2019 IEEE Intelligent Transportation Systems Conference (ITSC)
Auckland, NZ, October 27-30, 2019

Calibrating Uncertainty Models for Steering Angle Estimation

Christian Hubschneider*[1], Robin Hutmacher*[2†] and J. Marius Zöllner[1]

***Abstract*— Various approaches to end-to-end vehicle control using deep neural networks have been proposed recently, examining various architectures to predict steering angles based on raw sensor data. However, most of these approaches are only used as black boxes, which work well in most scenarios and drive vehicles in real traffic, but it is unclear when they will fail. In order to use such models in larger architectures used in autonomous vehicles, they need to reason about decisions or at least provide an additional measure of confidence that captures the uncertainty of the model. In this paper, we introduce and motivate different uncertainty models, comparing Monte Carlo dropout with network architectures based on the bootstrap ensembling method and a Gaussian mixture for the task of end-to-end vehicle control. Furthermore, we evaluate the presented uncertainty models regarding their driving performance as well as their model uncertainty calibration. The model calibration can be regarded as a measure of how well an uncertainty estimates fits to the expected performance. Well calibrated uncertainty estimates are crucial when embedding deep learning models into probabilistic models.**

I. INTRODUCTION

So far a lot of autonomous driving architectures consist of rule-based and probabilistic modules, i.e. every possible scenario has a determined decision outcome that can be verified and explained. With more complex scenarios and recent advances in deep learning and its application to autonomous driving, this paradigm is shifting. Deep learning systems, however, are usually not very transparent in their decisions and behave more like black boxes. Even if predictions are correct in most of the cases, failures are fatal.

This is especially true in end-to-end steering models that use raw sensor data input to a neural network architecture to predict a vehicle's steering angles, as was shown by various groups [1], [2], [3]. In such models, predictions consist of a single point estimate, e.g. the steering angle. One step towards a safer system and better understanding is an additional output concerning the models confidence in its decision. Such an output could be used to identify difficult situations and move steering dependency to the driver or in the case of multi-sensor systems, rely on other sensor data. When multiple information sources, such as camera, lidar and sensors are fused together, it is necessary for each of them to estimate their uncertainty in order to take a reasoned decision [4]. This is a hard task because there are no explicit labels usable in a supervised learning process.

[1] FZI Research Center for Information Technology, Karlsruhe, Germany. hubschneider, zoellner@fzi.de
[2] Bosch Center for Artificial Intelligence, Renningen, Germany. robin.hutmacher@de.bosch.com
*These authors contributed equally. †Work done while at FZI.

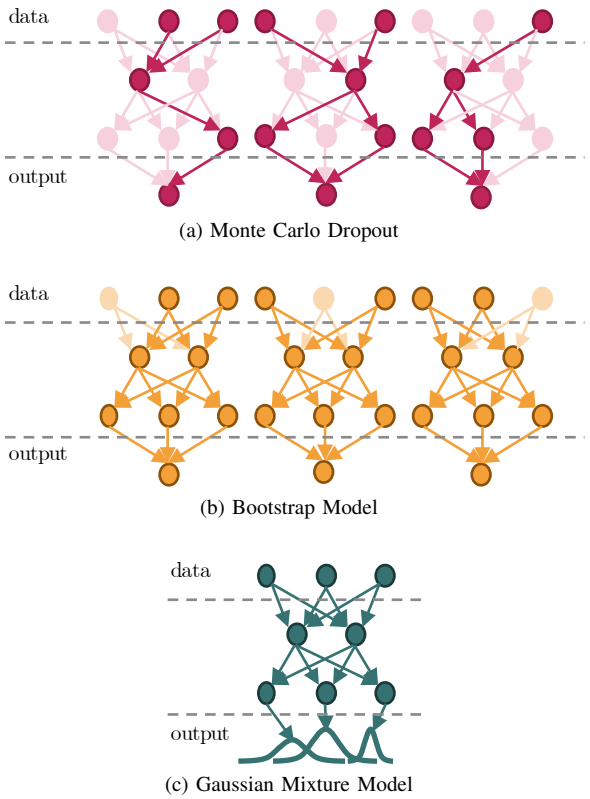

Fig. 1. Schematic comparison of the different uncertainty models, comparing Monte Carlo dropout with network architectures based on the bootstrap ensembling method and a Gaussian mixture for the task of end-to-end vehicle control.

In statistics and the computer vision and machine learning area, the terms aleatoric and epistemic are used to describe different types and sources of uncertainties [5]. Epistemic uncertainty is commonly referred to as model uncertainty and is caused by gaps in the input data distribution. Aleatoric uncertainty is also called sample uncertainty because of stochasticity inherent in the data or data source and can be viewed as data noise that can not be explained away by more data. This is reflected in a high entropy likelihood distribution. Aleatoric uncertainty can further be separated into homoscedastic and heteroscedastic. Homoscedasticity mean constant noise across all samples while heteroscedasticity describes noise that may vary across samples.

In our case of steering angle prediction, we expect high aleatoric uncertainty in situations where the steering angle is not distinct, i.e. multiple future paths and thus steering angles are possible and the training data contained multiple possibilities for a given situation. We expect epistemic uncertainty

978-1-5386-7024-8/19/$31.00 ©2019 IEEE

to relate to unknown driving situations such as completely different lightning situations, strong rain, snow, etc. that were not contained in the training data set. Distinguishing between the two uncertainty types is helpful as they potentially require different treatment. Epistemic scenarios are dangerous because the model has never encountered similar situations. Aleatoric situations might be less harmful, if the model can be forced to choose an option among multiple routes. When we are not able to separate the two uncertainties or want to refer to the total uncertainty in a situation, we will use the term predictive uncertainty.

