MARGINALLY CALIBRATED RESPONSE DISTRIBUTIONS FOR END-TO-END LEARNING IN AUTONOMOUS DRIVING

BY CLARA HOFFMANN[a] AND NADJA KLEIN[b]

Chair of Statistics and Data Science, Humboldt-Universität zu Berlin, [a]*clara.c.hoffmann@gmail.com,* [b]*nadja.klein@hu-berlin.de*

End-to-end learners for autonomous driving are deep neural networks that predict the instantaneous steering angle directly from images of the street ahead. These learners must provide reliable uncertainty estimates for their predictions in order to meet safety requirements and to initiate a switch to manual control in areas of high uncertainty. However, end-to-end learners typically only deliver point predictions, since distributional predictions are associated with large increases in training time or additional computational resources during prediction. To address this shortcoming, we investigate efficient and scalable approximate inference for the deep distributional model of Klein, Nott and Smith (*J. Comput. Graph. Statist.* **30** (2021) 467–483) in order to quantify uncertainty for the predictions of end-to-end learners. A special merit of this model, which we refer to as implicit copula neural linear model (IC-NLM), is that it produces densities for the steering angle that are marginally calibrated, that is, the average of the estimated densities equals the empirical distribution of steering angles. To ensure the scalability to large n regimes, we develop efficient estimation based on variational inference as a fast alternative to computationally intensive, exact inference via Hamiltonian Monte Carlo. We demonstrate the accuracy and speed of the variational approach on two end-to-end learners trained for highway driving using the comma2k19 dataset. The IC-NLM is competitive with other established uncertainty quantification methods for end-to-end learning in terms of nonprobabilistic predictive performance and outperforms them in terms of marginal calibration for in-distribution prediction. Our proposed approach also allows the identification of overconfident learners and contributes to the explainability of black-box end-to-end learners by using the predictive densities to understand which steering actions the learner sees as valid.

1. Introduction. In recent years there have been immense advances in autonomous driving. Nevertheless, the progression from mere driver assistance to fully autonomous drivers still poses great safety challenges. Owing to the high costs associated with wrong decisions made by autonomous drivers, models have to be extremely accurate and safe in a wide range of driving scenarios. As a response to the safety and scalability issues of traditional autonomous driving systems, a new family of models, called end-to-end learners, has emerged in the last decade. End-to-end learners predict steering angles of a moving vehicle directly from images or videos of the street ahead using a single deep neural network (DNN). These models usually only provide a point prediction for the steering angle, thus making it difficult to quantify uncertainty or address potential overconfidence of a learner.

Obtaining accurate predictive uncertainty measures for the steering angles is essential to assessing the safety of an end-to-end learner reliably. Predictive uncertainty can also be leveraged to initiate a switch to human control in areas of high uncertainty to avoid crashes. Recently, there has been much progress in obtaining reliable uncertainty estimates for a

Key words and phrases. Calibration, deep neural network, implicit copula, neural linear models, probabilistic forecasting, uncertainty quantification, variational inference.

DNN prediction. In contrast, the vast training set sizes as well as temporal and computational limitations during prediction in autonomous driving pose challenges to these methods. As a solution, we propose obtaining predictive densities for the steering angle, building on the marginally calibrated deep distributional regression model developed by Klein, Nott and Smith (2021). We label this approach the implicit copula neural linear model (IC-NLM) to highlight its connection to the class of neural linear models (NLMs). NLMs comprise DNNs in which the last layer is augmented to a Bayesian linear regression model. The IC-NLM is based on the implicit copula (Section 5 of Nelsen (2006)) of a vector of transformed response variables that arises from an NLM. The resulting copula allows the modelling of highly flexible relations between the feature vector and the response densities. The copula is combined with a nonparametrically estimated marginal distribution for the observed response variable to ensure certain calibration properties.

Our work delivers two main contributions to the quantification of uncertainty in end-to-end learners. First, we develop a scalable version of the IC-NLM based on approximate estimation using variational inference (VI). The original version of the IC-NLM is founded on Markov chain Monte Carlo (MCMC) estimation which is plagued by convergence issues and long runtimes when applied to the dataset sizes typically used in end-to-end learning. MCMC is thus not a scalable option for realistic autonomous driving models with large training set sizes n. We verify the accuracy of the VI approach by comparing it to Hamiltonian Monte Carlo (HMC; Neal (2011)) in an empirical setting based on the comma2k19 autonomous driving data (Schafer et al. (2018)) with $n > 300{,}000$ observations. Second, we demonstrate that the IC-NLM is competitive with other established methods of obtaining predictive densities for end-to-end learners with respect to the in-distribution prediction in terms of nonprobabilistic predictive accuracy while providing marginal calibration. To this end, we train end-to-end learners in lane keeping on highways on the comma2k19 data and quantify the uncertainty of the steering angle predictions using the IC-NLM and a number of state-of-the-art benchmark methods. In addition to its competitive performance, the IC-NLM only requires a single forward pass and no sampling at prediction time, making it suitable for fast and reliable real-time prediction.

The remainder of the paper is structured as follows. We start out with the basic idea and development of end-to-end learners as well as the unique challenges encountered when quantifying uncertainty for their predictions in Section 2. We also introduce the data used to train our end-to-end learners in this section. In Section 3 we present the IC-NLM, starting with the definition of NLMs and continuing with reviewing the construction of the implicit copula version of the NLM. Section 4 develops both exact and approximate inference through HMC and VI to estimate the IC-NLM. Section 5 contains our detailed analysis to quantify uncertainty in end-to-end learners. In Section 6 we present our conclusions.

2. Background and challenges in end-to-end learning.

Traditionally, autonomous driving approaches involve modularized models that rely on object classification and object tracking paired with if-else behavior rules (Chen et al. (2015)). But these models still struggle to navigate vehicles autonomously without any human intervention. One reason for the failure of traditional models lies in the limited training sets. Providing sufficient training data is costly: object-detection models require bounding boxes for each object in a single training image. Only humans can provide such bounding boxes for the training data manually, making large training sets very expensive to produce. Small training sets, in turn, lead to driving models that are not robust toward the diversity of situations encountered on real-world roads.

