This paper has been accepted for publication at the
RSS Workshop on Scene and Situation Understanding for Autonomous Driving (UAD), Freiburg, 2019

A General Framework for Uncertainty Estimation in Deep Learning

Mattia Segú, Antonio Loquercio, and Davide Scaramuzza
Dep. of Informatics University of Zurich and
Dep. of Neuroinformatics of the University of Zurich and ETH Zurich, Switzerland

***Abstract*—End-to-end learning has recently emerged as a promising technique to tackle the problem of autonomous driving. Existing works show that learning a navigation policy from raw sensor data may reduce the system's reliance on external sensing systems, (e.g. GPS), and/or outperform traditional methods based on state estimation and planning. However, existing end-to-end methods generally trade off performance for safety, hindering their diffusion to real-life applications. For example, when confronted with an input which is radically different from the training data, end-to-end autonomous driving systems are likely to fail, compromising the safety of the vehicle. To detect such failure cases, this work proposes a general framework for uncertainty estimation which enables a policy trained end-to-end to predict not only action commands, but also a confidence about its own predictions. In contrast to previous works, our framework can be applied to any existing neural network and task, without the need to change the network's architecture or loss, or to train the network. In order to do so, we generate confidence levels by forward propagation of input and model uncertainties using Bayesian inference. We test our framework on the task of steering angle regression for an autonomous car, and compare our approach to existing methods with both qualitative and quantitative results on a real dataset. Finally, we show an interesting by-product of our framework: robustness against adversarial attacks.**

Supplementary Material

For supplementary video see: https://youtu.be/1JtU78Heceg. The project's code is available at: https://github.com/mattiasegu/A_General_Framework_for_Uncertainty_Estimation_in_Deep_Learning.

I. Introduction

A current trend in autonomous driving consists of learning navigation policies end-to-end from raw sensor data. Such approaches are typically categorized by the way they are trained: (i) supervised learning [1]–[3] or (ii) reinforcement learning [4]. Despite their differences, both types of approaches rely on the predictions of one or more neural networks which, after training, are assumed to be accurate. Blindly trusting the prediction of a neural network could have severe implications: What if the current observation is very different from the training ones? What if the networks' inputs are corrupted by noise? In safety-critical applications, e.g., autonomous driving, such questions cannot be disregarded. To tackle these problems, we propose to accompany every prediction of a neural network with an uncertainty measuring the network's confidence to its prediction.

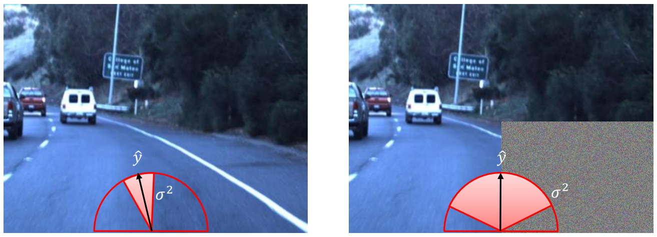


Fig. 1. Uncertainty estimation is necessary to build a safe autonomous driving system. Indeed, a system can be fully functional on a clean image (left), but can return erroneous prediction when processing a corrupted input (right). Uncertainty estimation provides an automatic detection mechanism of such failure cases. The red slices represent one standard deviation from the mean prediction.

Recent works have leveraged the idea of measuring uncertainty to increase the safety of autonomous driving systems. For example, Kahn et al. [4] propose a collision avoidance system trained with reinforcement learning which uses uncertainty predictions to minimize the risk of collision. More recently, Lee et al. [2] propose to exploit an ensemble of redundant neural networks trained on different sensors modalities, trusting at test time the prediction with lowest uncertainty. In addition, Feng et al. [3] show how object detection pipelines may benefit from uncertainty estimates in a driving scenario.

All previous works deployed heuristics to generate uncertainty estimates, which are generally problem specific and do not necessarily generalize across tasks or datasets. To improve the quality of uncertainty estimates, this work proposes to apply ideas borrowed by Bayesian inference [5] to deep learning models. In particular, we propose to divide the uncertainty into two distinct components: (i) *Model uncertainty*, which derives from the uncertainty on network weights [6] and represents a measure of the model's confidence about a specific sample, and (ii) *Data uncertainty*, which measures the impact of input sensor's noise on the output prediction. Generally, model uncertainty increases for samples which are not well represented in the training dataset, while data uncertainty is independent of the training dataset and is generated by sensor noise or adversarial attacks.

In our framework, we compute both types of uncertainty by forward propagating both probability distribution over weights and input noise through a (possibly pre-trained) neural network. This allows us to compute uncertainties without changing the network's architecture or loss, which is a feature that classic approaches for uncertainty estimation lack of. We test our framework on the task of end-to-end steering angle estimation, and compare it on a quantitative and qualitative basis with existing approaches for uncertainty estimation. In doing so, we show that our approach can accurately estimate uncertainties at no cost in term of performance. In addition, we qualitatively demonstrate the robustness of our method against adversarial attacks.

