Learning Steering Bounds for Parallel Autonomous Systems

Learning Steering Bounds for Parallel Autonomous Systems

Alexander Amini[1], Liam Paull[2], Thomas Balch[1], Sertac Karaman[3], Daniela Rus[1]

***Abstract*— Deep learning has been successfully applied to "end-to-end" learning of the autonomous driving task, where a deep neural network learns to predict steering control commands from camera data input. While these previous works support reactionary control, the representation learned is not usable for higher-level decision making required for autonomous navigation. This paper tackles the problem of learning a representation to predict a continuous control *probability distribution*, and thus steering control options and bounds for those options, which can be used for autonomous navigation.. Each mode of the distribution encodes a possible macro-action that the system could execute at that instant, and the covariances of the modes place bounds on safe steering control values. Our approach has the added advantage of being trained on unlabeled data collected from inexpensive cameras. The deep neural network based algorithm generates a probability distribution over the space of steering angles, from which we leverage Variational Bayesian methods to extract a mixture model and compute the different possible actions in the environment. A bound, which the autonomous vehicle must respect in our parallel autonomy setting, is then computed for each of these actions. We evaluate our approach on a challenging dataset containing a wide variety of driving conditions, and show that our algorithm is capable of parameterizing Gaussian Mixture Models for possible actions, and extract steering bounds with a mean error of only 2 degrees. Additionally, we demonstrate our system working on a full scale autonomous vehicle and evaluate its ability to successful handle various different parallel autonomy situations.**

I. Introduction

Recently, deep learning models have demonstrated the ability to directly learn control outputs from raw sensor data [1]–[4]. These types of architectures are referred to as "end-to-end" (sensor to actuation) and have yielded surprisingly promising results for reactionary control, such as effective autonomous lane following [4], navigation in closed environments [2], and full-scale autonomous driving [1], [3], [5].

However, there are still several issues with these approaches for autonomous driving. Firstly, end-to-end trained steering control systems offer only a single most likely control output as evidenced in the training data. Secondly, existing approaches do not reliably handle scenarios in which the car is faced with ambiguous situations where there may be multiple correct actions (intersection). Thirdly, autonomous and parallel autonomy systems require effective interplay between reactionary controls, navigation, path planning, and decision making capabilities. It is unclear how outputs of recently proposed end-to-end trained steering controls could be effectively integrated with higher level planning and navigation systems. For example, since these regression deep neural networks (DNNs) are trained end-to-end as black-boxes, they lack a definitive measure of associated *confidence* with the output and consequently cannot be integrated into a shared control (parallel autonomy) framework.

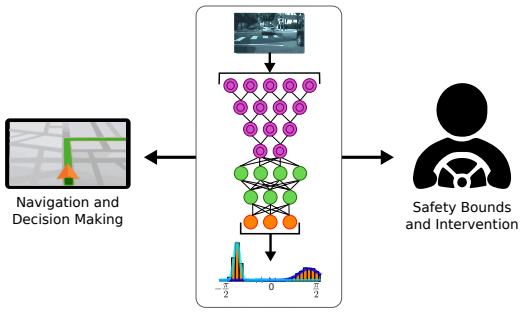

Fig. 1. **End-to-end deep learning of control distributions**. The output of the classification network encodes a distribution of control values, rather than a single value in previous end-to-end approaches [1]. Consequently, we can use the output for navigation and decision making. Furthermore, we can use our approach as part of a "parallel autonomy" shared control system in which sets of upper and lower bounds for steering trajectories can be computed and used as controller takeover points.

Parallel autonomy is a human-robot shared control paradigm whereby the human is in control of the vehicle, but the autonomy system always runs in the background and is responsible for preventing the human from causing an accident [6]. The goal is to produce inputs that minimally deviate from the human input while guaranteeing safety, even in complex driving scenarios. This framework is appealing since it can be integrated into an existing car before the full autonomous car problem is solved and still have a positive impact in terms of safety. However, for this to be possible, the autonomy system must output a degree of certainty related to its control output. This is essential for the system to "do no harm" since in the worst case of uncertainty in the autonomy system it can defer to the human driver.

Existing approaches to shared control of autonomous vehicles are model-based [7]–[18] and as such tend to have either high computational requirements rendering them inapplicable for all but the simplest of driving scenarios [7], [8], [16], [18], or make strong simplifying assumptions, such as constant velocity of obstacles [10] or availability of a perfect road model [11].

In this paper, we propose a new approach to end-to-end learning for autonomous driving that learns *steering control distributions* from raw image data, rather than single regressed values. A key advantage of learning control dis-

Support for this work was given by the Toyota Research Institute (TRI). However, note that this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the DGX-1 used for this research, as well as Berthold Horn and Ava Soleimany for their valuable comments.

[1] Computer Science and Artificial Intelligence Lab, Massachusetts Institute of Technology {amini,rus}@mit.edu

[2] Department of Computer Science and Operations Research, Université de Montréal {paulll}@iro.umontreal.ca

[3] Laboratory for Information and Decision Systems, Massachusetts Institute of Technology {sertac}@mit.edu

