

Ensemble Bayesian Decision Making with Redundant Deep Perceptual Control Policies

Keuntaek Lee, Ziyi Wang, Bogdan Vlahov, Harleen Brar, and Evangelos A. Theodorou

Autonomous Control and Decision Systems Laboratory

Georgia Institute of Technology

Atlanta, GA 30332, USA

keuntaek.lee@gatech.edu

Abstract—This work presents a novel ensemble of Bayesian Neural Networks (BNNs) for control of safety-critical systems. Decision making for safety-critical systems is challenging due to performance requirements with significant consequences in the event of failure. In practice, failure of such systems can be avoided by introducing redundancies of control. Neural Networks (NNs) are generally not used for safety-critical systems as they can behave in unexpected ways in response to novel inputs. In addition, there may not be any indication as to when they will fail. BNNs have been recognized for their ability to produce not only viable outputs but also provide a measure of uncertainty in these outputs. This work combines the knowledge of prediction uncertainty obtained from BNNs and ensemble control for a redundant control methodology. Our technique is applied to an agile autonomous driving task. Multiple BNNs are trained to control a vehicle in an end-to-end fashion on different sensor inputs provided by the system. We show that an individual network is successful in maneuvering around the track but crashes in the presence of unforeseen input noise. Our proposed ensemble of BNNs shows successful task performance even in the event of multiple sensor failures.

Supplementary video: https://youtu.be/poRbH_kB2M

Index Terms—Bayesian Neural Network, End-to-End Control, Ensemble Control, Safety-critical Systems

I. INTRODUCTION

Neural Networks (NNs) are currently one of the most powerful tools for solving difficult problems in decision making such as playing Go [1] or medical diagnosis [2], [3]. Notably, NNs are able to perform rapid and complex computations through relatively simple nonlinear calculations and massive parallel structures. Because of this, NNs have been applied to a variety of difficult classification and regression problems from object detection to robotics. One such task that benefits from the performance of NNs is end-to-end imitation learning for autonomous driving. In this task, difficulty arises in mapping sensor inputs into driving commands [4]. Previous work, such as [4]–[6], shows the successful use of end-to-end imitation learning with applications to autonomous driving and manipulation with visual inputs. Under the imitation learning settings, a system can efficiently learn a task, guided by an expert. However, much of the previous work does not investigate the robustness of the learned end-to-end model to compromised sensors.

This work was supported by Amazon Web Services (AWS) and Komatsu Ltd.

Although NNs are capable of successfully completing difficult tasks in a variety of applications, they are not without drawbacks. One drawback is that it can be nearly impossible to determine what the output of the NN will be given an input without using the NN itself. This is due to the nonlinear computational structure and a large number of parameters. Another drawback is that NNs are also heavily reliant on data; they generally can not make use of prior knowledge such as dynamics models. When confronted with a new input, the output of the network can vary drastically, even if there are similarities to inputs from training data. Even small perturbations to the input can alter the output of Deep NNs [7], [8] and the Deep NNs are easily fooled [9]. This means we do not have a consistent mapping from inputs to outputs. In the context of safety, traditional NNs do not provide a measure of uncertainty of the output.

In recent years, however, new improvements have been made on probabilistic NNs. Bayesian Neural Networks (BNNs) are a probabilistic network structure that produces a distribution of outputs rather than a single output. This output distribution provides valuable information showing how certain or uncertain the output is. With the ability to measure uncertainty, NNs become a viable option for ensemble techniques used for decision making. Ensemble techniques consist of a set of hypotheses from which they choose one as the output. In [10], the ensembles of perturbed models are used to perform robust trajectory optimization with respect to model uncertainty. The work in [11] demonstrated that a simple ensemble model can effectively approximate the predictive uncertainty of Deep Learning (DL) if the objective function obeys a proper scoring rule. This method used multiple NNs with different initializations to serve as individual models of an ensemble for approximating predictive uncertainty. However, the obtained predictive uncertainty was not directly used for improving the performance of the target task.

With knowledge of the uncertainty of each hypothesis, ensemble techniques can be used in safety-critical systems where the failure of the system causes tragic results. In this work, we propose a novel ensemble of end-to-end BNNs to provide an elegant solution to sensor failure in safety-critical systems. Our method is applied to the platform seen in Fig. 1, with the task of agile autonomous driving. With aggressive maneuvers on harsh terrain, sensors can fail from damage

arXiv:1811.12555v3 [cs.RO] 8 Jan 2020

or are unable to operate effectively with rapidly changing conditions.

The rest of this paper is organized as follows: in Section II, we provide background information for the key ideas used in this paper. In Section III we introduce our ensemble BNNs structures and provide the algorithm for decision making. We discuss the expert used for data collection in Section IV and present results in Section V. Finally, we give our conclusions and discuss future work in Section VI.