II. Related Work

Due to the large number of parameters and the non-linear activation functions, the true posterior distribution of a neural network is intractable to compute. To approximate the true posterior, existing methods deploy different techniques, mainly based on variational inference, which we present in the following sections.

A. Dropout to Estimate Model Uncertainty

To account for model uncertainty in deep learning, a distribution is placed over neural network (NN) weights $\boldsymbol{\omega}$, defining a Bayesian neural network [7]–[9]. The work of Gal et al. [10] provides a mathematically grounded framework to capture model uncertainty leveraging dropout at test-time [11]. Specifically, they propose to approximate the intractable posterior distribution over network weights $p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})$ given a specific training set $\mathrm{D} = \{\mathbf{X}, \mathbf{Y}\}$ by collecting multiple predictions for a single input, each with a different realization of weights due to dropout. This method is often referred to as *Monte Carlo* (MC) *dropout*.

Gal et al. show how the mean prediction over multiple MC samples improves predictive precision and proposes to recover the model uncertainty for prediction $\mathbf{y}$ given input $\mathbf{x}$ as:

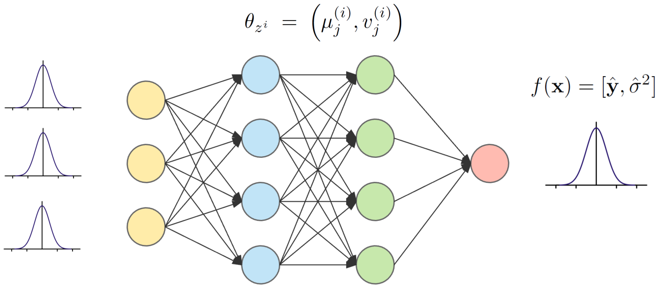
$$
\begin{aligned}
\mathrm{Var}_{q(\mathbf{y}|\mathbf{x})}(\mathbf{y}) &\approx \sigma^2 \mathbf{I}_D \\
&\quad + \frac{1}{T} \sum_{t=1}^{T} \widehat{\mathbf{y}}_t^T \widehat{\mathbf{y}}_t - \mathbb{E}_{q(\mathbf{y}|\mathbf{x})}(\mathbf{y})^T \mathbb{E}_{q(\mathbf{y}|\mathbf{x})}(\mathbf{y})
\end{aligned}
$$

with $\{\widehat{\mathbf{y}} \equiv \widehat{\mathbf{y}}(\mathbf{x}, \mathbf{W}_1^t, ..., \mathbf{W}_L^t)\}_{t=1}^T$ a set of T sampled outputs for randomly masked weights $\widehat{\mathbf{W}}_i^t \sim q(\mathbf{W}_i)$ extracted from the distribution of the i-th layer.

One of the main limitations of this work is that σ^2, a measure of the amount of noise in the data, is assumed to be constant for any input. However, self-driving cars operate in a variety of environmental conditions and are equipped with sensors that are often sensitive to temperature and illumination conditions. Consequently, different input samples might present different noise levels.

B. Learning Model and Data Uncertainty Together

A further step towards total uncertainty estimation was made by Kendall et al. [12], that proposed a framework to jointly estimate both data and model uncertainty under the assumption of having input points with different noise levels than others. The data uncertainty is learned by training the NN under the *heteroscedastic loss*:



$\theta_{z^i} = \left(\mu_j^{(i)}, v_j^{(i)}\right)$
$f(\mathbf{x}) = [\hat{\mathbf{y}}, \hat{\sigma}^2]$

Fig. 2. Assumed Density Filtering [13], [14] returns an estimate of both the predictive mean and the data uncertainty by propagating input uncertainty through the neural network.

$$
\mathcal{L}_{\mathrm{NN}}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2\sigma^2(\mathbf{x}_i)} ||\mathbf{y}_i - f(\mathbf{x}_i)||^2 + \frac{1}{2} \log \sigma^2(\mathbf{x}_i) \quad (1)
$$

where the input noise $\sigma^2 = \sigma^2(\mathbf{x}_i)$ has been made input-dependent, and $\mathbf{f}(\mathbf{x}_i)$ is the output of the NN with parameters θ for input $\mathbf{x}_i$. By training a neural network with heteroscedastic loss (Eq. 1) and by taking multiple forward samples applying dropout at test-time as in Sec. II-A, the total variance is recovered as:

$$
\begin{aligned}
\mathrm{Var}(\mathbf{y}) &\approx \frac{1}{T} \sum_{t=1}^{T} \hat{\sigma}_t^2 \\
&\quad + \frac{1}{T} \sum_{t=1}^{T} \widehat{\mathbf{y}}_t^T \widehat{\mathbf{y}}_t - \mathbb{E}_{q(\mathbf{y}|\mathbf{x})}(\mathbf{y})^T \, \mathbb{E}_{q(\mathbf{y}|\mathbf{x})}(\mathbf{y})
\end{aligned}
$$

with $\{\hat{\mathbf{y}}_t, \hat{\sigma}_t^2\}_{t=1}^T$ a set of T sampled outputs for randomly masked weights $\widehat{\mathbf{W}}_t \sim q(\mathbf{W})$. However, this approach needs to modify the network structure splitting its head into two parts to learn data uncertainty. Also, this approach forces to re-train under the heteroscedastic loss (Eq. 1) to retrieve an uncertainty estimate, which often results in a performance drop.