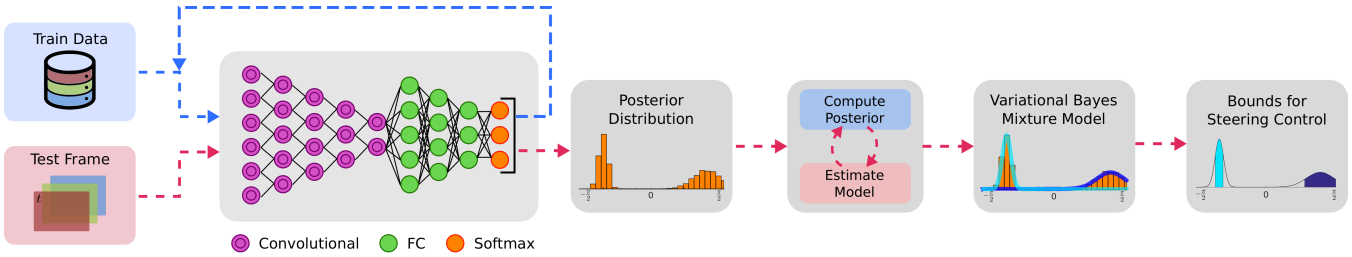

Fig. 2. **System framework overview**. A schematic outline of the image-to-control framework proposed in this paper, composed of a deep convolutional neural network (trained and tested with single image inputs), a Variational Bayesian mixture model, and a safety bound extraction step. The colors of the network layers are for illustration with the neuron function indicated below the architecture.

tributions, rather than a deterministic control value, is that we are able to account for scenarios where there are multiple possible directions and actions, thus enabling integration into other autonomous controllers which also consider navigation and parallel autonomy. We start by discretizing our action space of steering control values into small bins, allowing us to handle different possible outputs and ambiguous situations. This discrete output is then transformed into continuous probability distributions by fitting to a Gaussian mixture using Variational Bayesian methods. Consequently, we are able to provide an analytical continuous expression for the posterior probability and thus extract sets of upper and lower bounds for steering, where each set of bounds defines a control region surrounding a possible action. We evaluate our approach on real driving data collected with a full scale autonomous car. The contributions of this paper can be summarized as follows:

1) A novel representation of steering control outputs that provides control options necessary for navigation, path planning, and decision making capabilities;
2) A neural network architecture and integration with Variational Bayes mixture model and steering bound extraction that provides a measure of associated *confidence* with the output, and consequently can be effectively integrated into a shared control framework to serve as guidance to a human driver, and is trained entirely with unlabeled data;
3) A real world demonstration of the utility of the proposed approach in a parallel autonomy setting; and
4) Experimental validation on a real, full-scale autonomous vehicle and a challenging dataset with different road and weather conditions, times of day, and seasons.

The remainder of the paper is structured as follows: we summarize the related work in Sec. II, we formulate the problem and describe our proposed method in Sec. III, and provide an outline of our experimental setup and results in Sec. IV.

II. Related Works

Typical approaches to autonomous driving decompose the problem into standard components such as mapping and localization [19], [20], perception and scene understanding [21]–[23], planning [24], [25], and control [11], [26], with an individual algorithm applied to each submodule.

The concept of collapsing the entire problem into a single learning system (end-to-end) originated with the autonomous land vehicle in a neural network (ALVINN) system [27], which used pixel image inputs to train a multilayer perceptron network to output the direction the vehicle should travel. Recent advancements in computing and deep learning have brought about a resurgence in the end-to-end methods for autonomous vehicle control [1], [5], showing their enormous potential by driving a real-world vehicle through highways and city streets. The system in [3] uses crowd-sourced data to train a fully-convolutional network and applies the output to a long short-term memory recurrent neural network for generating a driving policy. Many studies have trained networks for subsets of the full autonomous driving task, such as [4], where the authors are able to estimate the vehicle's position within a lane without explicitly detecting the lane.

Recent work has also demonstrated end-to-end navigation for other application domains other than autonomous driving, such as the problem of navigating through similar-type maze environments with a prior map and a severely restricted field of view [2]. Additionally, imitation based approaches [28]–[30] attempt to imitate experts, but in the context of training driver models, suffer from cascading errors and frequent oscillation between actions [31]. Additionally, other NN based approaches for end-to-end driving either output a single control value [1] or are very limited in terms of the range of control outputs that are possible (e.g. only allow front, back, left, right) [2]. In both cases, it is not possible to integrate these outputs into either higher level navigation and decision making systems or a shared control paradigm which requires a stochastic interpretation of the output.

Interpreting the confidence of the output of machine learning algorithms has also been explored in great detail. Optimization of DNNs has been formulated as a maximum likelihood problem using a softmax activation layer on the output to estimate probabilities of discrete classes [32] as well as discrete probability distributions [2]. Introspective capacity interprets the distance in feature space between a train and test distribution as an algorithm's uncertainty and has been used to evaluate model performance for a variety of commonly used classification and detection tasks [33]. Bayesian deep learning provides yet another way to estimate confidence through Monte Carlo dropout sampling of weights in recurrent [34] and convolutional neural networks [35], and has been applied for semantic driving scene segmentation [36]. However, all of these approaches are

limited by their discreteness and are unable to handle their distributions analytically in the continuous domain. Additionally, due to the loss function used, they often yield noisy output distributions making it difficult to extract reliable bounds and thresholds. While it is possible to avoid this problem by training a model that directly outputs parameterized continuous mixture models [37], these approaches require ground truth knowledge of all mixture components for every data sample and do not allow the number of output mixtures to dynamically change over time. Requiring the ground truth distributions at every instant would require significantly more data labeling to manually identify the underlying mean and variance for every macro action in the scene. Our approach overcomes these shortcomings by learning a discrete probabilistic output which is then transformed into continuous probability distributions for control.

III. Learning Steering Distributions

In this section we present a framework for computing steering distributions and control bounds using only a single image frame as input. This framework is composed of a deep neural network to compute a discrete probability distribution over the space of steering angles and Variational Bayesian methods to extract a continuous mixture model and compute the different possible actions in the environment. A bound, which the autonomous vehicle must respect in our parallel autonomy setting, is then computed for each of these actions. Figure 2 illustrates the pipeline from training (blue arrows) to testing and inference (red arrows). In the following subsections we will detail each of the individual components of the pipeline.