Eliminating this manual annotation requirement opens new avenues for ultralarge training sets and motivates the evolution to so-called end-to-end learners. These models directly translate sensory inputs into driving instructions through a single, nonmodularized model.

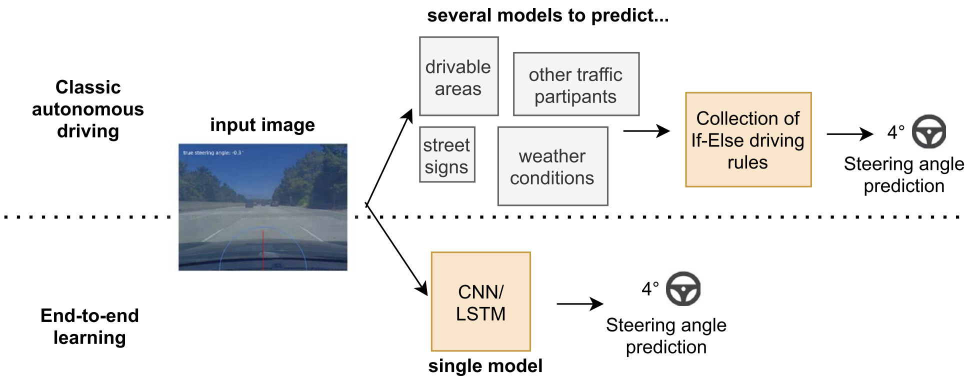

several models to predict...
Classic autonomous driving
input image
true steering angle: -0.3°
drivable areas
other traffic partipants
street signs
weather conditions
Collection of If-Else driving rules
4°
Steering angle prediction
End-to-end learning
CNN/ LSTM
single model
4°
Steering angle prediction

FIG. 1. *Classic autonomous driving models vs. end-to-end learning.*

Typically, the steering angle is predicted from images of the street ahead through a single DNN. The first end-to-end learners were convolutional neural networks (CNNs) that predict steering angles directly from the images of a forward-facing camera on a moving vehicle (Bojarski et al. (2016)). Nowadays, end-to-end learners have evolved to take spatial and time dimensions also into account (Xu et al. (2017), Chi and Mu (2017), Amini et al. (2019)) and use, for example, long short-term memory (LSTM) networks. The workflow underlying both the classic paradigm in autonomous driving and end-to-end learning is illustrated in Figure 1.

Formally, an end-to-end learner is a mapping from images of the road ahead to the steering angle. Let $\boldsymbol{Y} = (Y_1, \ldots, Y_n)^\top$ be a random vector of n steering angles from which we observe realizations $\boldsymbol{y} = (y_1, \ldots y_n)^\top$. The corresponding observed features $\boldsymbol{x}_i$, $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)^T$ are tensors that contain the image pixels. Typically, all pixels are stored in three-dimensional tensors. The dimensions correspond to position of the pixels along the width, height, and color channels of the image. An end-to-end learner is a mapping $f(\boldsymbol{x}_i) = \hat{y}_i$ that can be used to predict steering angles also from new feature values.

End-to-end learners typically only deliver point predictions $\hat{y}_i = E[Y_i \mid \boldsymbol{x}_i]$ without any uncertainty quantification. However, it is important to ensure the safety of end-to-end learners. Hence, one is interested in predicting not only $\hat{y}_i$ but rather the entire predictive density $p(y^* \mid \boldsymbol{x}^*)$ at an arbitrary steering angle y^*, given an arbitrary input image $\boldsymbol{x}^*$ (see Arnez et al. (2020), for an overview of existing methods). The latter can be used to identify areas of high uncertainty, initiate a switch to manual control, or act as an early alerting system for wrong predictions and crashes (Michelmore, Kwiatkowska and Gal (2018)). Overconfident end-to-end learners can also be identified through $p(y^* \mid \boldsymbol{x}^*)$, since they will associate wrong predictions with low predictive variance. Owing to its attractiveness in autonomous driving applications, obtaining suitable estimators for $p(y^* \mid \boldsymbol{x}^*)$ has recently become the topic of extensive research. However, most current methods exhibit computational difficulties or suffer from a lack of accuracy. The reason is that many methods for computing $p(y^* \mid \boldsymbol{x}^*)$ conflict with the unique hardware and prediction speed requirements in autonomous vehicles. Furthermore, existing methods often do not produce densities that are accurate, reliable, and consistent with the data as a result of unrealistic model assumptions.

Obtaining predictive densities for end-to-end learners in autonomous driving comes with unique challenges. To ensure safety, predictive densities should be well calibrated. Calibration is an essential criterion to ensure the reliability of predictive uncertainty. But currently, few methods based on DNNs produce calibrated estimates of $p(y^* \mid \boldsymbol{x}^*)$. We provide an overview of important calibration notions in Section 2.1. Another challenge is to obtain flexible predictive distributions $p(y^* \mid \boldsymbol{x}^*)$ for end-to-end learners at low cost. Predictive densities have to be complex (e.g., multimodal, skewed) to reflect the existence of several valid steering options. Training has to be relatively cheap, since large models with huge training sets are needed. Prediction not only has to be fast to ensure real-time results but also hardware-efficient, since only limited computational resources are available in an autonomous vehicle (Lin et al. (2018)). We discuss and investigate how well current methods for obtaining predictive densities in end-to-end learning handle these challenges in our analysis in Section 2.2 and compare them to the IC-NLM.

2.1. *Notions of calibration and reliability of predictive distributions.* In the machine learning literature, calibration is often understood as the reliability of prediction intervals. For the IC-NLM we will adhere to the more statistical notions of calibration introduced by Gneiting, Balabdaoui and Raftery (2007a), namely, *marginal* and *probabilistic* calibration. Reliable prediction intervals will arise as a natural by-product of these types of calibration, as we will see later.