C. Data Uncertainty Propagation with Assumed Density Filtering

Sampling approaches are often too slow for practical scenarios. Gast et al. [15] introduced a lightweight approach to recover uncertainty while maintaining the same network architecture, with minor changes to propagate both mean and variance of the input distribution. They propose to replace every intermediate network activation by distributions, following the work in [16] for non-linear Gaussian belief networks. Moreover, the distribution is propagated through the network in a single pass using Assumed Density Filtering (ADF) [13], [14] (Fig. 2).

The main advantage of this approach is that it is possible to assume a distribution over input data and propagate it through the network to obtain an estimate of *data uncertainty*. On the other hand, Gast et al. disregards the model uncertainty contribution, under the assumption that it can be explained away if large amount of data are available. However, in autonomous driving, where out-of-distributions examples may cause the model to return wrong predictions, disregarding model uncertainty might have severe implications.

III. Recovering Total Uncertainty

A. Fusing MC Dropout and Assumed Density Filtering

Autonomous cars are equipped with an increasing number of sensors, for which sensor variance is often known from the data sheet. Having detailed knowledge of the input variance, we suggest to recover data uncertainty by propagating uncertainty on input points through the network, as in Section II-C. We will now deploy a framework that can account for both model and data uncertainty by propagating input mean and variance with ADF network [15] and by taking MC samples of the resulting output mean and variance.

The ADF network from Section II-C can be seen as a probabilistic model $p(\mathbf{y}|\mathbf{z},\boldsymbol{\omega})p(\mathbf{z}|\mathbf{x})$, where $\mathbf{x}$ is the input sample and $\mathbf{z}$ is the input perturbed by white gaussian noise:

$$p(\mathbf{z}|\mathbf{x}) \sim \mathcal{N}\left(\mathbf{z};\mathbf{x},\sigma_n^2\right) \tag{2}$$

and σ_n^2 is the variance of the *n-th* input pixel.

To take into account also weight uncertainty, the ADF network is turned into a Bayesian Neural Network (BNN) by placing a distribution over its weights as in II-A. After training the standard NN with any desired loss, we convert it to its ADF version. Model uncertainty is recovered by collecting multiple stochastic forward samples with MC dropout applied on the ADF network itself. The network will output both predictive mean $\hat{\mathbf{y}}$ and data variance $\hat{\sigma}_{data}^2$, thus the total predictive uncertainty for prediction $\mathbf{y}$ given input $\mathbf{x}$ in this model can be approximated as:

$$\begin{aligned}\text{Var}(\mathbf{y}) \approx &\frac{1}{T}\sum_{t=1}^{T}\hat{\sigma}_{data,t}^2 \\ &+ \frac{1}{T}\sum_{t=1}^{T}\widehat{\mathbf{y}}_t^T\widehat{\mathbf{y}}_t - \mathbb{E}_{q(\mathbf{y},\mathbf{z}|\mathbf{x})}(\mathbf{y})^T\,\mathbb{E}_{q(\mathbf{y},\mathbf{z}|\mathbf{x})}(\mathbf{y})\end{aligned} \tag{3}$$

with $\widehat{\mathbf{y}}_t \equiv \widehat{\mathbf{y}}_t(\mathbf{x},\widehat{\mathbf{z}}_{1,t},...,\widehat{\mathbf{z}}_{L,t})$ and $\{\widehat{\mathbf{y}}_t,\hat{\sigma}_{data,t}^2\}_{t=1}^T$ a set of T sampled outputs for randomly masked weights $\widehat{\mathbf{W}}_t \sim q(\mathbf{W})$.

This result means that the data uncertainty retrieved with uncertainty propagation and the model uncertainty obtained by MC sampling can be directly *summed* to obtain the total uncertainty. We will now show how Equation 3 can be derived.