A. Learning a Steering Probability Distribution

We use a time invariant approach to predict a control signal distribution for the vehicle. Only a current frame, without any history, is fed into a deep convolutional neural network (CNN), with the steering command used to backpropagate errors back through the network. The network computes the *probability distribution* of possible controls, binned into small, discrete groups.

We define m discrete bins for steering sampled from a tunable logit function projected onto the x-axis on the interval $[-1, 1]$. Having the ability to tune our discretization affords more flexibility on how angles are binned over the space of all turning angles. Since the vast majority of driving data has steering angles within a small interval around the equilibrium of steering, having a discretization where bin angles are concentrated near the center of steering enables more precise classification when going straight or making small turns, while receiving more spread on larger turning angles.

We define the m bin edges of steering as:

$$\beta_j = \left(\frac{\pi}{2}\right) \frac{v_j(1-\gamma)}{\gamma - 2\gamma|v| + 1}$$

where v is an evenly spaced vector, of length m, from -1 to 1, and $\gamma \in (-1, 1)$ is a single tunable parameter which affects the skewness of the discretization. For example, setting $\gamma = 0$ yields $\beta_j = \frac{\pi}{2} v_j$, which is a linear function of bins (evenly spaced bins). On the other hand, raising the magnitude of γ increases the skew of the discretization towards either the equilibrium or the extremes. Specifically, $\gamma > 0$ results in a discretization concentrated on small angles, whereas $\gamma < 0$ concentrates discretization on large turning angles. Unless stated otherwise, for the remainder of this paper, $m = 15$, $\gamma = 0.7$, since these parameter choices yield a very precise discretization at angles near zero, while giving larger bin sizes to larger magnitude angles. More details on why this bin selection was chosen is covered in Section IV.

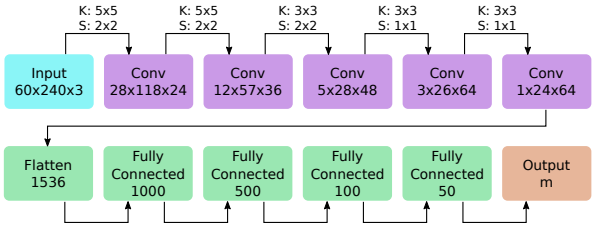

Fig. 3. **Convolutional Neural Network architecture**. Schematic representation of our CNN model with the size of each layer and type of each layer indicated within the respective rectangle. Kernel (K) and stride (S) sizes are also indicated between convolutional layers.

We trained a single CNN to predict the probability of steering distributions, given a single image frame, using stochastic gradient descent (SGD). To predict the full discretized distribution we optimize the cross entropy loss between the predicted distribution and the ground truth steering angle distribution defined below. Given a set of input images $X = \{x^{(1)}, \ldots, x^{(n)}\}$ and true steering outputs $Y = \{y^{(1)}, \ldots, y^{(n)}\}$, we define a neural network, N, with weights w capable of estimating steering outputs $\hat{y}^{(i)} = \mathsf{N}(x^{(i)}; w)$ for a specific input example. The loss function is defined as:

$$L(\hat{y}, y) = \frac{1}{n} \sum_{i=1}^{n} H(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{m} \hat{y}_j^{(i)} \log(y_j^{(i)})$$

where $H(\hat{y}^{(i)}, y^{(i)})$ is the cross entropy for training data i. Figure 3 outlines the architecture of the proposed probabilistic neural network. The first 10 layers used the network proposed in [1] as inspiration with a different input size. Our network can be split into two pieces where each piece is composed of (1) convolutional layers to generate feature maps and (2) fully connected layers to perform the classification and output the probability distribution.

We express ground truth steering as 1-of-m encoding vectors (as opposed to a single real-valued number). In such a scheme, at a given time instant the bin containing the true steering angle is given a probability of 1, while all other bins have a probability of 0. Final activations are passed through a normalizing softmax function to compute the probability distribution. More formally, one can express the probability that the vehicle should be steered within any one of the m bins, β_j, given input image x and a neural network N with weights w as:

$$\begin{aligned} P(\hat{y} \in \beta_j) &= \delta_j(\hat{y} \in \beta_j) \\ &= \frac{\exp(\hat{y} \in \beta_j)}{\sum_{l'=1}^{m} \exp(\hat{y} \in \beta_{l'})} \end{aligned}$$

where δ_j is the softmax function for estimating the probability of bin j.

Now that we have the entire discretized distribution of predicted steering trajectories, one could deduce a control signal using a variety of techniques. One simple example would be to take the maximum likelihood action. In other words, we could define the predicted steering command signal, $\hat{\theta}$, as $\arg\max_{\beta_j}\{P(\hat{y} \in \beta_j)\}$. While this is one simple estimate of the final predicted steering angle given a full probability distribution, we introduce a more robust technique to extract both valid control signal modes and steering bounds.

B. Variational Bayesian Mixture Models

Under normal driving conditions, an operator will often encounter states where multiple actions are appropriate. For example, if a vehicle is at an intersection where it is only possible to either turn left or right then we would like the probability distribution to be bimodal: one peak representing a right turn and the other representing a left turn. The magnitude of the peak would correspond to the statistical likelihood of the turn in each case. From this simple example we can see that in a state with A actions, we would expect to see a mixture of approximately A random variables centered at the specific turning angle for that action.