II. BACKGROUND

In this section, we will cover a few concepts central to our proposed research solution. In order to detect sensor failure, we will be measuring uncertainty in NN models, in comparison to the more traditional approach of sensor fault detection before passing sensor information to a NN. There exists an extensive literature on sensor failure detection [12]–[14] that demonstrates its application in various fields. However, this approach requires knowledge of the expected sensor outputs to determine whether a reading is normal or faulty. Our approach lets the learned model itself address this problem by utilizing the probabilistic counterpart of the traditional NNs, namely BNNs. This Bayesian approach of deep learning removes the need for any beforehand knowledge of expected sensor outputs. A brief overview of types of uncertainty is given to provide the motivation of BNNs. We finish by briefly covering Imitation Learning, which is the method we use to train our models.

A. Aleatoric and Epistemic Uncertainty

Model uncertainty can be classified into two major categories [15]: aleatoric and epistemic uncertainty. Aleatoric uncertainty is a result of the model’s inability to fully describe the environment, while epistemic uncertainty is a result of the inability to acquire unlimited data. In the first case, uncertainty arises when different outcomes are obtained even with the same experimental setup. The source of this type of uncertainty is the hidden variables that can not be perfectly characterized or measured. Epistemic uncertainty arises when the model is presented data not seen previously. The source for this type of uncertainty is a data set that does not fully cover the sample space. In application, it is not possible to completely eliminate either form of uncertainty as we do not have access to a perfect model or unlimited data.

The origin of aleatoric uncertainty suggests that we should be able to train a model to output this type of uncertainty



Fig. 1. *Left*: The 1/5 scaled ground vehicle for autonomous driving and racing. *Right*: The oval track used for experiments.

given data. Meanwhile, we should also be able to measure the epistemic uncertainty of a model through some form of sampling. In this paper, the total predictive uncertainty is calculated to be the combination of both uncertainty types.

B. Bayesian Neural Networks

Currently, there exist two popular methods to obtain a predictive probability distribution in the deep learning literature. The first technique uses Bayesian Backpropagation [16], which assigns a probability prior, usually Gaussian, to the weights in the network. The network is trained by minimizing the Kullback-Leibler (KL) divergence between the distribution on the weights and the true Bayesian posterior distribution.

The second approach uses dropout layers to produce a predictive distribution resulting from a probabilistic network structure [17]. The Monte Carlo dropout approach is adopted in this paper since the alternative approach requires at least doubling the number of parameters in the network, which makes it difficult to run a large scale convolutional Neural Networks with only the computational resources on-board the vehicle. Using the existing NN structure with dropout added to every weight layer, weights in the network are randomly dropped with a certain probability. At every forward pass, we sample a dropout mask from a Bernoulli distribution to determine weights dropped in each layer. During the backward pass, only the remaining weights are updated. The outputs of the network then become a Gaussian Distribution, returning the mean and variance of the prediction values. When trained with the loss function described in Section III-B and Section III-C, these outputs become a combination of aleatoric and epistemic uncertainty. The work in [17] shows the mathematical equivalence between an approximated deep Gaussian process and a NN with arbitrary depth and nonlinearities when dropout layers are applied before and after every weight layer. The output distribution is estimated with Monte Carlo sampling, which can be done in parallel to reduce run-time.

C. Imitation Learning

Imitation Learning (IL), also called “learning from demonstration”, is a type of supervised machine learning. IL is often used when the optimal solution to the task is not easily accessible or too computationally expensive to run in real time. IL algorithms assume that an oracle policy or expert is available. The expert can utilize resources that are unavailable for the imitation learner at test time, such as additional sensory information and computing power. In the case of autonomous driving, the expert can be a sophisticated optimal control algorithm or an experienced human driver. The observation-action or state-action pairs generated by the expert is then used to train the imitation learner. The goal of IL is to mimic the expert’s behavior as well as possible. In [4] IL’s ability to perform the autonomous driving task with low-cost sensors is demonstrated on real-world experiments.

In the traditional formulation, the goal is often to find a policy $\pi : \mathcal{O} \rightarrow \mathcal{U}$ that minimizes an expected loss or

maximizes an expected reward over a discrete finite time horizon H :

$$\min_{\pi} \mathbb{E}_{p_{\pi}} \left[\sum_{t=0}^{H-1} l(x_t, u_t) \right], \quad (1)$$

where $x_t \in \mathcal{S}$, $u_t \in \mathcal{U}$, and $\mathcal{S}, \mathcal{O}, \mathcal{U}$ are the state, observation and admissible control spaces respectively. l is the immediate loss function. p_{π} is the joint distribution of x_t, u_t , and $o_t \in \mathcal{O}$ for the policy π for $t = 0, \dots, H - 1$.