Formally, marginal calibration can be expressed as follows: Assume that we compute predictive densities for the elements of a stochastic process $\{Y_1, Y_2, \ldots\}$. The corresponding probabilistic predictions are predictive cumulative distribution functions (CDFs) which are continuous, strictly increasing, and collected in a sequence $(\hat{F}_i)_{i\in\mathbb{N}}$. The true CDFs of the data-generating process are denoted as $(F_i)_{i\in\mathbb{N}}$. Following Gneiting, Balabdaoui and Raftery (2007a), marginal calibration of $(\hat{F}_i)_{i\in\mathbb{N}}$, relative to $(F_i)_{i\in\mathbb{N}}$, occurs if the asymptotic limits of the average true distribution and the average predictive CDFs exist and are equal to each other, that is,

$$\lim_{n\to\infty}\left(\frac{1}{n}\sum_{i=1}^{n} F_i(y)\right) = \lim_{n\to\infty}\left(\frac{1}{n}\sum_{i=1}^{n} \hat{F}_i(y)\right) \quad \forall y \in \mathbb{R}.$$

Note that in our case of cross-sectional data, there is no ordering in the sequence. However, as noted in Gneiting, Balabdaoui and Raftery (2007b), the above framework can still provide related empirical notions of marginal calibration. Gneiting and Ranjan (2013) develop such a notion and demonstrate that marginal calibration can be defined as equality of the marginal distribution of the observation and the expected forecast distribution. In the context of end-to-end learning, marginal calibration can be interpreted as a stable distribution over driving trajectories. For example, when training an end-to-end learner for lane-keeping on highways, we would expect that most steering trajectories are straight and no extreme turns are made, except when leaving the highway. Probabilistic predictions that are not marginally calibrated could place too much probability mass on extreme steering angles which is inconsistent with the distribution over steering angles that we observe in practice. We will see in Section 5 that the IC-NLM produces marginally calibrated predictive densities for in-distribution observations per construction.

Another notion of calibration is probabilistic calibration. The predictive densities are probabilistically calibrated if, on average, the probability that we observe $Y_i \leq y$ under the predictive CDF $\hat{F}_i$ converges almost surely to the probability of observing $Y_i \leq y$ under F_i, that is,

$$\frac{1}{n}\sum_{i=1}^{n} F_i \circ \hat{F}_i^{-1}(p) \overset{\text{a.s.}}{\to} p \quad \forall p \in (0, 1),$$

where $\overset{\text{a.s.}}{\to}$ denotes almost sure convergence. Even though the IC-NLM does not provide theoretical guarantees for probabilistic calibration, we will see in Section 5 that its predictions still perform quite well under this aspect.

Predictive densities can also be used to compute prediction intervals and identify potentially wrong predictions. Following the definition of Pearce et al. (2018), an $1-\alpha\%$ prediction interval is an interval $[\hat{y}_{i,\mathrm{LB}}, \hat{y}_{i,\mathrm{UB}}]$ such that an observation y_i falls into this interval with a probability of, at least, $1-\alpha\%$

$$P(\hat{y}_{i,\mathrm{LB}} \leq y_i \leq \hat{y}_{i,\mathrm{UB}}) \geq 1-\alpha.$$

The coverage rates of the prediction intervals should be accurate such that the width of the prediction interval at a given level can be used to quantify the uncertainty of the current prediction.

2.2. *Probabilistic models and uncertainty quantification.* In current methods for probabilistic predictions with DNNs, a trade-off exists between computational cost and prediction accuracy. Probabilistic models often possess a substantially higher number of parameters than their nonprobabilistic counterparts such that researchers are confronted with long computing times or increased hardware requirements. This is the case for, for example, Bayesian neural networks (BNNs) which learn probability distributions over all network weights. Even state-of-the art algorithms for estimating BNNs (Blundell et al. (2015)) lead to, at least, twice as many parameters as in a non-Bayesian DNN. In contrast, most non-Bayesian, ensemble-inspired approaches that estimate $p(y^* \mid \boldsymbol{x}^*)$ require several parallel DNN evaluations. Given that the hardware in autonomous vehicles is restricted, evaluating a sufficient number of DNNs in parallel is not always feasible. Some methods that suffer from these drawbacks are the ensemble approach of Lakshminarayanan, Pritzel and Blundell (2017), and Monte Carlo (MC) dropout, as proposed in Gal and Ghahramani (2016) and applied to autonomous driving by Amini et al. (2019), Michelmore, Kwiatkowska and Gal (2018). In addition, both (pseudo-) ensembles and BNNs do not generally produce calibrated probabilistic predictions empirically. This is often attributed to a lack of model expressivity and diversity (Kuleshov, Fenner and Ermon (2018), Zhang, Dalca and Sabuncu (2019)).

A few approaches are exempt from the aforementioned trade-off. Uncertainty of end-to-end learners can be quantified by discretizing the steering angle into bins (Xu et al. (2017), Chi and Mu (2017)). The regression problem is thereby turned into a classification problem. The resulting class probabilities can easily be used to quantify uncertainty but are often not calibrated (Guo et al. (2017)). Explanations of this problem can be found in Zhang, Dalca and Sabuncu (2019). Beyond that, distributional models that can capture aspects of the response distribution beyond the mean have evolved also in deep learning. For instance, heteroscedastic Gaussian models (Kendall and Gal (2017)), deep versions of quantile regression (Rodrigues and Pereira (2020)), or generalized linear models (Tran et al. (2020)) have been suggested. Mixture density networks (MDNs; Bishop (1994), Uria, Murray and Larochelle (2013)) also produce predictive densities, are easy to train, and require little modification from a nonprobabilistic DNN. The only drawback is that, empirically, the resulting predictive densities are often not calibrated.

2.3. *Data.* We demonstrate the IC-NLM's speed and accuracy by quantifying the uncertainty of end-to-end learners trained on the comma2k19 dataset (Schafer et al. (2018)). The data contains over 33 hours of driving footage on highways in California, collected in 2019 video files. A camera mounted on the windshield records the road ahead and several sensory inputs are measured simultaneously. The comma2k19 data are particularly suited to training an end-to-end learner, since lane markings on highways are clearly visible, making it easy for a CNN to use them as a steering orientation. Two example frames from the data are depicted in Figure 2.

To train a meaningful end-to-end learner, we clean a fifth[1] of the data of erratic driving behavior and lane changes. This is a typical step in autonomous driving, to ensure that the model

[1]Only a fifth of the full data are used since manual cleaning is extremely time consuming.

