Safe Imitation Learning for End-to-End Control

Keuntaek Lee*, Kamil Saigol* and Evangelos A. Theodorou
Georgia Institute of Technology, Atlanta, GA, USA

I. Introduction

As the use of deep neural networks as controllers for robotic systems becomes more prevalent, the issue of safety within artificial intelligence becomes an increasingly important concern. Recently the deployment of end-to-end imitation learning to develop neural network control policies has increased rapidly in popularity due to the ease with which deep models can learn complex dynamics and infer global state from local data while bypassing the need for significant hand-tuning of parameters. However, collecting the large datasets required by these methods is a difficult process, and it is often impossible to collect and process a dataset in a sense that is complete enough to cover every possible scenario.

Hence the challenge is to design a system capable of taking an appropriate action given novel data at test time. The ability to effectively handle novel data is key to the introduction of safety in intelligent control.

Bayesian neural network models may be used to produce an uncertainty estimate along with a prediction. Corrective action, such as a return of control to the optimal controller or human expert, can then be taken when the uncertainty associated with a prediction exceeds a threshold. We propose an algorithm that combines reinforcement learning and end-to-end imitation learning to simultaneously learn a control policy as well as a threshold over the predictive uncertainty of the learned model, with no hand-tuning required. We validate our method on fully-observable and vision-based partially-observable systems on cart-pole and autonomous driving simulations using deep convolutional Bayesian neural networks. We demonstrate that our method is robust to uncertainty resulting from varying system dynamics as well as from partial state observability.

II. Background

A. Epistemic and Aleatoric Uncertainty

Uncertainty in machine learning models can come from two sources: incomplete data and incomplete knowledge of the environment [6]. The former, called *epistemic uncertainty* is the result of a data set that is either not large enough or does not cover enough of the "search space". Such uncertainty could be explained away by access to unlimited data and can be reduced by the collection of additional data. The latter source, called *aleatoric uncertainty*, is the result of our inability to completely sense all aspects of the environment. While such uncertainty could not be explained away with access to more data – or even with access to unlimited data – it could be explained away with access to unlimited sensing.

*Equal contribution.

Given data, we should be able to train a model to output this type of uncertainty as well. In principle, the total predictive uncertainty would be the combination of the two.

B. Bayesian Approximation with Dropout

The dropout technique, usually used to mitigate overfitting effects, can be used at test time to approximate Bayesian neural networks [4]. Using the dropout approximation, the same test-time input will result in a different output each time the input is passed through the network. Several samples of the output corresponding to the same input result in a sampling distribution that can be used to estimate epistemic uncertainty.

Aleatoric uncertainty is not learned via sampling, but instead learned by using the so-called *heteroscedastic loss* function and splitting the output layer of the network such that one channel is trained to output a predictive mean and the other the aleatoric uncertainty [6]. By combining the uncertainty obtained via sampling with that obtained during training, an accurate representation of total model uncertainty is obtained.

III. Reinforcement Learning of the Uncertainty Threshold

We propose learning autonomous vehicle control end-to-end using Bayesian neural network models with the dropout technique. In order to enhance system safety, however, our algorithm returns control to an expert controller (for example, a human operator or complex model predictive controller) when the predictive uncertainty generated by the network exceeds a threshold. In earlier works, similar thresholds are manually selected [7]; here, we propose selecting the threshold in a principled manner with reinforcement learning (RL). In particular, we use the Cross-Entropy Method (CEM) [3] applied to RL to learn the optimal uncertainty threshold as summarized in Algorithm 1.

Our results demonstrate that our framework converges to a threshold that allows the learned model to perform the task while still allowing the expert enough time to recover from or handle unusual situations significantly different from the learned model's training set.

IV. Experiments

Our expert controller in all tasks considered herein is a real-time MPC controller which applies the Differential Dynamic Programming (DDP) trajectory optimization algorithm in a receding horizon fashion. In all cases, the MPC-DDP expert has access to full state information.