In the domain of steering angle prediction, uncertainty will manifest as variance over steering angles. Instead of working with variances σ^2, we will use the standard deviation $\sigma = \sqrt{\sigma^2}$ (std), as it is easier to interpret since it is expressed in the same unit as the steering angle.

The steering angle prediction task could also be approached as a classification problem where steering angles are discretized into bins and normalized by a final softmax layer. However, following the argument of Gal in his thesis and previous work [6], [7], the output of a softmax layer should not be interpreted as probability output over the classes. Gal argues that, in the case of adversarial examples, the softmax outputs a very high probability for a wrong class but also constructs a simple binary classification problem. More recently Guo et al. also showed that the softmax outputs are not well calibrated, i.e. they can not be interpreted as true probabilities [8].

In this work, we present three variants of uncertainty models for end-to-end vehicle control. On overview of these models, namely a model using Monte Carlo dropout for uncertainty estimation, a bootstrap ensembling model as well as a Gaussian mixture model, can be found in Fig. 1. All models are compared regarding driving performance, as well as their respective model uncertainty estimation.

The remainder of the paper is structured as follows: an overview of related work is given in Sec. II, the different uncertainty model architectures are introduced in Sec. III, followed by the specification of evaluation metrics in Sec. IV. Evaluation results are discussed in Sec. V and we provide concluding remarks in Sec. VI.

II. Related Work

End-to-end steering models used for autonomous vehicle control were first reintroduced by Bojarski et al. [1], successfully demonstrating a car being controlled by a convolutional neural network (CNN). The idea of using neural networks to control a car, however, is not new: the concept was first proposed back in 1989 by Dean Pomerleau [9]. The baseline model used in this work to implement uncertainty models for steering angle prediction is in turn based on [2] and [10], extending the model by a spatial history to demonstrate successful navigation in urban streets. The general idea behind this type of end-to-end trained models is called *Imitation Learning*: a model is trained using human driving data and is tasked to imitate the human's performance.

There have also been reinforcement learning approaches learning how to drive from scratch. The major advantage of those methods is that they do not require extensive training material but the agent learns the task from a given reward function. Experiments with reinforcement learning in real world scenarios are hard to accomplish and thus mostly limited to simulated environments [11], [12]. However, a first approach on how to perform reinforcement learning in the real world has been proposed by Wayve using an actor-critic approach to vehicle control on visual input [13].

Furthermore, modelling uncertainty is becoming more and more necessary for a wide range of deep learning applications. Older ideas like Gaussian processes [52] and Bayesian neural networks [61] were mainly focused on uncertainty modelling, much more than a lot of todays deep learning models. But with the application of deep learning algorithms to more and more safety-critical applications, the need for proper self-assessment and modeling of uncertainties modeling is increasing again.

A fairly simple, but mathematically grounded method to obtain uncertainty estimates in neural networks was introduced by Gal et al. with their work on Monte Carlo dropout [6], [7], [14]. They showed that performing dropout during inference can be used to approximate variational inference in Bayesian neural networks. The methods described have been shown to be applicable in camera localization [15] and semantic segmentation [16], even being able to improve model performance on a variety of tasks. With following work they further refined the idea by automatically tuning dropout probabilities with a technique called *Concrete Dropout* [17]. While in the standard setting, the dropout model only allows modelling epistemic uncertainties, further work of Kendall et al. extended the original idea to also model aleatoric uncertainty [5].

A drawback of the dropout-based methods is the requirement for multiple forward passes at inference time which introduces computational complexity (even though this can be reduced by using batches). Osband et al. proposed a method that can be evaluated with a single forward pass and is inspired by the bootstrap method used in statistics [18]. While the original application was demonstrated in reinforcement learning, this method can also be used in an imitation learning setting. Furthermore, Lakshminarayanan et al. offer a non-Bayesian approach to modelling uncertainties with their method of deep ensembles [19]. They compare regular empirical ensembles, predictive models that output parameters of a distribution and ensembles of predictive models - with the best performing models being bootstrapped ensembles. Ilg et al. have reviewed uncertainty generating methods in the context of optical flow [20]. They compare Monte Carlo dropout schemes and ensemble methods similar to [18] and [19] and also introduce a new network architecture that can be seen as combination of the model by [5] used with the bootstrap idea of multiple heads.

In 1994, Christopher Bishop proposed mixture density networks, a combination of neural networks and mixture density models that are able to model any conditional prob-

ability distribution with a weighted mixture of Gaussians [21]. Choi et al. translated this idea and showed how to decompose its uncertainty output to aleatoric and epistemic uncertainties [22]. This mixture density network is evaluated on highway driving data used for steering angle prediction in a bird's-eye view representation of the environment. Part of their evaluation also focuses on the quality of the obtained uncertainties.