We formulate this problem as follows. Given a discrete probability distribution of steering angles, can we (1) infer the number of possible actions that can be executed, (2) compute a control signal for any of these actions, and (3) determine bounds of steering to properly execute any of these actions. If we assume each action can be approximated by a Gaussian random variable, then it is possible to sample our discrete distribution and fit a Variational Bayesian mixture model (VBMM) [38].

As opposed to performing a hard clustering (such as k-means), VBMM is a generative model that allows for soft probability classifications, wherein a single steering angle could have been generated by any of the A Gaussians with some probabilistic weighting. This is critical in our system since there may be certain steering angles in the space that correspond to multiple different actions in the environment. Furthermore, VBMM can be seen as an extension of the Expectation Maximization (EM) algorithm which also approximates data according to a model depending on unobserved latent variables. Perhaps most importantly, VBMM, unlike EM and k-means, does not require the number of mixture components, A, in the model to be specified before fitting. Rather, it automatically determines the number of mixtures throughout its optimization, iteratively shrinking mixtures to zero if they are not well fit within the data. For example, we start the algorithm with A_0 mixtures, for $A_0 > A$ (i.e., strictly greater than the true number of mixtures).

Figure 4 shows an example of the Variational Bayesian algorithm running on a probability distribution extracted while driving. From the distribution we can see that there are three potential actions that the driver can take. The algorithm starts out with $A_0 = 10$ mixtures placed throughout the data, and as it iterates some of these shrink (both in terms of weight and standard deviation) while other mixtures get stronger. Eventually, the algorithm converges to a combination of $A = 3$ Gaussians after about 200 steps.

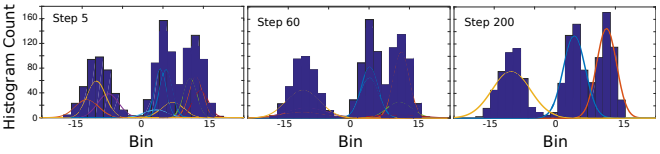

Fig. 4. **Variational Bayesian mixture model fitting**. Output of the Variational Bayesian algorithm running on a sample probability output extracted while driving and converging to the true number of mixtures ($A = 3$), causing some mixtures to die over time and others to progressively get stronger.

Through this process, the algorithm automatically determines the ideal number of potential actions which could be executed in the current state of the vehicle, and assigns each action its own generative Gaussian random variable. The extraction of each of these possible steering actions will be used to extract bounds for steering (as described presently).

C. Steering Control Bounds for Parallel Autonomy

The Variational Bayesian mixture model (VBMM) fits the data to a mixture of Normal-Wishart distributions, whose PDF can be sampled by a Gaussian random variable centered at the mean. More concretely, if the output of a VBMM has component k with strength α_k, mean $\bar{\mu}_k$, precision matrix Λ_k, and support v_k, we can sample θ from a Gaussian of the form $\mathcal{N}(\theta; \mu_k, \sigma_k)$ with

$$\phi_k = \frac{\alpha_k}{\sum_{k'=1}^{A} \alpha_{k'}}; \qquad \mu_k = \bar{\mu}_k; \qquad \sigma_k = (v\Lambda_k)^{-1}$$

Furthermore, we can determine the probability that any steering angle, θ, belongs to the kth mixture (or action) and use this to deduce which mixture contributes to it the most as

$$P(\theta; \mu_k, \sigma_k) = \phi_k \mathcal{N}(\theta; \mu_k, \sigma_k)$$
$$\hat{k} = \arg\max_k (\mathcal{N}(\theta; \mu_k, \sigma_k))$$

where ϕ_k represents the mixing proportion of action k.

Now that we have a full Gaussian mixture model (GMM) we can define the set of accepted steering angles as the set union of each bounded Gaussian, in other words:

$$S(\omega) = \left\{ \theta : \theta \in \bigcup_{k=1}^{A} [\mu_k - \omega\sigma_k, \mu_k + \omega\sigma_k] \right\}$$

where ω is a tunable constant representing the number of standard deviations away from the mean of each mixture. Now we are able to define a simple parallel autonomy controller which tries to keep a human operator within these bounds of steering $S(\omega)$. In other words, given a desired human steering angle, θ_H, which is implicitly computed from the torque applied on the steering wheel, we can determine the parallel autonomy control angle, θ_{PA}, as:

$$\theta_{PA} = \begin{cases} \theta_H & \theta_H \in S(\omega) \\ \frac{1}{2}(\theta_H + \theta_A) & \text{otherwise} \end{cases}$$

where θ_A, the predicted autonomous steering angle, is the angle within the predicted bounds which is closest to the

human controlled angle. More specifically, we solve θ_A as an optimization problem over the space of all predicted bounds:

$$\theta_A = \arg\min_{\theta} \left(S(\omega) - \theta_H\right)^2$$

where this difference in human and autonomous angle represents the least drastic change in control in order to re-enter the predicted bounds.

From this control policy we can see that as long as the human is executing "safe" commands (i.e. operating within the bounds of steering) the computer should not intervene. However, if the human was to execute a command outside these bounds, the autonomous controller should then apply an extra torque to the steering wheel, gently guiding the human back towards safe ranges of steering.

IV. Results

In this section, we briefly describe our dataset collection environment and provide a summary of results in inferring steering control bounds from raw pixel data collected from the on-vehicle camera.