We validate our framework for safe learning of model predictive control on two different tasks including cart-pole swing-up and autonomous driving with obstacle avoidance.

Algorithm 1: Cross-Entropy Method to optimize switching uncertainty threshold

```
Input  : $x$ : initial state, $n_{RL}$, $n_{ro}$ : episodes & rollouts per episode, $T$ : maximum time
         steps for task completion, $\mu$, $\sigma^2$ : initial distribution of the threshold, $r$, $\phi$ :
         immediate & terminal reward functions, $\underline{b}$, $\bar{b}$ : min & max number of candidates,
         $B$ : set of $\bar{b}$ best candidate solutions, $\pi_L$, $\pi_E$ : Learner(NN) and Expert policies
Output: $\mu^*$ : optimized mean value of the threshold,
         $R_{sum}$ : sum of rewards
1  for $i \leftarrow 1$ to $n_{RL}$ do
2      Reset the environment and $x$
3      $success \leftarrow 0$
4      Obtain $n_{ro}$ samples $X = [X(1), \ldots, X(n_{ro})] \sim \mathcal{N}(\mu, \sigma^2)$
5      for $j \leftarrow 1$ to $n_{ro}$ do
6          $R_j \leftarrow 0$,  actor $\leftarrow \pi_L$
7          Execute actions from $\pi_L$ until an adversarial input occurs at time step $t'$
8          for $t \leftarrow t'$ to $T - 1$ do
9              Execute actor action
10             actor $\leftarrow \pi_E$ if uncertainty $> X(j)$
11             $R_j \leftarrow R_j + r(t)$
12         end
13         $R_j \leftarrow R_j + \phi(T)$
14         Increment $success$ if $\phi(T) \neq 0$
15         sort $X$ by descending $R_j$
16         $B \leftarrow X[1 : \bar{b}]$
17         $(\mu, \sigma^2) \leftarrow (mean(B), var(B))$
18     end
19     Repeat from Step 3 if $success < \underline{b}$
20     $R_{sum}(i) \leftarrow \Sigma_{j=1}^{n_{ro}} R_j$
21 end
22 return $\mu$, $R_{sum}$
```

For vision-based tasks, we introduce the DropoutVGG Network. This Bayesian extension to VGG Net [8] uses the dropout technique as described above. In addition, we split the output layer and train the network to output aleatoric uncertainty.

For all tasks, we provide positive terminal reward in the CEM algorithm if the task was completed successfully by the Bayesian network model or by the controller. We provide zero terminal reward for RL episodes in which the task fails. In order to encourage use of the learned model, we provide a positive immediate reward for every time step for which the learned model is used. Otherwise, we provide zero immediate reward.

A. Cart-Pole Swing-Up

For the cart-pole swing-up task, the goal is to swing up and balance the pole only by applying a horizontal force to the cart. The terminal reward is typical of that used in OpenAI Gym simulator [2].

1) Fully Observable Case: Both the expert and the Bayesian neural network have perfect information regarding the complete system state $[x, \dot{x}, \theta, \dot{\theta}]^T$. In this case, we use a simple feedforward Bayesian network. After training the neural network with the expert, we allow the learned model to attempt the task. The mass of the cart is changed from 1.0 to 0.2 at an arbitrarily selected time to create a disturbance which the network has not been trained to handle.

2) Partially Observable Case: In the partially observable case, the neural network can estimate x and θ but velocities $\dot{x}$ and $\dot{\theta}$ are not directly accessible. DropoutVGG is trained with inputs consisting of (third-person view) images of the cart-pole system (Fig. 1) and observations of the expert control. In this case, a disturbance is created when the input image is randomly corrupted with the addition of a new object that was not present in the training data.