In the context of end-to-end driving systems, the necessity to model uncertainties is often posed and used as prime example for safety requirements in deep learning. However, there have not been a lot of publications on the matter so far. Amini et al. were the first to use MC dropout sampling to attain uncertainties in end-to-end learning for self driving cars [23]. They compared a spatial dropout technique to regular, element-wise dropout. In further work, they used a variational autoencoder (VAE) [24] to encode camera input data into a latent space representation, with one latent variable being supervised to predict the steering angle [25]. It is also shown, that the VAE architecture can conveniently be used to perform uncertainty estimation, which appears to be, however, mainly related to the data representation and is used to detect novel input data.

III. Uncertainty Models

Following [2], the baseline model for the steering angle regression consists of a rather simple visual encoder architecture as seen in Tab. I, followed by five fully-connected layers. During training and inference, three weight-shared encoders are used to create a spatial history of input images. This spatial history is created using the vehicle's odometry and the precise image timestamps to get input images approximately $4\,\mathrm{m}$ apart. A flattening and a dropout layer are added after the visual encoder. The following fully-connected block consists of five layers with 1164, 100, 50, 10 and 1 neurons, respectively. A single output $\hat{y}$ is interpreted as the next steering angle passed to the vehicle's controller. Training is performed using a L2 regression loss:

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \frac{1}{2} \left(y_i - \hat{y}_i\right)^2 \tag{1}$$

This baseline model is in turn used as base for the uncertainty model variants as well as reference for the evaluation. In the following subsections we will describe three variations of this baseline model to get uncertainty estimates in addition to the steering angle estimations.

A. Dropout Model

Gal et al. showed that using dropout during inference can be interpreted as approximation of a Gaussian process and Bayesian neural networks [7], [14]. Using dropout multiple times, sampling random subnets of the full network, approximately integrates over the possible models in a Bayesian neural network and thus can be seen as a Monte Carlo approximation to inference in a Bayesian neural network. The idea of dropout acting as a set of random ensembling is visualized in Fig. 1a. Dropout was prominently used as regularization method during training, but has been superseded by other regularization techniques such as batch normalization in recent publications. Fortunately, the current reinterpretation of dropout as approximate variational inference does translate to other regularization methods such as batch-normalization [26].

TABLE I
VISUAL ENCODER OF THE BASELINE ARCHITECTURE

type	patch size / padding / stride	channels
input	52x265	3
conv	3x3 / same / 1	24
conv	3x3 / same / 1	24
maxpool	2x2	24
conv	3x3 / same / 1	36
conv	3x3 / same / 1	36
maxpool	2x2	36
conv	3x3 / same / 1	48
conv	3x3 / same / 1	48
maxpool	2x2	48
conv	3x3 / same / 1	64
conv	3x3 / same / 1	64

To construct the dropout variant based on the baseline model, dropout layers are added after every fully-connected layer but the last. Dropout is implemented by sampling a Bernoulli random variable with probability p. Neuron k in layer l thus gets dropped with probability $z^{l,k} \sim \text{Bernoulli}(p)$. The dropout model is otherwise trained identically to the baseline model with dropout enabled an the L2 loss (c.f. Eq. 1).

Inference and Uncertainty: Inference for an unknown sample x is performed by doing multiple stochastic forward passes and averaging the result. Randomness comes from generating new dropout weight matrices $\hat{W}_t$ from the Bernoulli distribution for every forward pass t. A forward pass with network weights $\hat{W}_t$ is denoted as $f^{\hat{W}_t s}(x)$. The predicted mean steering angle can then be calculated as

$$\mathbb{E}[\hat{y}] = \frac{1}{T} \sum_{t=1}^{T} \hat{y}_t = \frac{1}{T} \sum_{t=1}^{T} f^{\hat{W}_t}(x). \tag{2}$$

We can also get the predictive variance of the stochastic forward passes, seen as our uncertainty estimate, by calculating the variance over all T results plus the inverse model precision τ^{-1} as

$$\mathrm{Var}(\hat{y}) = \underbrace{\tau^{-1}\mathbf{I}}_{\sigma^2} + \frac{1}{T} \sum_{t=1}^{T} \left(f^{\hat{W}_t}(x)\right)^2 - \left(\mathbb{E}[\hat{y}]\right)^2 \tag{3}$$

The model precision τ can be interpreted as label noise and thus representing the aleatoric uncertainty [5], [17]. Being a fixed constant it implies homoscedastic aleatoric uncertainty. Following [16] and [23], we omit the model precision and use the variance of the outputs as predictive uncertainty. An evaluation of different values for the dropout probability p is given in Sec. V.

B. Bootstrap Model

In the original sense, bootstrapping involves sampling K subsets of the training data and training K separate models

on the subsets. Strictly following this idea, we would have to train K models separately and do inference in those networks in parallel, which is time consuming and expensive during inference. Utilizing the idea as proposed by [18] for the reinforcement learning setting, we rely on a simplification instead: multiple models are fused into one model by sharing the visual encoder network, i.e. the convolutional layers. K separate heads are placed on top of the shared encoder network after the dropout and flatten layer. Every head-subnet has the same structure as the fully-connected part in the baseline network.

The process of sampling K subsets of the training data for each subnetwork is done by generating a mask for every sample indicating which head it is passed to during training. This ensures that every head sees only a subset of the whole training data. The data subsampling is realized by generating a binary mask $\{0,1\}^K$ for every training sample indicating whether it is seen by head $k \in [1,K]$. We generate the bitmask by drawing from a Bernoulli distribution (though other distributions like a Poisson distribution are also possible). For sample t and head k, this results in variable $m_t^k \sim \text{Bernoulli}(p)$ with $p = 0.5$ specifying whether a head is trained on a certain sample. A mask where all heads see all samples would equal a regular ensemble method with a shared network.