A. Experimental Setup

The vehicle base platform used for this study is a Toyota Prius 2015 V, which was retrofitted with sensors, power, and computing systems for parallel and autonomous driving [6]. A forward facing PointGrey Grasshoper 3 camera [39], which captures RGB images at approximately 20Hz, is the vision data source for this study. Sensors also collected steering wheel angle and encoder clicks, which are used to infer speed. The dataset was collected largely in the Boston metropolitan area across a variety of driving scenarios, including urban, suburban, and highway environments, and at varying times of the day and week. The dataset, approximately 7 hours (500 GB) of driving data in total, was split into training and testing portions with some of the training dataset used for validation. The video is sampled at 10Hz to expunge consecutive frames which are too similar to each other. Additionally, to filter out points where the car is not moving, we only consider frames when the vehicle is moving faster than 2 miles per hour.

An NVIDIA DGX-1 supercomputer was used for training and validation of the DNNs. While training, a random mini-batch of size 20 was randomly aggregated and fed through the network. Training frames were sampled from our dataset such that an approximately equal number of examples from each bin were fed through the network, to reduce bias towards any particular steering direction. To validate our results we periodically evaluated our model with a pre-extracted validation set. We tested and validated our model on a section of the dataset that does not overlap with the training dataset. Our model ran for approximately 20 epochs before convergence.

All results were obtained using a binning of $k = 15$ and $\gamma = 0.7$, which produced minimum bin sizes of 2.5 degrees at the center, and maximum bin sizes of 40 degrees at the extreme steering angles, $-\frac{\pi}{2}$ and $\frac{\pi}{2}$. This binning scheme was selected to produce tighter (i.e., more precise) bounds where turning angles were smallest, and allow larger bounds where turning angles were larger (i.e., less precise).

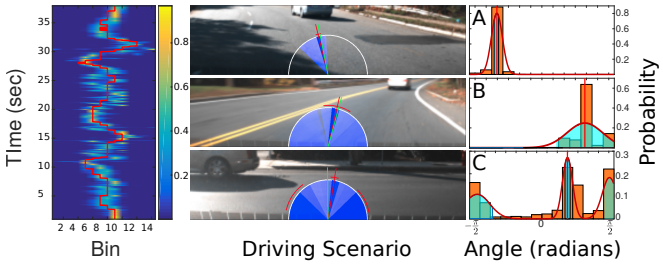

Fig. 5. **Sample inference over a test set**. The figure on the left illustrates the temporal evolution of the predicted steering probability distributions (over a 40 second sample). The heat-map represents the intensity of the probabilities while the human executed steering wheel angle data is overlaid as a red line. The figures on the center and right outline three common driving scenarios (A: left turn, B: right turn, C: ambiguous intersection) and the probability distribution overlaid on the steering wheel (center). Gaussian mixtures and bounds (right) of one standard deviation are overlaid on the posterior distribution.

In addition to offline testing or our algorithm, we also implemented our entire codebase to run onboard a laptop controlling the car using drive-by-wire. This includes running the entire inference and posterior estimation in real time (15Hz) to compute an autonomous steering command, as well as the corresponding bounds. We use an Intel i5 CPU with 4 cores to perform our computation, and in the future, plan to also install an NVIDIA Drive PX2 inside of our vehicle to increase neural network inference speeds even more.

B. Temporal Evolution of Steering Bounds

Figure 5 (left) illustrates actual human steering control inputs (red plot) over a 40 second period, overlaid on the predicted steering bounds, and shows the human inputs operating within the predicted bounds. Moreover, confidence is high (yellow) when bounds are tight, and lower (teal) when bounds are loose. The high confidence regions coincide well with inflection points in human control. For example, near $t = 0$, rightward steering with tight bounds is predicted and corresponds with the true steering action taken. At $t = 10$, the bounds reflect initiating leftward steering control up to $t = 12$, at which time the left turn execution completes, and the steering control returns to steering angle near 0 as the driver straightens out.

Intuitively, Figure 5 also illustrates the impact of bin size on the tightness of bounds and confidence in those bounds. Specifically, near $\beta = 0$ (straight-on steering control), the confidence in making what would amount to minor adjustments (due to the small bin size) is lower and the bounds are more loose. Additionally, larger adjustments to execute left turns (i.e., β approaches $-\frac{\pi}{2}$) and right turns (i.e., β approaches $\frac{\pi}{2}$), result in more significant adjustments to steering control, with higher confidence.

The center right section of Figure 5 provides illustrative results for 3 commonly encountered driving scenarios. For each scenario, the camera image on the left depicts the data provided as input to the inference pipeline in Figure 2, and the Gaussian Mixture and Bounds plot on the right depicts the output produced by the pipeline for the corresponding image. The x-axis is the turning angle (i.e., steering control)

and y-axis provides the probability for that steering control angle. Each image on the left includes an overlay (bottom, center), which is a projection of the turning angle onto a semi-circle, where the green line shows the predicted steering control angle, and the red line shows the actual steering control angle taken by the driver. Within a single shaded arc, the darkness of the shading represents the probability that the car should steer in that direction, with dark blue indicating the highest probability turning angles and lighter shading indicating lower probability turning angles. The red arc hovering slightly above the semi-circle indicates the steering bounds for control.

For example, Scenario A is an example of the vehicle approaching another vehicle positioned slightly to its right, with room for passing on the left. The steering control indicates a leftward turning angle, with tighter bounds on the right than on the left, as illustrated by the semi-circle overlay. Note that the Gaussian Mixture and bounds also indicate leftward steering control. In comparison, Scenario B is an example of the vehicle following another vehicle at a distance. Here, because the preceding vehicle is in far view, the bounds are more loose, as illustrated by both the wider arc on the overlaid semicircle and the Gaussian mixture on the right. Finally, Scenario C shows 3 turning ranges at approximately $[-\frac{\pi}{2}, -\frac{\pi}{4}]$, $[0, \frac{\pi}{8}]$, and $[\frac{\pi}{4}, \frac{\pi}{2}]$, which represent a) an immediate left turn, b) continuing straight-on and slightly to the right, or c) an immediate right turn.