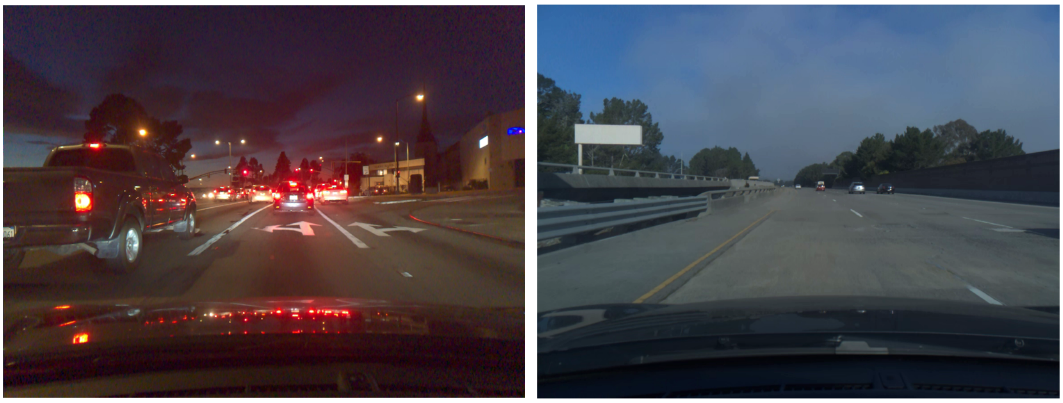


FIG. 2. *Example images from the comma2k19 dataset.*

does not learn faulty driving behavior such as minor swerving. After cleaning, the data are split into 43,736 training and 10,472 validation observations. In real-life autonomous driving applications, the datasets are usually much larger. To explore the scalability of VI estimation for the IC-NLM, we also use the full, uncleaned comma2k19 data. This large dataset is divided into a training and validation set with 355,543 training and 155,386 validation observations. However, with these data a distribution shift naturally occurs between the training and validation components. The IC-NLM is not designed for out-of-distribution prediction, as the marginal density of the training data will influence all predictive densities. Any shift between the validation and training distributions will, therefore, negatively impact predictive performance. For this reason we only use the uncleaned data to illustrate the computational gains of VI over HMC for the IC-NLM. Employing VI, instead of HMC, reduces the computation time by several days when using the full data.

Much interest lies in directly using raw data to train end-to-end learners (compare, e.g., Xu et al. (2017)), because cleaning datasets of erratic or unwanted driving behavior is extremely time consuming. However, training on raw data requires semisupervised methods, which are not the scope of this paper and which have been only partly developed so far in the context of end-to-end learning (see Abbasi et al. (2020), for an overview). In addition, semisupervised models cannot generalize accurately enough to be used in real-world scenarios. Therefore, hand-picking training examples from the raw data is still an essential step in training reliable end-to-end learners.

3. The implicit copula neural linear model.

3.1. *Primer on DNNs.* Typically, end-to-end learners for autonomous driving are DNNs that map from images of the road ahead directly to the steering angle. Generally, DNNs can be used to approximate arbitrary continuous functions $f^*(\boldsymbol{X}^*) = Y^*$ mapping a random vector of features $\boldsymbol{X}^*$ (not necessarily tabular but, e.g., images or text) to a scalar response Y^*. DNNs comprise layers of neurons, where each neuron receives the neuron activations from the previous layer as inputs. The neurons of the network jointly implement a complex nonlinear mapping from the input to the output through weight matrices of linear transformations and nonlinear activation functions. This mapping is learned from the data by adapting the weights of each neuron using a technique called error backpropagation (Rumelhart, Hinton and Williams (1986)).

For end-to-end learners the last layers of a DNN typically consist of fully connected, dense layers. The output $f^{(k)}$, $k \in \{1, \dots, K\}$ of the kth layer of such a DNN for a single observation with feature $\boldsymbol{x}^*$ may be represented as

$$f^{(k)}(\boldsymbol{x}^*) = g^{(k)}(\boldsymbol{Z}^{(k)}(\boldsymbol{x}^*)\boldsymbol{W}^{(k)} + \boldsymbol{b}^{(k)}),$$

where $\boldsymbol{Z}^{(k)}(\boldsymbol{x}^*)$ are the outputs of the previous (i.e., $(k-1)$th) layer, $\boldsymbol{W}^{(k)}$ is a weight matrix, $\boldsymbol{b}^{(k)}$ is a bias vector, and $g^{(k)}$ is a (non)linear activation function. The weights and biases of an entire K-layer DNN consist of $\{\boldsymbol{\zeta}, \boldsymbol{W}^{(K)}, \boldsymbol{b}^{(K)}\}$, with $\boldsymbol{\zeta}$ denoting the set of all weights and biases of the DNN up to the last hidden layer. All weights and biases are determined by minimizing an empirical loss criterion, such as the mean squared error (MSE) loss, $\sum_{i=1}^{n}(f^{(K)}(\boldsymbol{x}_i) - y_i)^2$, based on a training set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$ of features and responses (we refer interested readers to Goodfellow, Bengio and Courville (2016), Polson and Sokolov (2017), for a further discussion on how to determine the weights of a DNN and more information on DNNs, including regularization). In a regression setting, minimization based on the MSE is equivalent to assuming a homoscedastic Gaussian model (thus assuming that the responses are indeed conditionally Gaussian), although minimization does not require parametric model assumptions.

3.2. *Neural linear models.* The class of NLMs comprises Bayesian linear models in which the features are deep basis functions learned by a DNN with an identity activation function in the last layer (Ober and Rasmussen (2019), Pinsler et al. (2019), Snoek et al. (2015), Riquelme, Tucker and Snoek (2018)). Assume that we have trained a DNN on a training set $\mathcal{D}$. We denote the $n \times p$ matrix of outputs from the $(K-1)$th layer as $\boldsymbol{B}_\zeta(\boldsymbol{x}) = [\boldsymbol{\psi}_\zeta(\boldsymbol{x}_1) \mid \ldots \mid \boldsymbol{\psi}_\zeta(\boldsymbol{x}_n)]^T \in \mathbb{R}^{n\times p}$, where $\boldsymbol{\psi}_\zeta(\cdot)$ is the vector of p basis functions defined by the $(K-1)$th layer. Then, the NLM is of the form

$$\boldsymbol{y} = \boldsymbol{B}_\zeta(\boldsymbol{x})\boldsymbol{\beta} + \beta_0 + \boldsymbol{\varepsilon}, \tag{1}$$

where $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^\top \in \mathbb{R}^p$ is the vector of regression coefficients (or weights), $\boldsymbol{\varepsilon} = (\varepsilon_1, \ldots \varepsilon_n)^\top$ is the vector of i.i.d. error terms, $\boldsymbol{\varepsilon} \sim N(\boldsymbol{0}, \sigma^2\boldsymbol{I})$, and $\beta_0 \in \mathbb{R}$ is the intercept (also called bias in the machine learning literature). Both the vector of weights $\boldsymbol{\beta}$ and the bias β_0 are equipped with a prior in an NLM. This way it is possible to perform Bayesian inference over the model's parameters and compute the posterior of $\boldsymbol{\beta}$ and, subsequently, the predictive densities $p(y^* \mid \boldsymbol{x}^*)$ via the posterior predictive densities (see Section 3.4.1).