The sample mask is realized as dictionary mapping from the file name, that identifies a sample (consisting of three input images, the spatial history), to a binary array of size $\{0,1\}^K$ where K is the number of distinct heads. Every time a previously unseen sample is used for training, a new mask is generated and added to the dictionary. After all samples have been seen the mask dictionary is used as a lookup table for the masks. Every head that sees the current sample is included in the loss and therefore gets updated by the gradients. The loss is again an L2 loss. For a given batch size N, m_i^k indicates the binary mask value for sample x_i for head k and $\hat{y}_i^k$ is the steering angle estimate of head k for that sample. The loss thus is defined as

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \sum_{k=1}^{K} m_i^k \frac{1}{2} \left\| y_i - \hat{y}_i^k \right\|^2 . \tag{4}$$

Inference and Uncertainty: Inference is performed by forward-propagation through all heads resulting in K outputs. Note that only a single pass is required to get K outputs, instead of having to do K passes as in the dropout model. The expected steering angle is the mean over all head predictions and the variance is used as proxy for the predictive uncertainty. Thus, the inference can be written as

$$\mathbb{E}[\hat{y}] = \frac{1}{K} \sum_{k=1}^{K} f^{(k)}(x) \tag{5}$$

$$\text{Var}(\hat{y}) = \frac{1}{K} \sum_{k=1}^{K} \left(f^{(k)}(x) \right)^2 - (\mathbb{E}[\hat{y}])^2, \tag{6}$$

with $f^{(k)}(x)$ being the prediction given by head k for sample x. With the Bootstrap model there is no way to get aleatoric and epistemic uncertainty separated. Instead, Eq. 6 only yields the predictive variance which is used as predictive uncertainty. The inference step is computationally more expensive than inference in the baseline model as there are K more computations in the fully-connected layer.

C. Gaussian Mixture Model

The main idea is to learn parameters of a Gaussian mixture model (GMM) as neural network output. Choi et al. used this technique recently in the context of deeper neural networks for more modern tasks [22] and we follow their implementation in this work.

A Gaussian Mixture Model is composed of multiple Gaussians combined in a weighted sum. The parameters of a GMM are $\theta = \pi_i, \mu_i, \sigma_i^2$, $i \in [1,K]$ for K mixtures.

$$p(y|\theta) = \sum_{i=1}^{K} \pi_i \mathcal{N}\left(y|\mu_i, \sigma_i^2\right) \tag{7}$$

A neural network is used to learn to predict the weights π_i, means μ_i and variances σ_i^2 as output.

To improve numerical stability, following [22], the raw network outputs $\hat{\pi}_i, \hat{\mu}_i, \hat{\sigma}_i^2$ are post-processed to obtain the actual GMM parameters:

$$\pi_i = \frac{\exp\left(\hat{\pi}_i - \max(\pi)\right)}{\sum_{k=1}^{K} \exp\left(\hat{\pi}_k - \max(\pi)\right)} \tag{8}$$

$$\mu_i = \hat{\mu}_i \tag{9}$$

$$\sigma_i^2 = \sigma_{\max} \rho\left(\hat{\sigma}_i^2\right) \tag{10}$$

The maximum mixture weight among all mixtures in subtracted in Eq. 8 to prevent exploding values through the exponential functions. The individual mixture weights for mixture i are then calculated via softmax to ensure a sum of 1. $\rho(.)$ in Eq. 10 denotes the element-wise sigmoid function, which is applied and scaled via σ_{max}. The sigmoid is used instead of an exponential function because of the same reason of exploding values. As the sigmoid function maps to range $[0,1]$, σ_{max} then determines the maximum aleatoric uncertainty. For our experiments we used a value of $\sigma_{\text{max}} = 5$. In this work, we assume one-dimensional steering angle output and thus use the variance σ^2 in Eq. 10 instead of the covariance matrix Σ used by [22].

The mixture model is trained using the negative log-likelihood as loss. A small constant $\epsilon = 1e-6$ is added to the argument of the logarithm for numerical stability.

$$\begin{aligned} \mathcal{L}(\theta) &= -\frac{1}{N} \sum_{i=1}^{N} \log\left(p\left(y_i|x_i\right)\right) \\ &= -\frac{1}{N} \sum_{i=1}^{N} \log\left(\sum_{j=1}^{K} \pi_j^{(i)} \mathcal{N}\left(y_i \mid \mu_j^{(i)}, \sigma_j^{2(i)}\right) + \epsilon \right) \end{aligned} \tag{11}$$

with the superscript $^{(i)}$ noting the predictions of π, μ and σ for input x_i. N is the number of samples in a batch, K the number of mixtures used and $\mathcal{N}$ denotes the standard probability density function of the normal distribution. x_i

denotes the the current sample, in our case the three concatenated images with the true label y_i, i.e. the steering angle. For every sample, the normal probability density function of every mixture $\mathcal{N}\left(\mu_j\left(\mathbf{x_i}\right), \sigma_j^2\left(\mathbf{x_i}\right)\right)$ is evaluated at y_i and weighted by $\pi_j(x_i)$