B. Predictive Variance

Consider the probabilistic model of the ADF network and the probabilistic distribution over the input:

$$\begin{aligned}p(\mathbf{y},\mathbf{z}|\mathbf{x},\boldsymbol{\omega}) &= p(\mathbf{y}|\mathbf{z},\boldsymbol{\omega})p(\mathbf{z}|\mathbf{x}) \\ p(\mathbf{z}|\mathbf{x}) &\sim \mathcal{N}\left(\mathbf{z};\mathbf{x},\sigma_n^2\right)\end{aligned} \tag{4}$$

Let's now place a posterior distribution $p\left(\boldsymbol{\omega}|\mathbf{X},\mathbf{Y}\right)$ over network weights given the training data $\mathbf{D}=\{\mathbf{X},\mathbf{Y}\}$. Consequently, the full posterior distribution of the Bayesian ADF network can be parametrized as

$$\begin{aligned}p\left(\mathbf{y},\mathbf{z}|\mathbf{x},\mathbf{X},\mathbf{Y}\right) &= \left(\int p(\mathbf{y}|\mathbf{z},\boldsymbol{\omega})\cdot p(\boldsymbol{\omega}|\mathbf{X},\mathbf{Y})d\boldsymbol{\omega}\right)\cdot p(\mathbf{z}|\mathbf{x}) \\ &= \int p(\mathbf{y},\mathbf{z}|\mathbf{x},\boldsymbol{\omega})\cdot p(\boldsymbol{\omega}|\mathbf{X},\mathbf{Y})d\boldsymbol{\omega}\end{aligned} \tag{5}$$

where $p(\mathbf{y},\mathbf{z}|\mathbf{x},\boldsymbol{\omega}) = p(\mathbf{y}|\mathbf{z},\boldsymbol{\omega})\cdot p(\mathbf{z}|\mathbf{x}) \sim \mathcal{N}(\widehat{\mathbf{y}}_{\boldsymbol{\omega}},\sigma_{data,\boldsymbol{\omega}}^2\mathbf{I}_D)$ for each model weights realization $\boldsymbol{\omega}$. Also, we approximate the intractable posterior over network weights as

$$p(\boldsymbol{\omega}|\mathbf{X},\mathbf{Y}) \approx q(\boldsymbol{\omega}) = \text{Bern}(\mathbf{z}_1)\cdots\text{Bern}(\mathbf{z}_L) \tag{6}$$

where Bern$(\mathbf{z}_i)$ is a Bernoullian distribution over the activation of the i-th layer. Thus,

$$p\left(\mathbf{y},\mathbf{z}|\mathbf{x},\mathbf{X},\mathbf{Y}\right) \approx \int p(\mathbf{y},\mathbf{z}|\mathbf{x},\boldsymbol{\omega})\cdot q(\boldsymbol{\omega})d\boldsymbol{\omega} = q(\mathbf{y},\mathbf{z}|\mathbf{x}) \tag{7}$$

We will now prove that our framework actually recovers the total variance by plugging multiple stochastic forward passes with MC dropout in Equation 3.

Proof:

$$\begin{aligned}&\mathbb{E}_{q(\mathbf{y},\mathbf{z}|\mathbf{x})}\ (\mathbf{y}^T\mathbf{y}) \\ &\overset{(1)}{=}\int\Bigg(\ \int\mathbf{y}^T\mathbf{y}\cdot p(\mathbf{y},\mathbf{z}|\mathbf{x},\boldsymbol{\omega})\mathrm{d}\mathbf{y}\Bigg)q(\boldsymbol{\omega})\mathrm{d}\boldsymbol{\omega} \\ &\overset{(2)}{=}\int\Bigg(\ \text{Cov}_{p(\mathbf{y},\mathbf{z}|\mathbf{x},\boldsymbol{\omega})}(\mathbf{y}_{\boldsymbol{\omega}}) \\ &\qquad+\mathbb{E}_{p(\mathbf{y},\mathbf{z}|\mathbf{x},\boldsymbol{\omega})}(\mathbf{y}_{\boldsymbol{\omega}})^T\mathbb{E}_{p(\mathbf{y},\mathbf{z}|\mathbf{x},\boldsymbol{\omega})}(\mathbf{y}_{\boldsymbol{\omega}})\Bigg)\cdot q(\boldsymbol{\omega})\mathrm{d}\boldsymbol{\omega} \\ &\overset{(3)}{=}\int\Bigg(\ \text{Cov}_{p(\mathbf{y},\mathbf{z}|\mathbf{x},\boldsymbol{\omega})}(\mathbf{y}_{\boldsymbol{\omega}})+\mathbb{E}_{p(\mathbf{y},\mathbf{z}|\mathbf{x},\boldsymbol{\omega})}(\mathbf{y}_{\boldsymbol{\omega}})^T\mathbb{E}_{p(\mathbf{y},\mathbf{z}|\mathbf{x},\boldsymbol{\omega})}(\mathbf{y}_{\boldsymbol{\omega}})\Bigg) \\ &\qquad\cdot\text{Bern}(\mathbf{z}_1)\cdots\text{Bern}(\mathbf{z}_L)\mathrm{d}\mathbf{z}_1\cdots\mathrm{d}\mathbf{z}_L \\ &\overset{(4)}{=}\int\Bigg(\ \sigma_{data,\boldsymbol{\omega}}^2\mathbf{I}_D+\widehat{\mathbf{y}}(\mathbf{x},\mathbf{z}_1,...,\mathbf{z}_L)^T\widehat{\mathbf{y}}(\mathbf{x},\mathbf{z}_1,...,\mathbf{z}_L)\Bigg) \\ &\qquad\cdot\text{Bern}(\mathbf{z}_1)\cdots\text{Bern}(\mathbf{z}_L)\mathrm{d}\mathbf{z}_1\cdots\mathrm{d}\mathbf{z}_L \\ &\overset{(5)}{\approx}\frac{1}{T}\sum_{t=1}^{T}\hat{\sigma}_{data,t}^2+\frac{1}{T}\sum_{t=1}^{T}\widehat{\mathbf{y}}(\mathbf{x},\widehat{\mathbf{z}}_{1,t},...,\widehat{\mathbf{z}}_{L,t})^T\widehat{\mathbf{y}}(\mathbf{x},\widehat{\mathbf{z}}_{1,t},...,\widehat{\mathbf{z}}_{L,t})\end{aligned}$$