Most of the existing NLMs use simple, conjugate priors which, in turn, results in insufficiently complex, unimodal predictive densities $p(y^* \mid \boldsymbol{x}^*)$. These densities are not appropriate for modeling complex uncertainty scenarios in autonomous driving. The IC-NLM overcomes this issue by employing an implicit copula. The latter allows the whole distribution of the steering angle to vary flexibly with the features. Construction of the IC-NLM in the next section will be based on an NLM, and in this context, we will also give details on prior specifications and posterior inference.

3.3. *Derivation of the implicit copula neural linear model.* To ensure marginally calibrated densities, Klein and Smith (2019) and Smith and Klein (2021) introduce a new approach to distributional regression that uses a copula decomposition constructed from the so-called inversion method (Nelsen (2006)). Klein, Nott and Smith (2021) outline how to extend their approach to deep learning regression, which we refer to as the IC-NLM. We review the key ideas of this method and the adaptations necessary for our application in the following.

3.3.1. *Calibrated copula process.* Copula models with regression margins for multivariate responses $\boldsymbol{Y} \in \mathbb{R}^D$, $D > 1$ have been widely used in the literature (see, e.g., Craiu and Sabeti (2012), Klein and Kneib (2016a), Pitt, Chan and Kohn (2006), Song, Li and Yuan (2009)). However, another use of a copula with regression data is to capture the dependence between multiple observations on a single dependent variable Y, conditional on the feature values, which in our case are the steering angles and corresponding feature image tensors.

Doing so defines a *copula process* (Wilson and Ghahramani (2010)) on the feature space, which Smith and Klein (2021) call a *regression copula* because of its dependence on the features. When combined with a flexible marginal distribution for Y^*, the authors illustrate that it specifies a tractable and scalable distributional (i.e., probabilistic) regression model in which the entire distribution of Y^* varies with the features.

Klein, Nott and Smith (2021) use this idea when the regression copula is the implicit copula of the joint distribution of a *pseudo* response. Such pseudo responses can be modeled with a DNN regression. As a consequence, the pseudo responses can be obtained through density transformations from the observed response $\boldsymbol{Y}$, as detailed in Section 3.3.2 below. We shall now explain the copula construction and its relation to other statistical probabilistic models.

Sklar's theorem (Sklar (1959)) states that the n-dimensional distribution of $\boldsymbol{Y} \mid \boldsymbol{x}$ can be written as

$$p(\boldsymbol{y} \mid \boldsymbol{x}) = c^{\dagger}\big(F_{Y_1}(y_1 \mid \boldsymbol{x}_1), \dots, F_{Y_n}(y_n \mid \boldsymbol{x}_n) \mid \boldsymbol{x}\big) \prod_{i=1}^{n} p_{Y_i}(y_i \mid x_i),$$

where the n-dimensional copula with density $c^{\dagger}$ is a copula process on the covariate space and $F_{Y_i}(y_i \mid \boldsymbol{x}_i)$ is the marginal distribution function of $Y_i \mid \boldsymbol{x}_i$ with density p_{Y_i}. Both the copula and the marginal distributions are typically unknown, and it is common to make the copula dependent on some copula parameters $\boldsymbol{\theta}$. We follow Klein, Nott and Smith (2021) and use $c_{\text{DNN}}(\boldsymbol{u} \mid \boldsymbol{x}, \boldsymbol{\theta})$ with $u_i = F_{Y_i}(y_i)$ (cf. Section 3.3.2).

One further tractable but effective simplification is to allow the covariates only to affect the dependent variable through the copula function (Klein and Smith (2019)) and to calibrate the distribution of $Y_i \mid x_i$ to its invariant margin so that $F_{Y_i} \equiv F_Y$ has density $p_{Y_i}(y_i \mid x_i) = p_Y(y_i)$. This margin can be estimated nonparametrically. Thus, in the IC-NLM the response has the joint density

$$p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}) = c_{\text{DNN}}(u_1, \dots, u_n \mid \boldsymbol{x}, \boldsymbol{\theta}) \prod_{i=1}^{n} p_Y(y_i). \tag{2}$$

The marginal invariance assumption may seem counter-intuitive at first, given the usual conditional formulation of a regression model for $Y_i \mid \boldsymbol{x}_i$ through a univariate marginal model. Still, it is a valid approach for which $\boldsymbol{Y}$ is indeed dependent on $\boldsymbol{x}$ in the joint distribution defining a flexible distributional regression model; see Smith and Klein (2021) for details.

3.3.2. *Copula construction.* The basis for the construction of the IC-NLM of Klein, Nott and Smith (2021) is a DNN trained to predict density-transformed pseudo responses $\boldsymbol{Z}$, where $z_i = \Phi_1^{-1}(F_Y(y_i))$ with an identity activation function in the last layer. Let $\tilde{Z}_i$ be the ith pseudo response, $i = 1, \dots, n$. Then, $\tilde{Z}_i$ can be modeled using the output layer of the DNN plus some Gaussian noise $\varepsilon_i \sim N(0, \sigma^2)$, that is, $\tilde{Z}_i = f_i^{(K)} + \varepsilon_i$, where $f_i^{(K)}$ denotes the output when passing the ith feature observation to the DNN. Then, from (1) the vector $\tilde{\boldsymbol{Z}} = (\tilde{Z}_1, \dots, \tilde{Z}_n)^{\top}$ follows the linear model

$$\tilde{\boldsymbol{Z}} = \boldsymbol{B}_{\zeta}(\boldsymbol{x})\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \tag{3}$$

where the intercept β_0 was excluded, since it will not be identified in the copula. The weights $\boldsymbol{\zeta}$ can be easily obtained by training the DNN to predict the pseudo response (see Section 4 for details on how to fix them). Thus, (3) is a linear model with design matrix $\boldsymbol{B}_\zeta$ and vector of weights $\boldsymbol{\beta}$. Efficient estimates for $\boldsymbol{\beta}$ are produced via regularization with the conditionally Gaussian prior as a shrinkage prior, that is,

$$\boldsymbol{\beta} \mid \boldsymbol{\theta}, \sigma^2 \sim N\big(\mathbf{0}, \sigma^2 \boldsymbol{P}(\boldsymbol{\theta})^{-1}\big).$$