Inference and Uncertainty: The Mixture model allows to do inference with a single forward pass while having less computational complexity than the bootstrap model as the split into multiple mixtures is at the last layer. It also provides an elegant way to estimate uncertainties directly from the estimated GMM. Furthermore, the uncertainty output can be separated into aleatoric and epistemic uncertainty. The mean μ_j and variance σ_j^2 of a single Gaussian can be interpreted as steering angle and uncertainty. All mixture components are combined in a weighted sum and we get the following results for the expected steering angle:

$$\begin{aligned}\mathbb{E}[\hat{y}] &= \sum_{j=1}^{K} \pi_j(x) \int y \mathcal{N}\left(y|\mu_j(x), \sigma_j^2(x)\right) dy \\ &= \sum_{j=1}^{K} \pi_j(x)\mu_j(x) \end{aligned} \tag{12}$$

The total variance, in our context called the predicted variance which is interpreted as predictive uncertainty, decomposes into the weighted sum of the variances and the weighted variances of the means, i.e.

$$\begin{aligned}\mathrm{Var}(\hat{y}) &= \int (y - \mathbb{E}[\hat{y}])^2 p(y) dy \\ &= \underbrace{\sum_{j=1}^{K} \pi_j(x)\sigma_j^2(x)}_{\text{aleatoric}} \\ &\quad + \underbrace{\sum_{j=1}^{K} \pi_j(x) \left\| \mu_j(x) - \sum_{k=1}^{K} \pi_k(x)\mu_k(x) \right\|^2}_{\text{epistemic}}\end{aligned} \tag{13}$$

The second term is also referred to as explained variance or epistemic uncertainty as it vanishes with more data. A decreasing aleatoric uncertainty can be interpreted as decreasing variance of the mixture components. In turn, intuitively, a decreasing epistemic uncertainty equals to the means of the mixture components getting closer. However, this is only the case as long as there are no ambiguous situations as would be the case at intersection with multiple possibles routes. The models in this work had the turn indicators as additional input to handle such ambiguities. Some ambiguities, however, might still remain and this straightforward interpretation might not hold in all cases.

IV. Comparison Metrics

This section introduces metrics for assessing the quality of steering angle predictions, as well as the quality of the uncertainty estimates. The underlying model is treated as black box with output $\{\hat{y}, \sigma^2\}$. $\hat{y}$ is the steering angle estimate and σ^2 the predictive variance that is interpreted as predictive uncertainty.

Measuring the quality of uncertainty is difficult as compared to the steering angle, there are no training labels for uncertainty. We can think, however, about qualitative properties that reasonable uncertainty estimates should have. The goal of uncertainties in autonomous vehicles is that they can explain the models errors. Thus, we would expect high uncertainty in unknown (epistemic uncertainty) or multimodal scenarios (aleatoric uncertainty) where the model makes a mistake. We also assume high epistemic uncertainty in scenarios beyond the generalization scope of the model, that is if the test data is sufficiently different from the training data (e.g. driving in a completely different environment like snow, if the model was trained on summer roads).

A. Driving Performance

Driving performance is assessed with the mean absolute error (MAE) and the root mean squared error (RMSE) over a test set of size N with images x_i, true steering angle y_i and the predicted steering angle for sample i, $\hat{y}_i$:

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{N} \left(y_i - \hat{y}_i\right)^2}{N}} \tag{14}$$

$$\mathrm{MAE} = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i| \tag{15}$$

B. Calibration

Calibration, also called confidence calibration, tries to validate the uncertainty estimate from a frequentist point of view by comparing local uncertainty predictions with long-run frequencies [19], [27]. In the classification setting, this can be interpreted as, e.g. checking if 80% out of all samples with predicted confidence of 80% are correct. Intuitively, this means that the uncertainty estimations have to be a true probability and should reflect the true likelihood [8]. Well calibrated uncertainty estimates play an decisive role when neural network models are to be embedded in probabilistic models, in order to be able to correctly combine estimates from different sources without overrating some of them. Calibration is orthogonal to a models driving performance, i.e. it can yield good steering angle estimates but the accompanying uncertainty outputs do not represent the true probability.

Calibration can be visualized using calibration plots, also called reliability diagrams. In a calibration plot, the observed sample accuracy is plotted as function of the confidence or uncertainty, i.e. the expected accuracy. Calculating the calibration in our regression setting is performed as follows: First, a z% confidence interval is computed for each sample in the data set based on the predicted mean μ and variance σ^2. The confidence interval (CI) specifies the lower and upper bound, arranged symmetrically around the mean, containing z% of the given Gaussian distributions mass:

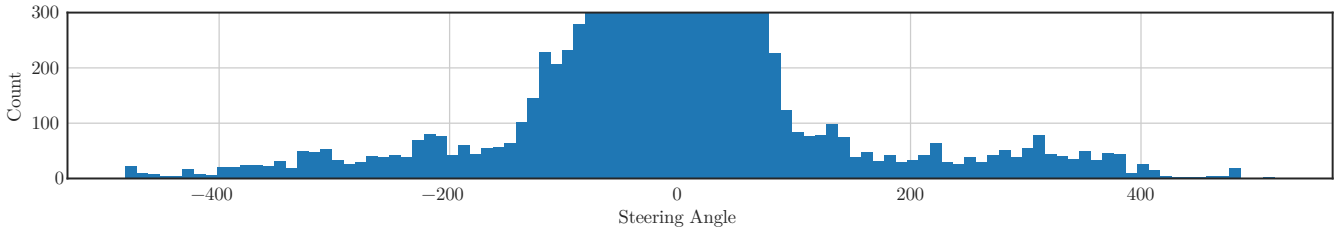

Fig. 2. Histogram of steering angle truths in training set. Samples with small steering angles predominate, as there are more than 40'000 samples for the steering angle range from $-5°$ to $5°$. The y-axis is limited to [0,300] in order to show the distribution of larger steering angles.