(1) follows by the definition of expected value.
(2) follows by the definition of covariance:

$$\text{Cov}(\mathbf{y}) = \mathbb{E}(\mathbf{y}^T\mathbf{y}) - \mathbb{E}(\mathbf{y})^T\,\mathbb{E}(\mathbf{y})$$

(3) follows from Equation 6.
(4) since $p(\mathbf{y},\mathbf{z}|\mathbf{x},\boldsymbol{\omega}) = \mathcal{N}\big(\mathbf{y};\widehat{\mathbf{y}}(\mathbf{x},\mathbf{z}_1,...,\mathbf{z}_L),\sigma_{data}^2\mathbf{I}_D\big)$.
(5) approximation by Monte Carlo integration.

Consequently, from the result just obtained and by the definition of variance, it can be easily shown that the total variance can be computed as:

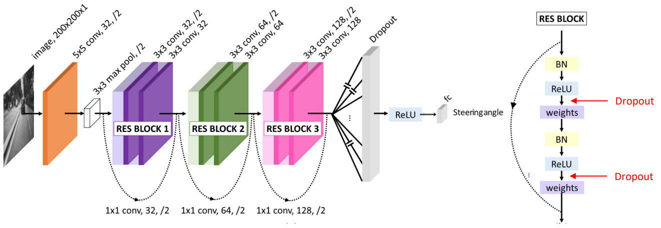


Fig. 3. Our dropout version of DroNet that leverages only the branch for steering angle prediction.

$$\begin{aligned}\text{Var}(\mathbf{y}) &= \mathbb{E}_{q(\mathbf{y},\mathbf{z}|\mathbf{x})}\left(\mathbf{y}^T\mathbf{y}\right) - \mathbb{E}_{q(\mathbf{y},\mathbf{z}|\mathbf{x})}(\mathbf{y})^T\,\mathbb{E}_{q(\mathbf{y},\mathbf{z}|\mathbf{x})}(\mathbf{y}) \\ &\approx \frac{1}{T}\sum_{t=1}^{T}\widehat{\mathbf{y}}(\mathbf{x},\widehat{\mathbf{z}}_{1,t},\dots,\widehat{\mathbf{z}}_{L,t})^T\widehat{\mathbf{y}}(\mathbf{x},\widehat{\mathbf{z}}_{1,t},\dots,\widehat{\mathbf{z}}_{L,t}) \\ &- \mathbb{E}_{q(\mathbf{y},\mathbf{z}|\mathbf{x})}(\mathbf{y})^T\,\mathbb{E}_{q(\mathbf{y},\mathbf{z}|\mathbf{x})}(\mathbf{y}) + \frac{1}{T}\sum_{t=1}^{T}\hat{\sigma}^2_{data,t}\end{aligned}$$

which amounts to the sum of the sample variance of T MC samples *(model uncertainty)* and the average of the corresponding *data variances* $\sigma^2_{data,t}$ returned by the ADF network. ■

The proof shows an important result: the *data variance* obtained with ADF and the *model variance* obtained with MC sampling applied on the ADF NN itself can indeed be summed.

In contrast with the uncertainty recovered in II-A, here the uncertainty has been made data-dependent. Moreover, we propose a Bayesian framework that can retrieve both model and data uncertainty without changing network loss and with a negligible increase in the number of parameters. This means that any existing network with weights trained according to a specific criterion (e.g. *MSE loss*) can be easily adapted to provide a total uncertainty estimate without time-consuming re-training and avoiding to change loss, which may cause precision loss. In conclusion, the final output of our framework is $[\mathbf{y}^*, \sigma^2_{total}]$, where $\mathbf{y}^*$ is the mean of the mean predictions $\{\hat{\mathbf{y}}_t\}_{t=1}^T$ collected over T stochastic forward passes. Data and model variance are summed to obtain the total variance, as we have shown to be correct in the proposed proof.