The (sparse) precision matrix $\boldsymbol{P}(\boldsymbol{\theta})$ is a function of the copula parameters $\boldsymbol{\theta}$. The density c_{DNN} is then derived by integrating $\boldsymbol{\beta}$ out in (3) (see Klein and Smith (2019), for a detailed derivation). This results in a Gaussian copula (Song (2000)) with density

$$c_{\mathrm{DNN}}(\boldsymbol{u} \mid \boldsymbol{x}, \boldsymbol{\theta}) = \frac{p(\boldsymbol{z} \mid \boldsymbol{x}, \sigma^2, \boldsymbol{\theta})}{\prod_{i=1}^n p(z_i \mid \boldsymbol{x}, \sigma^2, \boldsymbol{\theta})} = \frac{\phi_n(\boldsymbol{z}; \mathbf{0}, \boldsymbol{R}(\boldsymbol{x}, \boldsymbol{\theta}))}{\prod_{i=1}^n \phi_1(z_i)}, \tag{4}$$

where

$$\boldsymbol{R}(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{S}(\boldsymbol{x}, \boldsymbol{\theta})\big(\boldsymbol{I} + \boldsymbol{B}_\zeta(\boldsymbol{x})\boldsymbol{P}(\boldsymbol{\theta})^{-1}\boldsymbol{B}_\zeta(\boldsymbol{x})^\top\big)\boldsymbol{S}(\boldsymbol{x}, \boldsymbol{\theta}), \tag{5}$$

$z_i = \Phi_1^{-1}(u_i)$, $\boldsymbol{z} = (z_1, \dots, z_n)^\top$, and $\phi_n(\cdot; \mathbf{0}, \boldsymbol{R})$ and ϕ_1 are the densities of $N_n(\mathbf{0}, \boldsymbol{R})$ and $N(0, 1)$ distributions, respectively. The random variables Z_i are standardized versions of $\tilde{Z}_i$, $\boldsymbol{Z} = (Z_1, \dots, Z_n)^\top = \sigma^{-1}\boldsymbol{S}(\boldsymbol{x}, \boldsymbol{\theta})\tilde{\boldsymbol{Z}}$, where $\boldsymbol{S}(\boldsymbol{x}, \boldsymbol{\theta}) = \mathrm{diag}(s_1, \dots, s_n)$ is a diagonal scaling matrix with elements $s_i = (1 + \boldsymbol{\psi}_\zeta(\boldsymbol{x}_i)^\top \boldsymbol{P}(\boldsymbol{\theta})^{-1}\boldsymbol{\psi}_\zeta(\boldsymbol{x}_i))^{-1/2}$, which ensures that $Z_i \mid \boldsymbol{x}, \sigma^2, \boldsymbol{\theta} \sim N(0, 1)$.

3.3.3. *Shrinkage for deep regression copulas.* For specific choices of $\boldsymbol{P}(\boldsymbol{\theta})$, we compare the two shrinkage priors employed in Klein, Nott and Smith (2021), namely, the ridge and the horseshoe prior:

Ridge. The ridge prior is one of the simplest forms of shrinkage priors, where $\beta_j \mid \tau^2 \sim N(0, \tau^2)$, $j = 1, \dots, p$ and a hyperprior is used on the variance τ^2. For this we employ the robust and principled choice of scale-dependent priors of Klein and Kneib (2016b), which corresponds to a Weibull prior, $\tau^2 \sim \mathrm{WB}(1/2, \nu)$ with scale parameter ν. We found the predictive densities to be rather robust with respect to the actual value of ν, and we follow Klein, Nott and Smith (2021) and set $\nu = 2.5$ in our analysis.

Horseshoe. The horseshoe prior is attractive owing to its robustness, local adaptivity, and analytical properties (Carvalho, Polson and Scott (2010)). It is a scale mixture of the hierarchical form $\beta_j \mid \lambda_j \sim N(0, \lambda_j^2)$, with $\pi_0(\lambda_j \mid \tau) = \text{Half-Cauchy}(0, \tau^2)$ and $\pi_0(\tau) = \text{Half-Cauchy}(0, 1)$. Compared to the ridge prior, this prior is a global-local shrinkage prior which allows for weight-specific local shrinkage for each coefficient in addition to overall regularization through τ^2.

Even though the dimension of $\boldsymbol{\beta}$ is quite small in our application ($p = 10$, see Section 5), regularizing the weights can make the posterior response densities more robust to noise and enhance the predictive quality, as we demonstrate in Section 5. While the ridge prior introduces only one further scalar parameter $\boldsymbol{\theta} = \{\tau^2\}$, the horseshoe prior comes with $p + 1$ hyperparameters $\boldsymbol{\theta} = \{\lambda_1, \dots, \lambda_p, \tau^2\}$.

The corresponding correlation matrices for both priors are

$$\boldsymbol{R}(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{S}(\boldsymbol{x}, \boldsymbol{\theta})\big(\boldsymbol{I} + \tau^2 \boldsymbol{B}_\zeta(\boldsymbol{x})\boldsymbol{B}_\zeta(\boldsymbol{x})^\top\big)\boldsymbol{S}(\boldsymbol{x}, \boldsymbol{\theta}),$$

$$\boldsymbol{R}(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{S}(\boldsymbol{x}, \boldsymbol{\theta})\big(\boldsymbol{I} + \boldsymbol{B}_\zeta(\boldsymbol{x})\,\mathrm{diag}(\boldsymbol{\lambda})^2 \boldsymbol{B}_\zeta(\boldsymbol{x})^\top\big)\boldsymbol{S}(\boldsymbol{x}, \boldsymbol{\theta}).$$

Allowing for more sophisticated shrinkage of each regression coefficient introduces additional parameters to the model which, in turn, can slow down estimation. However, our results reveal that allowing for local shrinkage in addition to global regularization significantly enhances predictive performance and the model's ability to identify potentially high-error predictions. This is why we argue that employing a more complex shrinkage prior through the horseshoe is worth the additional computational cost. The full hierarchical IC-NLM is illustrated as a graphical model in Figure 3.