$$\mathrm{CI}(z) = \left[\mathrm{ppf}\left(\frac{1-z}{2}\right), \mathrm{ppf}\left(\frac{1+z}{2}\right)\right], \tag{16}$$

$$\mathrm{ppf}(x) = \mathbb{P}\left[\mathcal{N}\left(\mu, \sigma^2\right) \leq x\right]. \tag{17}$$

We calculate the confidence intervals for 11 values from 0 to 1, i.e. $0, 0.1, \ldots, 1$ For every sample, we can check if the samples true value lies within the calculated z% confidence interval. For every prediction interval, we calculate the fraction of the samples fulfilling that condition and plot this over the value z%. A diagonal line indicates perfect calibration, because a point on the diagonal equals to the situation that the expected fraction of samples lying inside the confidence interval equals the observed fraction. Note that the calibration plot does not contain information on how many or which samples the model is calibrated on. To compensate, we can computed the calibration plot for different subsets of the data to be able to detect if a model is miss-calibrated for a specific subset of the data.

V. Evaluation Results

All models were trained to 350,000 iterations with a batch size of 64 on a single NVIDIA 1080 Ti GPU. This resulted in average training times of one to two days. Other hyper-parameters like learning rate, dropout values for bootstrap and the mixture density model as well as the network architecture were loosely tuned, but room for further improvements might remain. The focus, however, is on comparing the different uncertainty models rather than perfecting the driving performance, as real world driving performance for all models was at least as good as what was possible in previous work (c.f. [2]) and the models were able to drive in various scenarios and varying weather conditions.

The data, about 5 hours of driving in multiple weather conditions as well as street types, is split into a training, validation and test sequences of relative sizes 60-20-20, to prevent dependency between samples. The training corpus thus consists of about 200,000 samples (tuples of 3 images forming the spatial history). Training samples with zero speed are not included. Fig. 2 shows the distribution of the steering angles contained in the training data and highlights a major problem that is hard to avoid with regular driving recordings: The steering angle distribution is highly imbalanced, with a strong focus on small and zero steering angles and a strongly decreasing amount of samples for larger steering angles. This is a natural result from street layouts containing mostly straight segments or large radii. The fact of label imbalance complicates the task, but also raises the opportunity to test our uncertainty predictions. This imbalance is smoothed during training by adapting the sampling probability to the steering angle frequency.

A. Driving Performance

Tab. II summarizes the driving performance for the uncertainty models in comparison to the baseline model, the dashed line marking the performance of the best-performing baseline model. Except for the different Monte Carlo dropout models, which reach similar performance to the baseline, the proposed uncertainty models all perform better in terms of RMSE and MAE.

For the dropout model, dropout probabilities $p \in \{0.5, 0.4, 0.3\}$ are evaluated, but the dropout probability has not significant or systematic influence on model performance. 10 forward passes are calculated for every sample to obtain uncertainty estimates.

According to Tab. II, the number of heads in the bootstrap model has only a small influence on the driving performance. The spread of RMSE and MAE values across models are low and there is no clear correlation between number of heads and driving performance detectable.

Compared to the other models, the mixture models perform better with almost 10% relative improvement over the best baseline models. Experimentally, the optimal number of mixtures for our training data was found to be 10, but this number vary with other data sets. The reason is that different mixtures will learn to focus on different scenarios. With data sets containing a greater variety of scenarios, e.g. more weather conditions or road types, a greater number of mixtures might be required.

B. Inference Time

Inference time is measured by averaging the time required for an inference step over 8000 samples. For models with Monte Carlo dropout a single inference step includes multiple stochastic forward passes. The results can also be found in Tab. II. As expected, the multiple forward passes required to obtain uncertainty estimates for the dropout models - even in batch inference implementation - is considerably higher than for the other models, especially when compared to the mixture models. The mixture models on the other hand have the least amount of parameters of the evaluated model architectures. Contrary to the assumption, inference

TABLE II
EVALUATION RESULTS

Model	RMSE	MAE	Time [ms]
Mixture 10	**30.94**	**10.44**	3.57
Mixture 8	33.16	10.56	3.58
Mixture 15	33.34	10.99	3.57
Bootstrap 13	33.78	11.96	6.49
Bootstrap 8	33.79	12.08	5.19
Mixture 5	33.95	11.42	3.58
Bootstrap 4	34.01	11.98	4.18
Bootstrap 6	34.17	12.11	4.70
Bootstrap 7	34.32	12.36	4.94
Bootstrap 10	34.84	12.12	5.71
Dropout 0.5	35.00	12.37	11.15
Dropout 0.3	35.12	12.81	11.16
Dropout 0.4	37.56	13.44	11.17
Base 0.4	34.84	13.00	**3.30**
Base 0.5	35.71	12.24	3.30

time does not scale linearly with the amount of heads in the bootstrap model or mixtures for the mixture model.

Fig. 3. Example scenarios with the lowest predictive uncertainties, evaluated with the mixture density network.