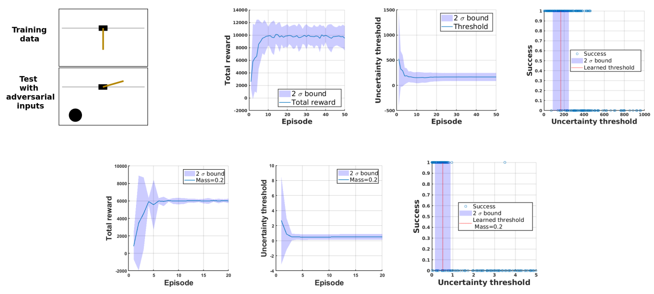

Fig. 1: Results of learning a switching threshold for a cart-pole swing-up task. Top: partially observable case only using images. Bottom: fully observable case. The rightmost plots show success/failure trials per sample of the uncertainty threshold where we see that learned optimal thresholds are located between the success and failure margins.

B. Autonomous Navigation and Obstacle Avoidance

We demonstrate the success of our framework on the autonomous navigation and obstacle avoidance task using the simulator provided by the AutoRally project [1][5]. This software simulates driving a 1/5th scale autonomous vehicle on a track and is capable of providing simulated color images from onboard cameras. DropoutVGG is trained with observations consisting images from the vehicle's onboard camera (Fig. 2) as well as the corresponding steering angle produced by the MPC-DDP expert as it navigates around the track. The goal is to train the network to learn to steer the vehicle. The learned model only access the camera image. As usual, no disturbances are given to the neural network in the training phase.

When a previously unseen obstacle comes into view, the predictive uncertainty increases significantly, indicating potential failure of the learned model. Our method successfully trains the learned model to pass control back to the expert in time to avoid the obstacle.

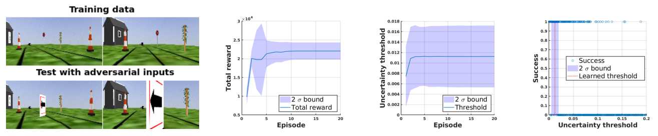

Fig. 2: Results of learning a switching threshold for an autonomous navigation and obstacle avoidance task in partially observable case, only using images(leftmost).

V. Conclusion

We have presented herein a fully end-to-end framework for safe learning of control. Our approach successfully trains a Bayesian neural network to perform complex vision-based MPC with access to a dataset containing image observations and corresponding controls from an expert. This approach allows the learned model to effectively know when it will be unable to safely complete the task and return control to an expert capable of doing so. Our framework requires no manual parameter tuning; instead we propose the use of a reinforcement learning method to learn uncertainty thresholds in a principled manner. We validate our method on cart-pole swing-up and autonomous driving tasks.

References

[1] Autorally. http://autorally.github.io/, 2016.

[2] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016. URL https://arxiv.org/abs/1606.01540.

[3] Pieter-Tjerk de Boer, Dirk P. Kroese, Shie Mannor, and Reuven Y. Rubinstein. A Tutorial on the Cross-Entropy Method. *Annals of Operations Research*, 134(1):19–67, feb 2005. ISSN 0254-5330. doi: 10.1007/s10479-005-5724-z. URL http://link.springer.com/10.1007/s10479-005-5724-z.

[4] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA, 20–22 Jun 2016. PMLR. URL http://proceedings.mlr.press/v48/gal16.html.

[5] Brian Goldfain, Paul Drews, Changxi You, Matthew Barulic, Orlin Velev, Panagiotis Tsiotras, and James M. Rehg. Autorally an open platform for aggressive autonomous driving. 06 2018. URL https://arxiv.org/pdf/1806.00678.pdf.

[6] Alex Kendall and Yarin Gal. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? mar 2017. URL http://arxiv.org/abs/1703.04977.

[7] Kunal Menda, Katherine Driggs-Campbell, and Mykel J. Kochenderfer. Dropoutdagger: A bayesian approach to safe imitation learning. 09 2017. URL https://arxiv.org/abs/1709.06166.

[8] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL http://arxiv.org/abs/1409.1556.