For imitation learning, this equation changes slightly. The goal now becomes to learn a policy that minimizes a loss function that characterizes the difference between the learned model and the expert policy π^* rather than the most optimal π :

$$\pi_{NN} = \arg \min_{\pi} \mathbb{E}_{p_{\pi^*}} [l(x_t, u_t)], \quad (2)$$

where $u_t = \pi(o_t)$ and π_{NN} denotes the neural network policy chosen. In our work, we trained our networks with batch imitation learning.

III. ENSEMBLE BAYESIAN DECISION MAKING

A. Problem Formulation

The main problem considered in this paper is the autonomous driving task for a 1/5 scaled ground vehicle (Fig. 1) using deep neural network-based end-to-end control policies under the sensor failure cases. As mentioned in Section I, many applications of deep end-to-end control do not provide a principle solution to sensor failure. Most self-driving cars today depend on different kinds of sensors including Lidar, Radar, GPS, and cameras. However, in the real world, these sensors are vulnerable to noise. In one example, differential GPS (dGPS) is widely used for autonomous driving to obtain global positions in the world frame. Despite many advances in GPS technology, there is always the probability that the GPS signal jumps or slightly diverges from the true position. In areas with obstacles such as tall buildings or indoor parking lots, GPS tends to fail altogether. Additionally, cameras are sensitive to light conditions and interference from external sources. In autonomous driving, even a slight shift of the GPS data or an obscured camera may cause the car to pass the center line of the road with significant consequences. To avoid these failures, system redundancy is crucial for the safe operation of autonomous driving.

System redundancy is commonly applied in many safety-critical applications, where multiple backup systems exist to prevent catastrophic failure from one faulty component, as shown in [18]. Redundancy is usually achieved by either duplicating the same system or using different systems that perform the same task. It is easy to have duplicative systems available in case of failure, but they are vulnerable to faults of the underlying system. Dissimilar backups, where different hardware, software, and control laws are used in each backup system, can alleviate this problem, but it is hard to determine how the backup systems are prioritized when a failure occurs.

In this work, an ensemble Bayesian decision making process is used to provide system redundancy. Multiple BNNs are implemented on the vehicle, each taking in a different sensory input and having the capability of performing the task on its own. Each BNN is trained end-to-end, learning the low-level control actions from each sensor input. When one or more of the sensors is compromised, the associated predictive uncertainty to the failed sensor should see a significant increase that causes the system to switch to the remaining functional networks.

B. Ensemble Structure

Our ensemble consists of 3 BNNs. Each BNN differs in their network input as well as their network structure. They output the mean, \hat{u} , and the variance, $\hat{\sigma}^2$, of the model caused by the aleatoric uncertainty. The first BNN is trained on the fully-observable state data gathered from GPS module. Its network structure is fully connected with ReLU activation functions and layers of width 1024, 512, 256, and 128, respectively. The second network is trained on images taken from the camera on the left side of the vehicle shown in Fig. 3. It is using the VGG 16 [19]-like network, with modifications to include dropout at each layer as well as output the variance, $\hat{\sigma}^2$ stemming from aleatoric uncertainty [20]. This deep neural network is composed of ~ 30 million trainable parameters, depending on the size of the input. The last network is trained on images taken from the camera on the right side of the vehicle shown in Fig. 3 and has the same network structure as the second network. The overall network structure can be seen in Fig. 2.

To ensure that the outputs of each BNN are the mean and variance due to aleatoric uncertainty, the *heteroscedastic* loss function is used. This loss function used is defined in [15], and is as follows:

$$\mathcal{L}(\pi) = \frac{1}{2\hat{\sigma}^2} \|u^* - \hat{u}\|^2 + \frac{1}{2} \log(\hat{\sigma}^2), \quad (3)$$

where π is the current policy of the network, u^* is the expert's action for input x , \hat{u} is the aleatoric mean for input x , $\hat{\sigma}^2$ is the aleatoric variance for input x . To see how $\hat{\sigma}^2$ is a measure of the aleatoric variance, let us think about how aleatoric variance should behave. We would like that the predictions (\hat{u}) that are close to the expert's output (u^*) – resulting in a low residual error – have low aleatoric variance. Predictions that are far away from the expert's output – resulting in a high residual error – should also have high aleatoric variance. To minimize Eq. (3) when the residual error is high, $\hat{\sigma}^2$ must increase so that the residual error does not have a strong impact on the loss. When the residual error is small, it is observed that $\hat{\sigma}^2$ also needs to be small in order to minimize Eq. (3). Intuitively, since $\hat{\sigma}^2$ follows the increase or decrease of the residual error to obtain a minimal loss, [15] concludes that $\hat{\sigma}^2$ is at least an approximation of the aleatoric variance. In practice, the *heteroscedastic* loss function is modified slightly to:

$$\mathcal{L}(\pi) = \frac{1}{2} \exp(-s) \|u^* - \hat{u}\|^2 + \frac{1}{2} s, \quad (4)$$

where $s = \log(\hat{\sigma}^2)$. By regressing with s , we avoid a potential 'division by 0' error and can still easily calculate $\hat{\sigma}^2$. These