Fig. 4. Examples with the highest predictive uncertainties and with an absolute steering angle smaller than 10, evaluated with the mixture network.

C. Uncertainty Calibration

Fig. 5 shows the calibration calculated as described in Sec. IV-B for the best performing models of each category (also c.f. Tab. II). The x-axis denotes the expected fraction of samples with the respective true steering angle lying inside the $z\%$ confidence interval, that is, the interval containing $z\%$ of the prediction. The y-axis denotes the observed fraction of samples with the true steering angle inside the prediction interval. Better calibrated models result in curves closer to the diagonal.

For our experiments, the dropout model exhibits the best calibration of the compared models, while the bootstrap and mixture models both seem undercalibrated. Undercalibration implies the uncertainty estimate being too small and is expressed as a curve below the diagonal. This means that the confidence interval, which has its width determined by the predicted uncertainty, is too narrow to contain the expected fraction of samples.

D. Predictive Uncertainties

Fig. 6 depicts the predictive uncertainty measurement over the steering angle. For all models, the predictive uncertainty is increasing with the steering angle. Since the uncertainty is expected to increase with fewer data and our training data is imbalanced, this observation affirms the uncertainty estimates, but also indicates that the models learn a connection between high steering angles and high uncertainties. However, Fig. 4 indicates that the models are still able to detect high uncertainties for small steering angles. This confirms that the models do not learn a simple mapping from steering angle to uncertainty estimate, but are nevertheless able to identify uncertain scenarios for all steering angles. Furthermore, since the relation between steering angles and the driven radius is non-linear, higher uncertainty estimates are also plausible.

VI. CONCLUSION

Our evaluation has shown that it is possible to construct end-to-end control models for steering angle prediction that are not only performing well at the driving task, b0tut are also capable of estimating their confidence with an uncertainty measure as would be required when integrating

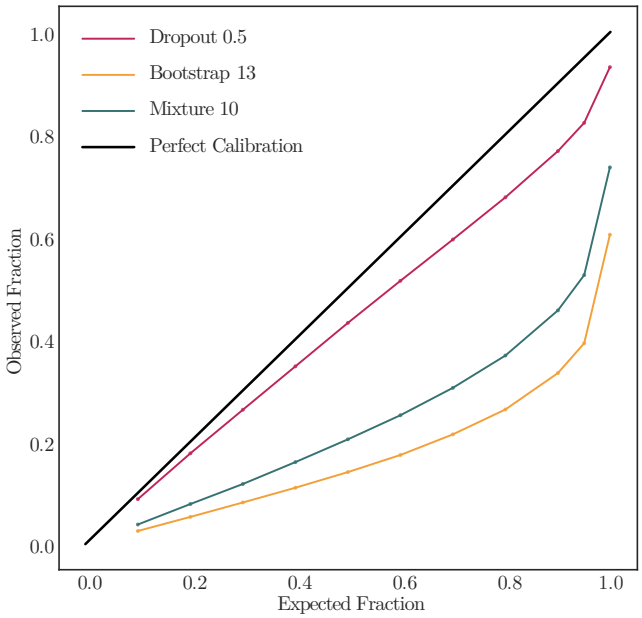

Fig. 5. Calibration for the best performing models of each type. The x-axis shows the predicted fraction of samples with the respective true steering angle lying inside the $z\%$ confidence interval, the y-axis the observed fraction. The black diagonal represents perfect calibration. Values below the diagonal are undercalibrated, i.e. estimate uncertainty too low and values above the diagonal are overcalibrated.

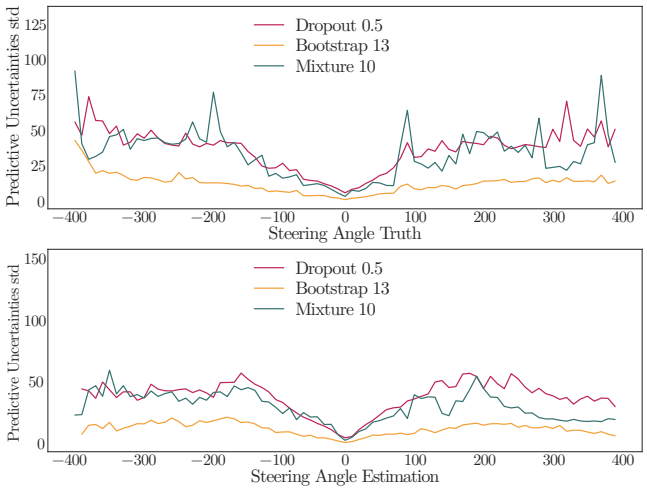

Fig. 6. Predictive uncertainties std over true steering angles and steering angle estimation. With absolutely increasing steering angle, the uncertainty estimates also rise, which is affected by less available training data for higher steering angles. Uncertainties are calculated as the mean of bins over the steering angle with bin size of 10°.

learned models into probabilistic frameworks. The driving performance measured by RMSE and MAE is comparable or better than the performance of the baseline without uncertainty estimates, while the uncertainty extensions only require minimal architectural changes. The computational overhead is model-specific, but is still well below critical.

Mixture models, even though they seem to be undercalibrated, offer a very promising perspective to scale well into larger data sets and might also be able to capture multimodalities in forking situations or in intersections. Furthermore, they can easily be created to model both epistemic and aleatoric uncertainty. Some more work has to be done on obtaining proper calibration for mixture density networks to incorporate them into probabilistic architectures.