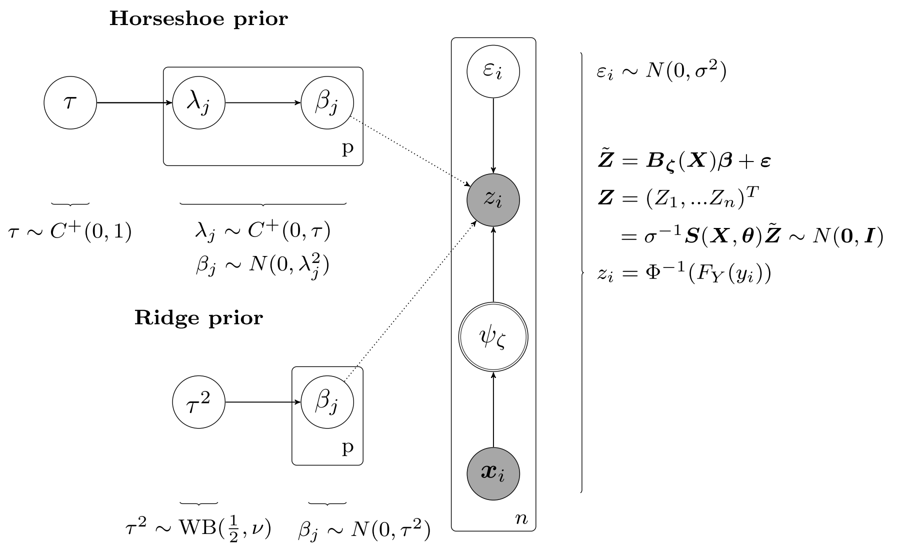

Horseshoe prior
τ
λ_j
β_j
p
$\tau \sim C^+(0,1)$
$\lambda_j \sim C^+(0,\tau)$
$\beta_j \sim N(0,\lambda_j^2)$
Ridge prior
τ^2
β_j
p
$\tau^2 \sim \mathrm{WB}(\frac{1}{2},\nu)$
$\beta_j \sim N(0,\tau^2)$
ε_i
z_i
ψ_ζ
$\boldsymbol{x}_i$
n
$\varepsilon_i \sim N(0,\sigma^2)$
$\tilde{\boldsymbol{Z}} = \boldsymbol{B}_\zeta(\boldsymbol{X})\boldsymbol{\beta} + \boldsymbol{\varepsilon}$
$\boldsymbol{Z} = (Z_1, ...Z_n)^T$
$= \sigma^{-1}\boldsymbol{S}(\boldsymbol{X},\boldsymbol{\theta})\tilde{\boldsymbol{Z}} \sim N(\boldsymbol{0},\boldsymbol{I})$
$z_i = \Phi^{-1}(F_Y(y_i))$

FIG. 3. *The IC-NLM as a graphical model. Dotted arrows indicate that only one of the dependencies can hold at a time, that is, either the ridge or horseshoe prior.*

3.4. *Estimation of the IC-NLM.* Algorithm 1 summarizes how estimation of the IC-NLM is realized.

Step 1 involves estimating the marginal distribution $\hat{F}_Y$ via a kernel density estimator (KDE). Later, we will use a nonparametric KDE with a Gaussian kernel (Racine (2008)); see Section 5. Steps 2 and 3 are dependent upon the choice of architecture, which we discuss later in Section 5. Step 4 requires evaluation of the likelihood which is given by the copula decomposition at (2). To do so directly requires evaluation of the copula density at (4) which is computationally infeasible, in general, because of the need to invert the $n \times n$ matrix $\boldsymbol{R}$. Klein and Smith (2019) solve this problem by instead using the likelihood conditional also on $\boldsymbol{\beta}$, which is

$$
\begin{aligned}
p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\beta}, \boldsymbol{\theta}) &= p(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{\beta}, \boldsymbol{\theta}) \prod_{i=1}^{n} \frac{p_Y(y_i)}{\phi_1(z_i)} \\
&= \phi_n\big(\boldsymbol{z}; S(\boldsymbol{x},\boldsymbol{\theta}) B_\zeta(\boldsymbol{x})\boldsymbol{\beta}, S(\boldsymbol{x},\boldsymbol{\theta})^2\big) \prod_{i=1}^{n} \frac{p_Y(y_i)}{\phi_1(z_i)},
\end{aligned}
\tag{6}
$$

Algorithm 1: Estimation of the IC-NLM

```
    Input : Features $\boldsymbol{x}$, responses $\boldsymbol{y}$ for training and new features $\boldsymbol{x}_0$ for prediction.
    Output: Posterior densities for the response $\hat{p}(y_0 \mid \boldsymbol{x}_0)$
1   Estimate the empirical density $\hat{F}_Y$ from the observed responses $\boldsymbol{y}$ and transform $\boldsymbol{y}$ to $\boldsymbol{z}$
    via $z_i = \Phi_1^{-1}(\hat{F}_Y(y_i))$.
2   Train a $K$-layer DNN with transformed responses $\boldsymbol{z}$ as targets and $\boldsymbol{x}$ as inputs, and
    denote the trained weights of all layers except the last hidden layer as $\boldsymbol{\zeta}$; see Section 5.1
    and Supplementary Material B.1 (Hoffmann and Klein (2023)).
3   Predict the outputs $\boldsymbol{B}_\zeta(\boldsymbol{x})$ of the $(K-1)$th layer of the DNN by forward-passing the
    features $\boldsymbol{x}$ through the model up to the $(K-1)$th layer.
4   Estimate $p(\boldsymbol{\beta}, \boldsymbol{\theta} \mid \boldsymbol{x}, \boldsymbol{y})$ by MCMC or VI, as described in Section 4.
5   Compute posterior predictive densities $\hat{p}(y_0 \mid \boldsymbol{x}_0)$ as described in Section 3.4.1.
```

and can be evaluated in $O(n)$ operations because $S(\boldsymbol{x}, \boldsymbol{\theta})$ is diagonal. We present exact and approximate Bayesian inference for this step in Section 4. Computation of Step 5 is based on (6) and described next. All steps are implemented in Python, and the code is available on GitHub.