Figure 5 helps to illustrate why learning bounds, as opposed to simply predicting a steering control angle, provide important advantages for parallel autonomy. For example, in autonomous mode, the predicted steering control angle can be used directly for control, or navigation can be optimized with the computed bounds; in human control mode, the human can be alerted when the vehicle steering angle is approaching the computed bounds; in parallel autonomy mode, the autonomy system can bring the vehicle to within computed bounds when human steering inputs to not conform to computed bounds.

C. Accuracy of Steering Control Bounds

We assessed the accuracy of our steering control bounds technique by aggregating the normalized probability distribution of a true steering angle predicted over the space of all possible steering control bins in Figure 6A. This plot is shown on the logarithmic scale to better visualize the small differences between probabilities.

The bright green regions along the diagonal illustrate good alignment between the predicted bin and the bin selected by the driver, and a high confidence in that prediction. We also observe in Figure 6A that the off-diagonal regions, especially at predicted bin 6-10 (where the steering angle is closer to 0) show dispersion across bins. The challenge with this measure of accuracy is that in most cases, there is no single correct steering control. That is, measuring the accuracy of a neural network by comparing the predicted control to the control executed by a human driver (or some statistical average of control executed by human drivers) is valid when only if there is "gold standard" or uniquely correct action to be compared against. While commonly used to assess neural

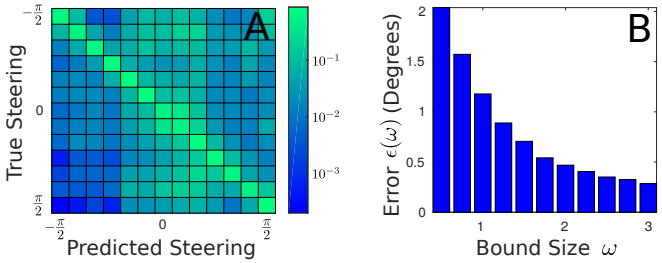

Fig. 6. **Numerical results**. A) The output of the DNN represented as a probabilistic confusion matrix. B) The error evaluated as the mean distance a human control signal is away from the bounds of steering computed by our algorithm as a function of the size of the bound.

network based driving algorithms [5], in the absence of a gold standard, this technique does not actually measure the accuracy of the controller to be safe, rather, it measures the similarity between the controller and human drivers. Simulator based approaches [1] provide a more realistic sense of accuracy since the model is only penalized when it deviates far enough that a human driver must intervene. However, even in this case, the simulator is still constrained to driving along the same general path of the human who collected the data, making it an unsuitable metric for our algorithm which allows for multiple different actions within the road (ex. an intersection).

Instead, we define a measure of accuracy of our system as the percent of time that a human driver stays within an upper and lower bound, which the algorithm computes via Variational Bayesian methods. Defining accuracy in this sense means that a controller which operates within the expected bounds not be penalized (even if it might be different than one particular human driver). Additionally, this allows us to account for the different possible actions that arise when driving.

More formally, as long as the human driver is within any of the possible computed bounds the model prediction is deemed valid, otherwise it is penalized according to the Mahalanobis distance, which measures the distance between a point and a distribution. The point in question is the human steering command, θ, while the distribution is characterized by $(\mu_{\hat{k}}, \sigma_{\hat{k}})$ with $\hat{k}$ defined in Sec. III-C. Putting this all together we can define the error of our model to stay within the bounds of our ground truth human driving according to:

$$\epsilon(\omega) = \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{1}_{S(\omega)}(\theta)\left(\theta^{(i)} - \mu_{\hat{k}}^{(i)}\right)^2 \sigma_{\hat{k}}^{(i)}\right)$$

where ω, like defined in Sec. III-C, is a tunable parameter representing the number of standard deviations over which the steering bound extends and $\mathbf{1}_{S(\omega)}(\theta)$ is the mathematical indicator function which is 1 when $\theta \notin S(\omega)$ and 0 otherwise. Figure 6B shows the error of our system as a function of ω. Starting with a bound of $\pm\frac{\sigma}{4}$ on the far left x-axis, which is approximately the variation we observe from human drivers, we obtain a mean error of approximately 2 degrees. Increasing ω, and thus increasing the bounds, causes our error to drop roughly exponentially. This is precisely what we would expect to see since increasing the bounds means the model is able to encapsulate more and more steering angles within each action. We test up until $\omega = 3$ (i.e. ± 3

Fig. 7. **Scenes from our test environment**. The lack of lane markers and extensive shadows, cracks, and vegetation make autonomous driving particularly challenging. Images are taken from a camera mounted on the roof rack of our autonomous vehicle.

standard deviations from the mean), and observe a decrease in error to only about $\frac{1}{4}$ degrees.

Overall, these results (i.e., mean error of 2 degrees at a bound of $\pm\frac{\sigma}{4}$) are extremely promising; however, we acknowledge that with only 7 hours of driving data it is unlikely that our model was able to generalize well to every complex driving scenario. To provide a more robust test environment we also implemented our algorithms onboard our own autonomous vehicle to evaluate its performance in a real-time testbed.

D. Physical Experimentation

In this section, we outline an experiment we conducted to evaluate our system's ability to predict bounds of steering control in a real-time physical testbed. All experimentation was conducted on a test track, spanning about 2 km in length. This test track contains no lane markers, many large cracks, vegetation growing into the road, as well as huge amounts of variation in luminosity (as illustrated in Figure 7). All of these factors make it a difficult and meaningful test environment which captures a large number of possible corner cases that one may encounter during driving.

