; Zhao, J.; Deng, W. Vehicle trajectory prediction considering driver uncertainty and vehicle dynamics based on dynamic bayesian network. *IEEE Trans. Syst. Man Cybern. Syst.* **2022**, *53*, 689–703. [CrossRef]
36. Brechtel, S.; Gindele, T.; Dillmann, R. Probabilistic decision-making under uncertainty for autonomous driving using continuous POMDPs. In Proceedings of the 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), Qingdao, China, 8–11 October 2014; pp. 392–399.
37. Hoel, C.J.; Wolff, K.; Laine, L. Tactical decision-making in autonomous driving by reinforcement learning with uncertainty estimation. In Proceedings of the 2020 IEEE Intelligent Vehicles Symposium (IV), Las Vegas, NV, USA, 19 October–13 November 2020; pp. 1563–1569.
38. Ma, J.; Xie, H.; Song, K.; Liu, H. A bayesian driver agent model for autonomous vehicles system based on knowledge-aware and real-time data. *Sensors* **2021**, *21*, 331. [CrossRef]
39. Minderhoud, M.M.; Bovy, P.H. Extended time-to-collision measures for road traffic safety assessment. *Accid. Anal. Prev.* **2001**, *33*, 89–97. [CrossRef]
40. Van der Horst, A.R.A. *A Time-Based Analysis of Road User Behaviour in Normal and Critical Encounters*; TU Delft Library: Delft, The Netherlands, 1991.
41. Ramezanı-khansarı, E.; Nejad, F.M.; Moogeh, S. Comparing time to collision and time headway as safety criteria. *Pamukkale Üniversitesi Mühendislik Bilim. Derg.* **2020**, *27*, 669–675. [CrossRef]
42. Ovadia, Y.; Fertig, E.; Ren, J.; Nado, Z.; Sculley, D.; Nowozin, S.; Dillon, J.; Lakshminarayanan, B.; Snoek, J. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 1254.