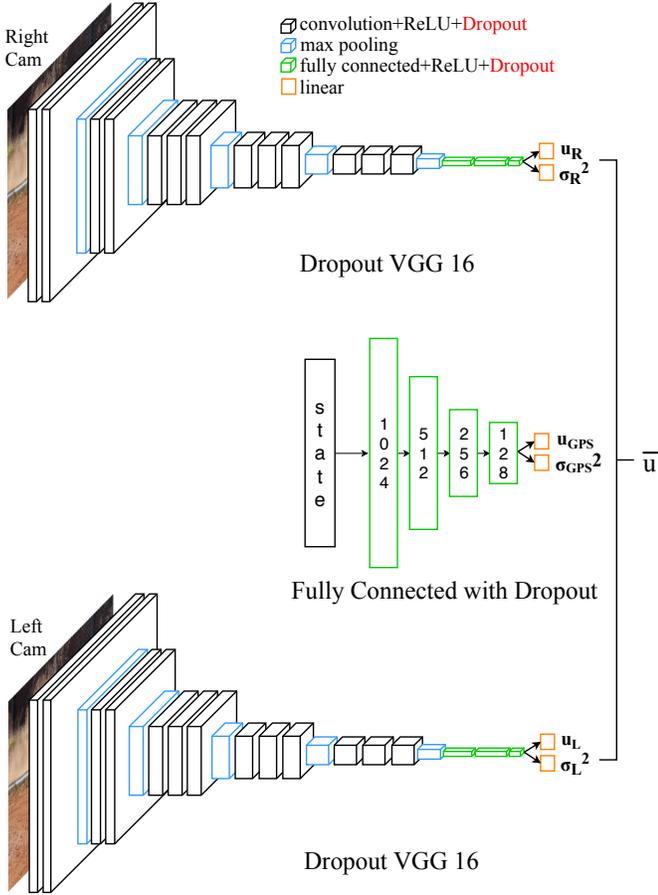


Fig. 2. Ensemble Network Structure composed of end-to-end Bayesian Networks.

means and variances are then used to find the output of the ensemble network as described in the next section.

C. Implementation

To get a better calibrated uncertainty measure, we used Concrete Dropout [21], which allows for automatic tuning of the dropout probability in large models. The output of the redundant system structure is calculated in Algorithm 1. As described in Section II-B, we need to sample our networks multiple times in order to generate the output predictive distribution. In Algorithm 1, instead of conducting multiple runs of the network i on a single input x_i , we duplicate each input n_{MC} times to create an input sequence \mathbf{X}_i , and then input this sequence through the network, where n_{MC} is the number of samples used for Monte Carlo sampling. The two outputs of network i are a vector of control commands $\hat{\mathbf{u}}_i$ and a vector of the aleatoric variances $\hat{\sigma}_i^2$. Using these vector outputs the overall variance (aleatoric and epistemic combined) of each BNN is calculated with the following equation (step 4 in

Algorithm 1 Ensemble Bayesian Neural Networks

Input:

\mathbf{x}_L : Image from left camera;
 \mathbf{x}_R : Image from right camera;
 \mathbf{x}_{GPS} : States from GPS

1: while Task not failed do

2: $\mathbf{X}_L, \mathbf{X}_R, \mathbf{X}_{GPS} \leftarrow$ Duplicate $\mathbf{x}_L, \mathbf{x}_R, \mathbf{x}_{GPS}$

3: $\hat{\mathbf{u}}_L, \hat{\sigma}_L^2 \leftarrow$ DropoutVGG16(\mathbf{X}_L)

$\hat{\mathbf{u}}_R, \hat{\sigma}_R^2 \leftarrow$ DropoutVGG16(\mathbf{X}_R)

$\hat{\mathbf{u}}_{GPS}, \hat{\sigma}_{GPS}^2 \leftarrow$ FC4(\mathbf{X}_{GPS})

4: $u_L, u_R, u_{GPS} \leftarrow$ Mean($\hat{\mathbf{u}}_L, \hat{\mathbf{u}}_R, \hat{\mathbf{u}}_{GPS}$)

$\sigma_L^2, \sigma_R^2, \sigma_{GPS}^2 \leftarrow$ Var($\hat{\mathbf{u}}_L, \hat{\mathbf{u}}_R, \hat{\mathbf{u}}_{GPS}, \hat{\sigma}_L^2, \hat{\sigma}_R^2, \hat{\sigma}_{GPS}^2$)

5: $\bar{u} \leftarrow$ MinVar($u_L, u_R, u_{GPS}, \sigma_L^2, \sigma_R^2, \sigma_{GPS}^2$)