In order to quantitatively understand the performance of our predicted bounds and control signal we setup an experiment while running under "parallel autonomy" mode as defined in section III-C. We started by setting up 10 evenly spaced locations on our test track and observing the human machine interaction with two different types of driver:

1) An "ideal" driver who adheres as close as possible to the correct behavior of steering at that location; and
2) An "erratic" driver who tries to execute a dangerous maneuver (such as driving off the road) at that location.

We ran this experiment by driving (speed $\sim$25 miles per hour) around our track as the two drivers, ten times each. We measured the performance of our system to accurately predict control steering bounds by its ability to intervene and prevent the human from performing any erratic action and its ability to not intervene while a human is executing proper control.

Figure 8 shows the results for this experiment under both types of human behavior. The track is mapped out for illustration using a 3D laser scanner with all points projected onto the 2D ground plane and drawn in grayscale in the figure. The top map shows the fraction of times which the

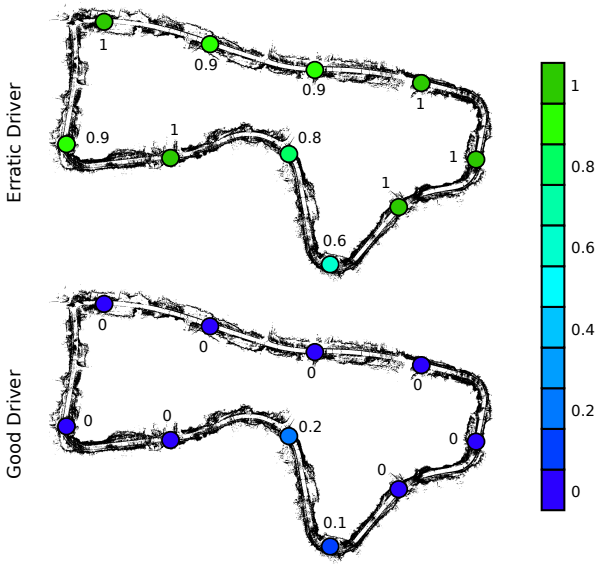

Fig. 8. **Parallel autonomy interventions under different scenarios**. The fraction of times that the controller intervened over ten loops around our test track (mapped in black). At each point, the driver was instructed to act in one of two ways (1) execute proper control (bottom) or (2) try to drive off the road (top). The color of each dot represents the fraction at which these interventions occurred at that given location.

human intervened the ideal human driver at each of the 10 locations around the track. The respective color of each point corresponds to this fraction. Averaging over all the locations and runs, we see that our algorithm successfully determined that the ideal human was operating within the bounds of steering (97%) of the time and incorrectly intervened on a very small number of trials.

Since ω, the number of standard deviations away from the mean of each mixture, is a tunable constant we can arbitrarily increase this accuracy even more. By defining ω to be some very large number, we essentially say that the entire space of steering commands are acceptable and the computer should never intervene. Of course, this is not the desired behavior since we clearly want to intervene when a driver acts improperly as well. Therefore, we are faced with a pair of constraints on ω, where we wish to minimize the number of interventions while the human is acting properly, but maximize the number of interventions when the driver is not behaving well. The bottom map of Figure 8 shows the number of interventions for the erratic driver; averaging again over all trials, we observed a successful intervention rate of 91%. Together with the ideal driver, we observed an average successful behavior of the system 94% of the time. There are several reasons for the differences in performance between the two types of driver. Most importantly, we observed that the largest discrepancies came on tight turns, where the room for error is naturally smaller than on straight or more gentle curves. Since our model uses a constant ω for all steering wheel angles we are unable to capture this, and as a result are unable to recover from minor errors made on these tight turns. We plan to address this in greater detail in the future.

V. Conclusion

This paper presents a novel deep learning based algorithm for autonomous driving that computes an intermediate prob-

ability distribution of steering angles. While previous work in end-to-end learning presents a form of reactionary control, lane following, and object avoidance, this technique encodes a much richer set of information in a probability distribution, thereby making it an attractive algorithm for incorporating navigation and decision making capabilities. Our results indicate the ability to dynamically compute the number of potential actions at any point in time and to accurately extract steering bounds from each of these mixtures with a mean deviation error of approximately 2 degrees.

We demonstrate our algorithms working on an autonomous vehicle testbed operating in real-time, both to autonomously control the car as well as to provide parallel autonomous control, successfully intervening when a human makes dangerous decisions. Overall, our algorithm maintained ideal behavior throughout this parallel autonomy paradigm for 94% of the test evaluation.

In the future, we aim to incorporate higher level decision making capabilities into our pipeline, specifically targeting more complex turn-by-turn navigation. Further, we seek to improve the human-computer interface within our parallel autonomy platform, particularly under situations of shared human-robot control.

REFERENCES

[1] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[2] J. Zhang, J. T. Springenberg, J. Boedecker, and W. Burgard, "Deep reinforcement learning with successor features for navigation across similar environments," *CoRR*, vol. abs/1612.05533, 2016.

[3] J. Zhang and K. Cho, "Query-efficient imitation learning for end-to-end autonomous driving," *CoRR*, vol. abs/1605.06450, 2016.

[4] A. Gurghian, T. Koduri, S. V. Bailur, K. J. Carey, and V. N. Murali, "Deeplanes: End-to-end lane position estimation using deep neural networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, June 2016, pp. 38–45.

[5] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," *arXiv preprint arXiv:1612.01079*, 2016.