IV. Experiments

Our framework for uncertainty estimation is validated on a dropout version (Fig. 3) of DroNet, a CNN trained for steering angles prediction [1]. To take into account model uncertainty, we place Dropout layers before every weight layer. Also, DroNet was deployed to output both a prediction of the steering angle and a collision probability. We remove the branch for the collision probability evaluation and we train the resulting CNN on the center camera images of the publicly available **Udacity dataset**. This dataset presents images taken from camera sensors mounted on a car under different lightning and traffic conditions.

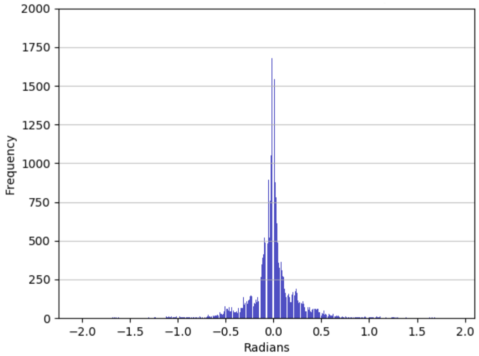


Fig. 4. Distribution of steering samples in training split of Udacity dataset.

We first train the non-Bayesian version of DroNet for the task of regressing steering angles and we evaluate the RMSE for its predictions. Afterwards, our framework is applied on the NN to compute mean predictions and variance estimates. Once we have collected uncertainty estimates, we need to solve the problem of evaluating them. Given that there is no ground-truth for uncertainty estimates of Neural Network predictions, we propose both a quantitative evaluation of our framework considering RMSE and log-likelihood, and a qualitative evaluation performing adversarial attacks on our model and showing that it is robust to adversarial noises in the range for the assumed input variance.

A. The Importance of Model and Data Uncertainty

We will now show that in life-critical tasks such as autonomous driving it is essential to take into account both model and data uncertainty.

By looking at the distribution of steering angles in the training dataset (Fig. 4), we can notice that the most represented samples are those with steering angles close to zero. Driving on straight lines is indeed the most natural situation, thus samples with higher steering angles are the least represented in training set. Hence, the model will tend to overfit on the most represented samples. Model uncertainty is in this case useful to evaluate how much a model is certain about its predictions and we expect our framework to return a high level of variance for images taken while the car is steering, since these are the cases on which the NN is trained the least. This can be seen from Fig. 5, where it is important to highlight that the prediction of our CNN is satisfactory for both images. Although the prediction is remarkably precise also for the *right image*, the model uncertainty is in this case much higher since the car is steering, and this is one of the least represented samples in the training data. Thus, we have just shown how model uncertainty is essential in this regression task to discriminate between images that are well-known from our model and other cases about which the model is less instructed. Although data uncertainty remains roughly constant for images where the car is either steering or not, it is still important in this task, because in some cases data uncertainty is higher than model

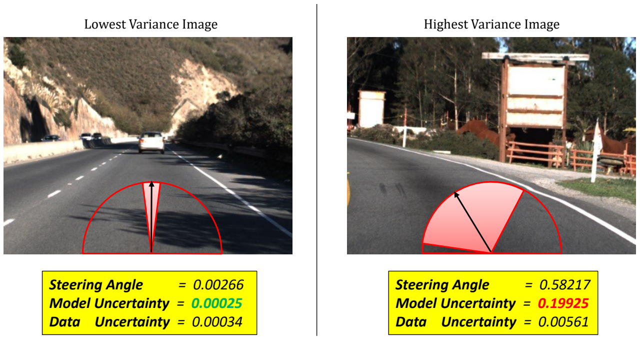

Lowest Variance Image
Steering Angle = 0.00266
Model Uncertainty = 0.00025
Data Uncertainty = 0.00034
Highest Variance Image
Steering Angle = 0.58217
Model Uncertainty = 0.19925
Data Uncertainty = 0.00561

Fig. 5. Comparison between a case with low model uncertainty *(left image)* and a case with high model uncertainty *(right image)*. The red slices represent one standard deviation from the mean prediction.

uncertainty *(left image)* and thus plays a fundamental role in total uncertainty estimation.

B. Quantitative Evaluation

To assess our estimate quantitatively, we compare the scores for log-likelihood and RMSE of our framework against other approaches previously presented. We choose to validate these methods comparing the average test log-likelihood on the dataset $D = \{\mathbf{x}, \mathbf{y}\}$

$$l(\hat{\mathbf{y}}, \boldsymbol{\sigma}^2_{\text{tot}}; \mathbf{y}) = \frac{1}{D} \sum_{i \in D} \Bigg(-\frac{1}{2}\log(2\pi) - \frac{1}{2}\log(\boldsymbol{\sigma}^2_{tot,i}) - \frac{1}{2\boldsymbol{\sigma}^2_{tot,i}}(\mathbf{y}_i - \hat{\mathbf{y}}_i)^2 \Bigg)$$

since it is a measure of how well the target fits to the output Gaussian distribution of our model.