6: end while

Output: \bar{u} : Steering command for the vehicle

Algorithm 1):

$$\sigma_i^2 = Var(u_i) \approx \underbrace{\frac{1}{K} \sum_{k=1}^K \hat{u}_{i_k}^2 - \left(\frac{1}{K} \sum_{k=1}^K \hat{u}_{i_k} \right)^2}_{epistemic} + \underbrace{\frac{1}{K} \sum_{k=1}^K \hat{\sigma}_{i_k}^2}_{aleatoric}, \quad (5)$$

where u_i is the output of network i , \hat{u}_i is the aleatoric mean of network i , $\hat{\sigma}_i^2$ is the aleatoric variance of network i , and $(\hat{\mathbf{u}}_i, \hat{\sigma}_i^2)$ is the set of K sampled outputs from network i . The control of each network i is calculated as the mean of that network's sampled outputs:

$$u_i = \frac{1}{K} \sum_{k=1}^K \hat{u}_{i_k}. \quad (6)$$

Note that, in step 3 in Algorithm 1, the computation (prediction) of all Bayesian Networks happens in parallel.

Finally the output \bar{u} is chosen from the network i with the lowest variance, σ_i^2 , as shown in step 5.

There are two possible ways to do the ensemble Bayesian decision making. One approach is weighting individual network policies with some weights that inversely proportional to their variance. However, this weighting approach is not an optimal solution when the system encounters a multi-modal situation. For example, if one of the network policies tries to drive a vehicle to the left and the other tries to steer it to the right and they have almost equal weights, the ensemble of the policies will guide the vehicle to go straight. This can lead to a tragic result if the network policies made a prediction to drive either left or right and there is an obstacle on the straight.

The other way to do ensemble Bayesian decision making is to pick the best decision according to its confidence, as we proposed in step 5 in Algorithm 1. This approach helps avoid choosing non-optimal control policies in multi-modal decision cases.

IV. DATA COLLECTION

In order to train each learner (BNN) in the ensemble, the iterative Linear Quadratic Gaussian/Model Predictive Control Differential Dynamic Programming (iLQG/MPC-DDP)

[22] algorithm was used as an expert. Differential Dynamic Programming (DDP) is an algorithm that uses second-order approximations of the cost function and system dynamics around a nominal trajectory to solve the Bellman equation. The optimal control policy is then used to update the nominal trajectory. Running DDP in Model Predictive Control (MPC) fashion means that at every timestep, only the first control action is executed by the system, and the control policy is re-optimized at the next timestep when new state information is received. For our long-term autonomous driving task, we used the receding horizon DDP [22].

Using GPS data, the expert had the following state space $[p_x, p_y, \theta, \psi, V_x, V_y, \dot{\theta}]$ as input, where p_x and p_y are global positions in the world frame, θ and ψ are the heading and roll angle of the car, V_x and V_y are the body frame longitudinal and lateral velocities, and $\dot{\theta}$ is the derivative of the heading angle.

We considered the cost function for the optimal controller composed of an arbitrary state-dependent cost and a quadratic control cost. The state-dependent term was designed to stay in the center of the track $(p_{x,des}, p_{y,des})$ while maintaining the desired forward velocity $V_{x,des}$. We set $V_{x,des}$ as 5m/s when we collect data. For the control cost, we used the same weights for both throttle and steering.

The expert drove around an oval track seen in Fig. 1 for 100 laps in one direction to gather data for each learner. As it drove around, the GPS data and truncated 64x128x3 RGB images from the left and right cameras were saved in order to train each of the learner models described in Section III. For training of the Dropout VGG 16 Net [20], we did not use any data augmentation technique (random flips, rotations, contrast, brightness, saturation, jitter, etc.) but we truncated and cropped the original 4k image to reduce the size of it to 64x128x3. All of our models were trained in batch with Tensorflow¹ using the Adam optimizer [23] and the heteroscedastic loss in Eq. (4).

V. EXPERIMENTS/RESULTS

All computation was executed on-board the vehicle with our NVIDIA GeForce GTX 1050 GPU and we were able to obtain 10 Monte Carlo samples ($n_{MC} = 10$) of the ensemble in real time (20Hz). We injected artificial noise signal to each

¹<https://www.tensorflow.org/>



Fig. 3. Platform sensors connected to the on-board computer. *Left*: Two vision cameras, 1280x1024, 70fps, global shutter, synchronously triggered. *Right*: RTK-corrected Hemisphere Eclipse P307 GPS module, position at 20Hz.

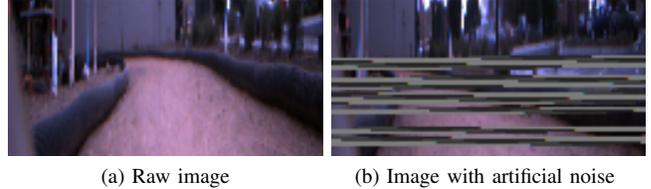


Fig. 4. Artificial noise injected to an input image at test time.

sensor similar to a real-world situation in which a sensor malfunctions. The position noise was sampled from a uniform distribution to make the "new position" appear to be outside of the track. This is commonly seen as GPS data jumps from one location to another. For images, rows of gray bars were added to simulate periodic noise caused by electro-mechanical interference during the image capturing process (Fig. 4b).