[6] F. Naser, D. Dorhout, S. Proulx, S. D. Pendleton, H. Andersen, W. Schwarting, L. Paull, J. Alonso-Mora, M. H. Ang Jr., S. Karaman, R. Tedrake, J. Leonard, and D. Rus, "A parallel autonomy research platform," in *IEEE Intelligent Vehicles Symposium (IV)*, June 2017.

[7] T. Fraichard and H. Asama, "Inevitable collision states. a step towards safer robots?" in *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, 2003.

[8] M. Forghani, J. M. McNew, D. Hoehener, and D. D. Vecchio, "Design of driver-assist systems under probabilistic safety specifications near stop signs," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 1, pp. 43–53, 2016.

[9] D. Hoehener, G. Huang, and D. D. Vecchio, "Design of a lane departure driver-assist system under safety specifications," 2016.

[10] J. Alonso-Mora, P. Gohl, S. Watson, R. Siegwart, and P. Beardsley, "Shared control of autonomous vehicles based on velocity space optimization," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 1639–1645.

[11] W. Schwarting, J. Alonso-Mora, L. Paull, S. Karaman, and D. Rus, "Parallel autonomy in automated vehicles: Trajectory generation with real-time obstacle avoidance and human input optimization," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017.

[12] S. J. Anderson, S. B. Karumanchi, K. Iagnemma, and J. M. Walker, "The intelligent copilot: A constraint-based approach to shared-adaptive control of ground vehicles," *Intelligent Transportation Systems Magazine, IEEE*, 2013.

[13] S. J. Anderson, S. B. Karumanchi, and K. Iagnemma, "Constraint-based planning and control for safe, semi-autonomous operation of vehicles," in *Intelligent Vehicles Symposium (IV), 2012 IEEE*. IEEE, 2012.

[14] S. M. Erlien, "Shared Vehicle Control using Safe Driving Envelopes for Obstacle Avoidance and Stability," Ph.D. dissertation, STANFORD UNIVERSITY, 2015.

[15] S. M. Erlien, S. Fujita, and J. C. Gerdes, "Shared steering control using safe envelopes for obstacle avoidance and vehicle stability," *IEEE Transactions on Intelligent Transportation Systems*, 2016.

[16] R. Verma and D. D. Vecchio, "Semiautonomous multivehicle safety," *IEEE Robotics Automation Magazine*, 2011.

[17] Y. Gao, A. Gray, A. Carvalho, H. E. Tseng, and F. Borrelli, "Robust nonlinear predictive control for semiautonomous ground vehicles," in *2014 American Control Conference*. IEEE, 2014, pp. 4913–4918.

[18] H. Ahn, A. Rizzi, A. Colombo, and D. D. Vecchio, "Robust supervisors for intersection collision avoidance in the presence of uncontrolled vehicles," *CoRR*, vol. abs/1603.03916, 2016.

[19] J. Levinson and S. Thrun, "Robust vehicle localization in urban environments using probabilistic maps," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 2010.

[20] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," *Artificial intelligence*, vol. 128, no. 1-2, pp. 99–141, 2001.

[21] A. Amini, B. Horn, and A. Edelman, "Accelerated convolutions for efficient multi-scale time to contact computation in julia," *arXiv preprint arXiv:1612.08825*, 2016.

[22] A. A. Assidiq, O. O. Khalifa, M. R. Islam, and S. Khan, "Real time lane detection for autonomous vehicles," in *Computer and Communication Engineering, 2008. ICCCE 2008. International Conference on*. IEEE, 2008, pp. 82–88.

[23] Z. Kim, "Robust lane detection and tracking in challenging scenarios," *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 1, pp. 16–26, 2008.

[24] U. Lee, S. Yoon, H. Shim, P. Vasseur, and C. Demonceaux, "Local path planning in a complex environment for self-driving car," in *Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), 2014 IEEE 4th Annual International Conference on*, 2014.

[25] C. Richter, W. Vega-Brown, and N. Roy, "Bayesian learning for safe high-speed navigation in unknown environments," in *Proceedings of the International Symposium on Robotics Research (ISRR 2015), Sestri Levante, Italy*, 2015.

[26] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, "Predictive active steering control for autonomous vehicle systems," *IEEE Transactions on control systems technology*, 2007.

[27] D. A. Pomerleau, "Alvinn, an autonomous land vehicle in a neural network," Carnegie Mellon University, Computer Science Department, Tech. Rep., 1989.

[28] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems*, 2016.

[29] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 1.

[30] G. Neu and C. Szepesvári, "Apprenticeship learning using inverse reinforcement learning and gradient methods," *arXiv preprint arXiv:1206.5264*, 2012.

[31] A. Kuefler, J. Morton, T. Wheeler, and M. Kochenderfer, "Imitating driver behavior with generative adversarial networks," *arXiv preprint arXiv:1701.06699*, 2017.

[32] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[33] H. Grimmett, R. Triebel, R. Paul, and I. Posner, "Introspective classification for robot perception," *The International Journal of Robotics Research*, vol. 35, no. 7, pp. 743–762, 2016.

[34] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *Advances in neural information processing systems*, 2016, pp. 1019–1027.

[35] ——, "Bayesian convolutional neural networks with bernoulli approximate variational inference," *arXiv preprint arXiv:1506.02158*, 2015.

[36] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" *arXiv preprint arXiv:1703.04977*, 2017.

[37] C. M. Bishop, "Mixture density networks," 1994.

[38] D. M. Blei, M. I. Jordan, *et al.*, "Variational inference for dirichlet process mixtures," *Bayesian analysis*, vol. 1, no. 1, pp. 121–143, 2006.

[39] *Grasshopper 3*, PointGrey, uSB3 Vision.