Moreover, many cited methods to estimate uncertainty cause a drop in precision. We show that our method is generally better than others in terms of both RMSE and log-likelihood.

Method	Uncertainty	RMSE	Avg Test ll
MC Dropout	Model	0.088	0.7758
ADF	Data	0.1571	-4.2504
Heteroscedastic	Total	0.1144	1.1097
Ours	*Total*	*0.0987*	*1.0389*

TABLE I
Comparison between approaches frequently used in literature and our framework. *(T=1000 for sampling approaches)*

By looking at the log-likelihoods in Table I, it is possible to notice that our method outperforms the baselines for uncertainty estimation of MC Dropout and ADF, which consider respectively only model uncertainty and data uncertainty. The heteroscedastic NN combined with MC Dropout, as proposed by Kendall [12], is instead slightly better than our approach in terms of log-likelihood, since it is specifically trained to minimize the Negative Log-Likelihood of a Gaussian. However, the heteroscedastic NN approach comes with some limitations: the network needs to be re-trained under the *heteroscedastic loss* (1), causing a loss in precision. In contrast, our method, here tested with an *assumed input noise* of 10^{-3}, reports a lower RMSE than Kendall's approach. This is justified by the fact that our framework takes the standard NN trained on the desired MSE loss and applies both ADF and MC Dropout on it, without changing the weights of the NN. Furthermore, our method is also more precise than ADF alone. Only MC Dropout outperforms our approach in terms of RMSE, because it leverages the non-Bayesian NN trained for this task and collects multiple samples applying dropout at test time. The reason why our framework cannot reach the same precision as the standard neural network is that it loses precision in non-linear layers, which present mean-variance interaction.

As a trade-off to this loss in precision, we will show that our framework gains robustness to adversarial attacks [17].

C. Robustness to Adversarial Attacks

Our framework propagates input mean and variance through the NN, recovering a distribution over the output prediction. This distribution propagation is based on the assumption that the input is corrupted by white Gaussian noise, with variance σ_n^2, where σ_n^2 is a parameter that can either be selected to match the sensor noise reported on the sensor's data sheet or can be tuned to tackle adversarial attacks with adversarial noise comparable to the assumed input noise. The latter claim is confirmed by the results shown in Fig. 6. On the left are shown results computed with the standard NN, onto which MC Dropout is performed to recover a measurement of model uncertainty. On the right, results from our framework using the ADF NN are displayed. Concerning the *top-left* image pair, we see that the realistic-looking adversarial image generated with adversarial noise $\epsilon = 0.01$ is able to fool the neural network, which indeed reports a high uncertainty level. On the other side, by looking at the *top-right* image pair it is important to highlight how our framework, with an *assumed input noise* of 10^{-3}, is able to tackle such adversarial attack, reporting a surprising precision accompanied by a relatively low level of uncertainty. It is necessary to perform a more aggressive and unrealistic adversarial attack with $\epsilon = 0.1$ (*bottom-right image*) to fool our framework, which anyhow raises a high level of uncertainty for this wrong prediction, giving an ulterior feeling for robustness.

To further testify the robustness of our approach, we show how assuming an input noise of 0.1 is enough to contrast an adversarial image generated with adversarial noise $\epsilon = 0.1$. This remarkable result is shown in Fig. 7, where, for both the real and the adversarial image, our framework outputs precise predictions.

V. Discussion

In this work, we proposed a framework that estimates the total uncertainty by propagating a Gaussian distribution over input data that is suitable to any existing (possibly-trained) neural network. To do so, we used a layer-wise approximation *(ADF)* that allows to propagate uncertainties through a given network. By further placing dropout before every weight layer, it is possible to take into account also model uncertainty. The

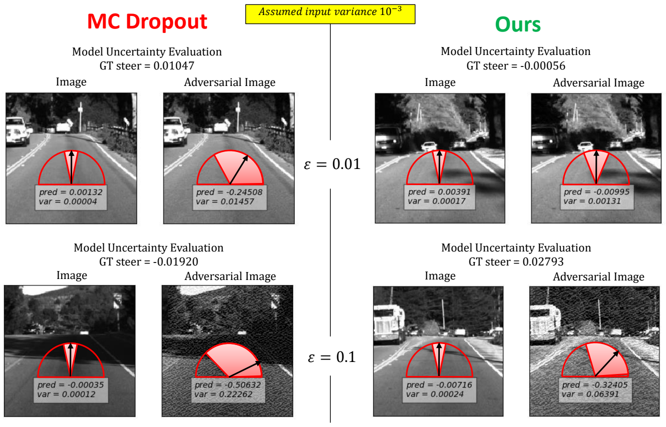


Fig. 6. Comparison between efficacy of adversarial attacks if applied on non-Bayesian NN *(left)* against efficacy of adversarial attacks performed on our framework *(right)*. An input variance of 10^{-3} is here assumed.