First, we tested each BNN in the ensemble network without any artificial noise injected. Each BNN was able to drive the vehicle autonomously until the vehicle's batteries run out and there were no failures. In all experiments, we considered the failure cases as when the vehicle crashes to the boundaries of the track and cannot move forward.

Next, each learner in the ensemble was tested individually on the vehicle with artificial noise injected. After 4 laps of normal operation, noise was added to the corresponding sensor and crashes occurred immediately, as shown in Fig. 5b, Fig. 5c and Fig. 5d. The test was repeated 10 times for each learner and crashes followed promptly after noise injection every time.

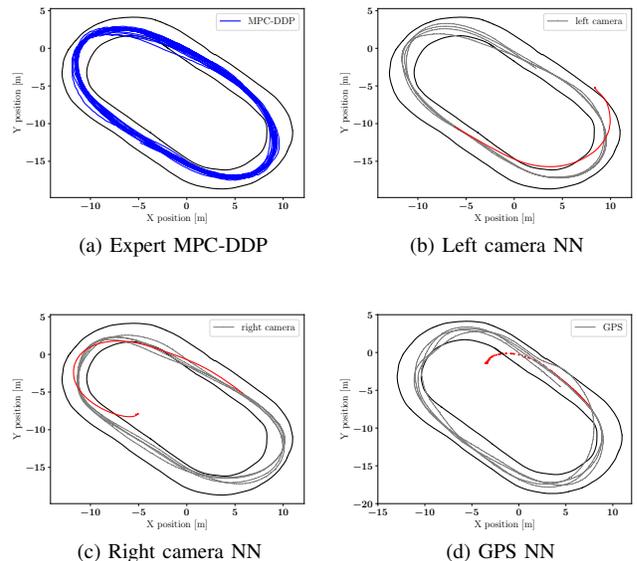


Fig. 5. Trajectory plots of imitation learning of autonomous driving with injected noise. The red trajectories are after the noise injected to each model's input after 4 laps of autonomous driving. We ran 10 experiments for each and all end-to-end learners immediately failed the task.

Following this experiment, the Ensemble Bayesian Neural Networks algorithm was tested without noise injection. The

vehicle achieved similar performance to the expert, as seen in Fig. 5a, and was able to run at a high speed with no crashes.

Finally, the Ensemble BNN algorithm was tested with noise injection. The time horizon for testing was set to be 17 laps. The algorithm was tested on the track 3 times for a total of 51 laps. After 4 laps of normal operation (Fig. 6a), frequent noise was added to each sensor in the order of GPS, left camera, and right camera for 2 laps. Normal operation resumed for another 2 laps before the next noise injection. As we can see from the normal operation case in Fig. 6a, GPS NN was usually used for most of the time. This is because the structure of the GPS NN and the data (7 states) used for it were not complex as the structure of the Dropout VGG 16 network [20] and the data (RGB image) used for it. With a simpler structure and data, it is reasonable to have a smaller variance from the probabilistic network after training. Moreover, without any injection of artificial noise, GPS data at test time does not change much compared to the training data whereas the image from the camera slightly changes due to the change of the lighting conditions and the environment around the vehicle. Fig. 6b shows that when artificial GPS noise was added, the algorithm opted to use camera inputs for navigation as a result of orders of magnitude increase in prediction uncertainty from the fully connected GPS NN. For both normal case and GPS-noise injected case, we observe that the left camera NN was used more often than the right camera NN. We believe this behavior is task-specific, as the vehicle run the oval track in counterclockwise for both data-collecting and testing. Since the left camera is able to see the left track boundary more often than the right camera does, the left camera NN is more confident about its prediction, resulting in smaller variance. Fig. 6c demonstrated a decrease in usage of the left camera input, since image noise caused uncertainty from the corresponding network to double. Similar results can be found in Fig. 6d when image noise was added to both left and right cameras. Compared to the cases where we did not inject any noise (Fig. 6a) or noise was injected in a single camera (Fig. 6c), we can see the decreased usage of both cameras. For all cases of noise injection scenarios, the noise was injected frequently, but not always, so the noise-injected learner could be used intermittently when the noise did not exist. The usage of each learner with sensor noise injection is listed in Table I. In all cases, the usage of the sensor(s) was decreased when the artificial noise was injected to the sensor(s). Even with large noise, which causes the immediate failure of the task for an individual BNN, all laps were completed without any failure. The complete trial can be seen in the video online².