As measure for the quality of uncertainty estimates, we proposed evaluating a models uncertainty calibration. Among the uncertainty models, the dropout model exhibit the best calibrated estimates, while the bootstrap and mixture models seem to overestimate their confidence.

Apart from enhancing neural network models with additional confidence estimates, uncertainties can also be used in an active learning setting. This is an interesting topic for future research.

ACKNOWLEDGMENTS

This research was supported by the Ministry of Economic Affairs, Labour and Housing Baden-Württemberg and performed within the project ”Leistungszentrum Profilregion Mobilitätssysteme Karlsruhe - Pilotphase II”.

REFERENCES

[1] 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,” *arXiv:1604.07316*, 2016.

[2] C. Hubschneider, A. Bauer, M. Weber, and J. M. Zöllner, “Adding navigation to the equation: Turning decisions for end-to-end vehicle control,” *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017.

[3] A. Amini, G. Rosman, S. Karaman, and D. Rus, “Variational End-to-End Navigation and Localization,” *arXiv:1811.10119*, 2018.

[4] R. McAllister, Y. Gal, A. Kendall, M. Van Der Wilk, A. Shah, R. Cipolla, and A. V. Weller, “Concrete problems for Autonomous Vehicle Safety: Advantages of Bayesian Deep Learning.” International Joint Conferences on Artificial Intelligence, Inc., 2017.

[5] 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*, 2017.

[6] Y. Gal, “Uncertainty in Deep Learning,” 2016.

[7] Y. Gal and Z. Ghahramani, “Dropout as a Bayesian Approximation: Appendix,” *arXiv:1506.02157*, 2015.

[8] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, “On Calibration of Modern Neural Networks,” *arXiv:1706.04599*, 2017.

[9] D. A. Pomerleau, “ALVINN: An Autonomous Land Vehicle in a Neural Network,” in *Advances in Neural Information Processing Systems 1*, D. S. Touretzky, Ed. Morgan-Kaufmann, 1989.

[10] C. Hubschneider, A. Bauer, J. Doll, M. Weber, S. Klemm, F. Kuhnt, and J. M. Zöllner, “Integrating end-to-end learned steering into probabilistic autonomous driving,” in *International Conference on Intelligent Transportation Systems (ITSC)*, 2017.

[11] S. Sharifzadeh, I. Chiotellis, R. Triebel, and D. Cremers, “Learning to Drive using Inverse Reinforcement Learning and Deep Q-Networks,” *arXiv:1612.03653*, 2016.

[12] P. Wolf, C. Hubschneider, M. Weber, A. Bauer, J. Hartl, F. Dürr, and J. M. Zöllner, “Learning how to drive in a real world simulation with deep Q-Networks,” in *2017 IEEE Intelligent Vehicles Symposium (IV)*. Los Angeles, CA, USA: IEEE, 2017.

[13] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah, “Learning to Drive in a Day,” *arXiv:1807.00412*, 2018.

[14] Y. Gal and Z. Ghahramani, “Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference,” *arXiv:1506.02158*, 2015.

[15] A. Kendall and R. Cipolla, “Modelling Uncertainty in Deep Learning for Camera Relocalization,” *arXiv:1509.05909*, 2015.

[16] A. Kendall, V. Badrinarayanan, and R. Cipolla, “Bayesian SegNet: Model Uncertainty in Deep Convolutional Encoder-Decoder Architectures for Scene Understanding,” *arXiv:1511.02680*, 2015.

[17] Y. Gal, J. Hron, and A. Kendall, “Concrete Dropout,” *arXiv:1705.07832*, 2017.

[18] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, “Deep Exploration via Bootstrapped DQN,” in *Advances in Neural Information Processing Systems 29*, 2016.

[19] B. Lakshminarayanan, A. Pritzel, and C. Blundell, “Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles,” 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.

[20] E. Ilg, O. Çiçek, S. Galesso, A. Klein, O. Makansi, F. Hutter, and T. Brox, “Uncertainty Estimates and Multi-Hypotheses Networks for Optical Flow,” *arXiv:1802.07095*, 2018.

[21] C. M. Bishop, “Mixture Density Networks,” Tech. Rep., 1994.

[22] S. Choi, K. Lee, S. Lim, and S. Oh, “Uncertainty-Aware Learning from Demonstration using Mixture Density Networks with Sampling-Free Variance Modeling,” *arXiv:1709.02249*, 2017.

[23] A. Amini, A. Soleimany, S. Karaman, and D. Rus, “Spatial Uncertainty Sampling for End-to-End Control,” *arXiv:1805.04829*, 2018.

[24] D. P. Kingma and M. Welling, “Auto-Encoding Variational Bayes,” *arXiv:1312.6114*, 2013.

[25] A. Amini, W. Schwarting, G. Rosman, B. Araki, S. Karaman, and D. Rus, “Variational autoencoder for end-to-end control of autonomous driving with novelty detection and training de-biasing,” *International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[26] M. Teye, H. Azizpour, and K. Smith, “Bayesian Uncertainty Estimation for Batch Normalized Deep Networks,” *arXiv:1802.06455*, 2018.

[27] A. Niculescu-Mizil and R. Caruana, “Predicting Good Probabilities with Supervised Learning,” in *Proceedings of the 22Nd International Conference on Machine Learning (ICML)*, 2005.