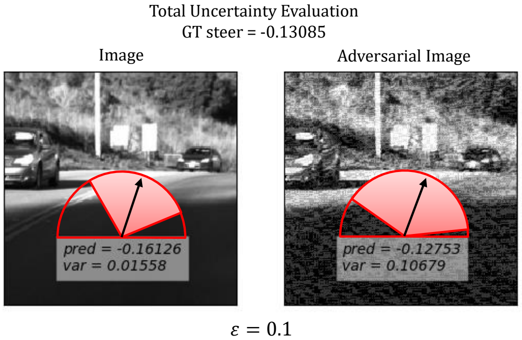


Fig. 7. Efficacy of our framework against strong and unrealistic adversarial attacks ($\epsilon = 0.1$) for assumed input noise $\sigma_n^2 = 0.1$, matching with the applied adversarial noise.

framework that we derived proved to be robust to adversarial attacks and noisy inputs, making an important step towards safe autonomous driving. A downside of our approach is precision loss [17] due to mistuned input variance or to numerical instability reasons, caused by near-zero divisions. However, when assuming a certain level of input noise, users can choose their preference between precision in predictions and robustness to larger noise. Also, it is necessary to perform MC Dropout to collect multiple stochastic forward samples from the NN to compute model uncertainty, introducing a high computational footprint. Nevertheless, according to our experiments conducted on this task with a GPU Nvidia GTX 1050Ti, we are able to collect at least 20 MC samples, which has proven to return satisfactory variance estimates. Finally, our framework works under the assumption that the full posterior distribution of the neural network is Gaussian, but it could be multi-modal. This is an approximation that we need to apply to tackle the intractability of the true posterior distribution; other approximations could be used though.

For future work, we propose to approximate and propagate the distribution over network weights in a similar fashion as ADF does with activation distributions; this approach may advance a lightweight alternative to the otherwise computationally expensive MC Dropout.

References

[1] A. Loquercio, A. I. Maqueda, C. R. D. Blanco, and D. Scaramuzza, “Dronet: Learning to Fly by Driving,” *IEEE Robotics and Automation Letters*, 2018.

[2] K. Lee, Z. Wang, B. I. Vlahov, H. K. Brar, and E. A. Theodorou, “Ensemble Bayesian Decision Making with Redundant Deep Perceptual Control Policies,” *ArXiv*, 2018.

[3] D. Feng, L. Rosenbaum, and K. Dietmayer, “Towards Safe Autonomous Driving: Capture Uncertainty in the Deep Neural Network For Lidar 3D Vehicle Detection,” *Proceedings of the 21st IEEE International Conference on Intelligent Transportation Systems*, 2018.

[4] G. Kahn, A. Villaflor, V. Pong, P. Abbeel, and S. Levine, “Uncertainty-Aware Reinforcement Learning for Collision Avoidance,” *ArXiv 1702.01182*, 2017.

[5] A. D. Kiureghian and O. Ditlevsen, “Aleatoric or epistemic? does it matter?,” *Structural Safety*, vol. 31, no. 2, pp. 105–112, 2009.

[6] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, “Weight Uncertainty in Neural Networks,” *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

[7] J. Denker and Y. LeCun, “Transforming neural-net output levels to probability distributions,” *Advances in Neural Information Processing Systems 3*, 1991.

[8] D. J. MacKay, “A practical Bayesian framework for backpropagation networks,” *Neural Computation*, vol. 4, no. 3, pp. 448–472, 1992.

[9] R. M. Neal, “Bayesian learning for neural networks,” *PhD thesis, University of Toronto*, 1995.

[10] Y. Gal and Z. Ghahramani, “Dropout as a Bayesian Approximation,” *Proceedings of the 33rd International Conference on Machine Learning*, 2015.

[11] Srivastava, Hinton, Krizhevsky, Sutskever, and Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *The Journal of Machine Learning Research*, vol. 15, no. 1, 2014.

[12] A. Kendall and Y. Gal, “What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?,” *31st Conference on Neural Information Processing Systems (NIPS 2017)*, 2017.

[13] Boyen and Koller, “Tractable inference for complex stochastic processes,” *InUAI*, pp. 33–42, 1998.

[14] Ghosh, D. Fave, and Yedidia, “Assumed Density Filtering Methods for Learning Bayesian Neural Networks,” *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*, 2016.

[15] J. Gast and S. Roth, “Lightweight Probabilistic Deep Networks,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[16] Frey and Hinton, “Variational learning in nonlinear Gaussian belief networks,” *Neural Comput.*, vol. 11, no. 1, pp. 193–213, 1999.

[17] Jin, Dundar, and Culurciello, “Robust convolutional neural networks under adversarial noise,” *InWorkshop Proceedings of the ICLR*, 2016.