VI. CONCLUSIONS

In this paper, we introduced an Ensemble Bayesian Neural Network structure for system redundancy in the decision making of safety-critical systems. Our algorithm was implemented on an autonomous driving task using end-to-end Imitation

TABLE I
LEARNER USAGE ON EACH LAP

Laps	1-4, 7-8, 11-12	5-6	9-10	13-14
Noise injected in	-	GPS	Left Cam	Both Cams
GPS NN(%)	73.4	22.7	72.5	83.6
Left Cam NN(%)	13.9	42.7	10.0	8.1
Right Cam NN(%)	12.7	34.6	17.5	8.3

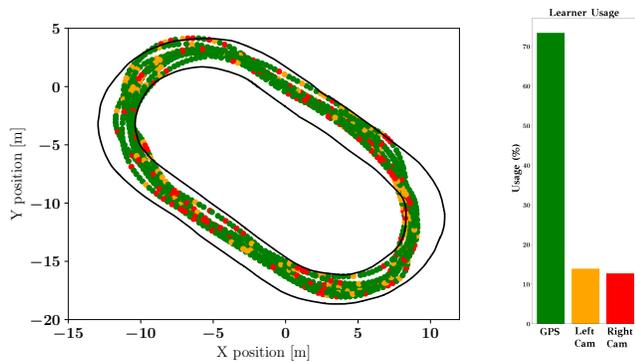
Learning. Prediction uncertainty capturing both model imperfection and data insufficiency of each BNN within the ensemble was used to switch between the different policy outputs. Experimental results verified the robustness of our proposed method against compromised sensor inputs. Our method can play an important role in any kind of autonomous systems using multiple sensors, especially in dealing with safety-critical tasks.

For future works on Ensemble Bayesian decision making, we will further investigate the switching mechanism in the ensemble to ensure safe and stable operation during switching. Furthermore, we will explore smooth Bayesian mixing models as an alternative to our current switching mechanism. Finally, we would like to also extend this Ensemble Bayesian approach to robust filtering and state estimation problems, where we use multiple sensors or networks.

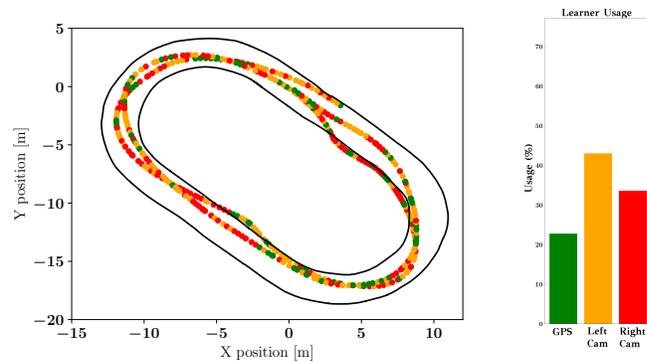
REFERENCES

- [1] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan 2016. [Online]. Available: <http://www.nature.com/articles/nature16961>
- [2] G. Litjens, C. I. Snchez, N. Timofeeva, M. Hermsen, I. Nagtegaal, I. Kovacs, C. Hulsbergen van de Kaa, P. Bult, B. van Ginneken, and J. van der Laak, "Deep learning as a tool for increased accuracy and efficiency of histopathological diagnosis," *Scientific Reports*, 2016. [Online]. Available: <https://doi.org/10.1038/srep26286>
- [3] D. Shen, G. Wu, and H.-I. Suk, "Deep learning in medical image analysis," *Annual Review of Biomedical Engineering*, vol. 19, no. 1, pp. 221–248, 2017, pMID: 28301734. [Online]. Available: <https://doi.org/10.1146/annurev-bioeng-071516-044442>
- [4] Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E. A. Theodorou, and B. Boots, "Agile autonomous driving using end-to-end deep imitation learning," *Robotics: Science and Systems*, 2018. [Online]. Available: <http://www.roboticsproceedings.org/rss14/p56.pdf>
- [5] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, 2016. [Online]. Available: <http://jmlr.org/papers/v17/15-522.html>
- [6] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to End Learning for Self-Driving Cars," apr 2016. [Online]. Available: <https://arxiv.org/abs/1604.07316>
- [7] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *CoRR*, vol. abs/1710.08864, 2017. [Online]. Available: <http://arxiv.org/abs/1710.08864>
- [8] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," *arXiv*, 2017. [Online]. Available: <https://arxiv.org/abs/1702.02284>
- [9] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, 2015. [Online]. Available: <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7298640>

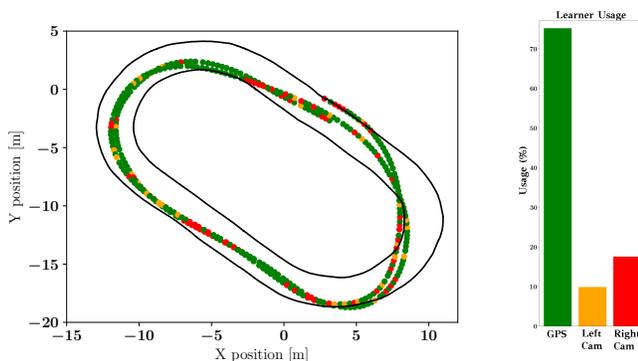
²https://youtu.be/poRbH_kB2M



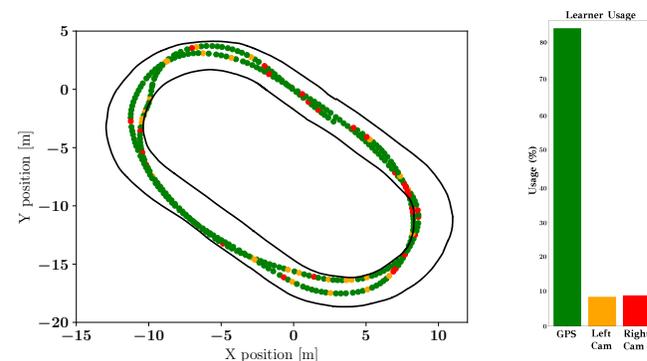
(a) 1-4th, 7-8th, 11-12th, and 15-17th laps of trajectory plots of Ensemble Bayesian decision making without any artificial noise injected.



(b) 5-6th laps with frequent noise injected in position x and y data in the GPS signal.



(c) 9-10th laps with frequent noise injected in the left camera.



(d) 13-14th laps with frequent noise injected in both cameras.

Fig. 6. Each lap is plotted with the colors of the learner whose action has been chosen: **GPS NN**, **Left camera NN** and **Right camera NN**. The algorithm was tested on the track 3 times for a total of 51 laps without failure of the task.

- [10] I. Mordatch, K. Lowrey, and E. Todorov, "Ensemble-cio: Full-body dynamic motion planning that transfers to physical humanoids," *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, 2015. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/7354126/>
- [11] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," *Advances in Neural Information Processing Systems* 30, pp. 6402–6413, 2017. [Online]. Available: <https://arxiv.org/pdf/1612.01474.pdf>
- [12] S. Hussain, M. Mokhtar, and J. M. Howe, "Sensor failure detection, identification, and accommodation using fully connected cascade neural network," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 3, pp. 1683–1692, March 2015.
- [13] R. Isermann, "Process fault detection based on modeling and estimation methods a survey," *automatica*, vol. 20, no. 4, pp. 387–404, 1984.
- [14] S. J. Qin, "Data-driven fault detection and diagnosis for complex industrial processes," *IFAC Proceedings Volumes*, vol. 42, no. 8, pp. 1115–1125, 2009.
- [15] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" in *Advances in Neural Information Processing Systems* 30, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 5574–5584. [Online]. Available: <https://arxiv.org/abs/1703.04977>
- [16] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural network," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 1613–1622. [Online]. Available: <http://proceedings.mlr.press/v37/blundell15.html>
- [17] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 48. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 1050–1059. [Online]. Available: <http://proceedings.mlr.press/v48/gal16.html>
- [18] G. Stein, "Respect the Unstable," *IEEE Control Systems Magazine*, 2003. [Online]. Available: <https://ieeexplore.ieee.org/document/1213600/>
- [19] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *International Conference on Learning Representations (ICLR)*, 2015. [Online]. Available: <http://arxiv.org/abs/1409.1556>
- [20] K. Lee, K. Saigol, and E. A. Theodorou, "Early failure detection of deep end-to-end control policy by reinforcement learning," *2019 IEEE International Conference on Robotics and Automation (ICRA)*, 2019. [Online]. Available: <https://ieeexplore.ieee.org/document/8794189>
- [21] Y. Gal, J. Hron, and A. Kendall, "Concrete dropout," in *Advances in Neural Information Processing Systems* 30, 2017. [Online]. Available: <https://arxiv.org/abs/1705.07832>
- [22] Y. Tassa, T. Erez, and W. D. Smart, "Receding horizon differential dynamic programming," *Advances in Neural Information Processing Systems* 20, pp. 1465–1472, 2008. [Online]. Available: <http://papers.nips.cc/paper/3297-receding-horizon-differential-dynamic-programming.pdf>
- [23] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, vol. abs/1412.6980, 2014. [Online]. Available: <http://arxiv.org/abs/1412.6980>

CITATIONS

Plain Text:

K. Lee, Z. Wang, B. Vlahov, H. Brar, and E. A. Theodorou, "Ensemble Bayesian Decision Making with Redundant Deep Perceptual Control Policies," The 18th IEEE International Conference on Machine Learning and Applications, 2019.

BibTeX:

```
@ARTICLE{lee2019ensemble,  
author={Keuntaek {Lee} and Ziyi {Wang} and Bogdan {Vlahov} and Harleen {Brar} and Evangelos A. {Theodorou}},  
journal={The 18th IEEE International Conference on Machine Learning and Applications},  
title={{Ensemble Bayesian Decision Making with Redundant Deep Perceptual Control Policies}},  
year={2019}  
}
```