3.4.1. *Predictive densities and beyond.* The desired predictive uncertainty for the steering angle at new feature values $\boldsymbol{x}_0$ is based on the posterior predictive density $p(y_0 \mid \boldsymbol{x}_0, \boldsymbol{x}, \boldsymbol{y})$ of the IC-NLM, which, for a new value y_0, is given by

$$p(y_0 \mid \boldsymbol{x}_0, \boldsymbol{x}, \boldsymbol{y}) = \int p(y_0 \mid \boldsymbol{x}_0, \boldsymbol{x}, \boldsymbol{\beta}, \boldsymbol{\theta}) p(\boldsymbol{\beta}, \boldsymbol{\theta} \mid \boldsymbol{x}, \boldsymbol{y})\, d(\boldsymbol{\beta}, \boldsymbol{\theta}), \tag{7}$$

where we use $p(y_0 \mid \boldsymbol{x}_0, \boldsymbol{x}, \boldsymbol{\beta}, \boldsymbol{\theta})$ instead of $p(y_0 \mid \boldsymbol{x}_0, \boldsymbol{x}, \boldsymbol{\theta})$ to avoid direct computation of $\boldsymbol{R}(\boldsymbol{x}, \boldsymbol{\theta})$. If $z_0 = \Phi_1^{-1}(F_Y(y_0))$, then $|\frac{dz_0}{dy_0}| = \frac{p_Y(y_0)}{\phi_1(z_0)}$, and by changing variables from y_0 to z_0, estimates $\hat{p}$ of (7) can be obtained via

$$\hat{p}(y_0 \mid \boldsymbol{x}_0) = \frac{p_Y(y_0)}{\phi_1(z_0)} \hat{p}(z_0 \mid \boldsymbol{x}_0, \boldsymbol{x}, \boldsymbol{y}) = \frac{p_Y(y_0)}{\phi_1(\Phi_1^{-1}(F_Y(y_0))} \frac{1}{\hat{s}_0} \phi_1\left(\frac{\Phi_1^{-1}(F_Y(y_0)) - \hat{m}(\boldsymbol{x}_0)}{\hat{s}_0}\right).$$

Here, $\hat{m}(\boldsymbol{x}_0) = \hat{s}_0 \boldsymbol{\psi}_\zeta(\boldsymbol{x}_0)^T \hat{\boldsymbol{\beta}}$ and $\hat{s}_0 = (1 + \boldsymbol{\psi}_\zeta(\boldsymbol{x}_0)^\top \boldsymbol{P}(\hat{\boldsymbol{\theta}})^{-1} \boldsymbol{\psi}_\zeta(\boldsymbol{x}_0))^{-1/2}$ based on the posterior mean VI estimates $\hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\theta}}$ in case of approximate posterior estimation (see Section 4.2) or from the MCMC output $\{(\boldsymbol{\beta}^{(1)}, \boldsymbol{\theta}^{(1)}), \dots, (\boldsymbol{\beta}^{(J)}, \boldsymbol{\theta}^{(J)})\}$ via $\hat{m}(\boldsymbol{x}_0) = \boldsymbol{\psi}_\zeta(\boldsymbol{x}_0)^T (\frac{1}{J} \sum_{j=1}^J s_0^{[j]} \times \boldsymbol{\beta}^{[j]})$ and $\hat{s}_0 = \frac{1}{J} \sum_{j=1}^J s_0^{[j]}$ in case of exact posterior estimation (see Section 4.1). Note that $\hat{m}(\boldsymbol{x}_0)$ is an estimator for $\mathbb{E}[Z_0 \mid \boldsymbol{x}_0, \boldsymbol{x}, \boldsymbol{y}]$.

Later in our analysis, we will also make use of the posterior predictive variance, which is $\operatorname{Var}(Y_0 \mid \boldsymbol{x}_0, \boldsymbol{x}, \boldsymbol{y}) = \mathbb{E}[Y_0^2 \mid \boldsymbol{x}_0, \boldsymbol{x}, \boldsymbol{y}] - \mathbb{E}[Y_0 \mid \boldsymbol{x}_0, \boldsymbol{x}, \boldsymbol{y}]^2$, with posterior predictive mean

$$\begin{aligned}\mathbb{E}[Y_0 \mid \boldsymbol{x}_0, \boldsymbol{x}, \boldsymbol{y}] &= \int \mathbb{E}[Y_0 \mid \boldsymbol{x}_0, \boldsymbol{x}, \boldsymbol{\beta}, \boldsymbol{\theta}] p(\boldsymbol{\beta}, \boldsymbol{\theta} \mid \boldsymbol{x}, \boldsymbol{y})\, d(\boldsymbol{\beta}, \boldsymbol{\theta}) \\ &= \int F_Y^{-1}(\Phi_1(z_0)) \frac{1}{s_0} \phi_1\left(\frac{z_0 - s_0 \boldsymbol{\psi}_\zeta(\boldsymbol{x}_0)^\top \boldsymbol{\beta}}{s_0}\right) dz_0,\end{aligned}$$

and

$$\mathbb{E}[Y_0^2 \mid \boldsymbol{x}_0, \boldsymbol{x}, \boldsymbol{y}] = \int (F_Y^{-1}(\Phi_1(z_0)))^2 \frac{1}{s_0} \phi_1\left(\frac{z_0 - s_0 \boldsymbol{\psi}_\zeta(\boldsymbol{x}_0)^\top \boldsymbol{\beta}}{s_0}\right) dz_0.$$

Both integrals involve univariate numerical integration only. Estimators for the integrands are obtained in the same fashion as for the predictive densities. If the true steering angle is available, the prediction error can be measured by the squared or absolute deviation from $\hat{\mathbb{E}}(Y_0 \mid \boldsymbol{x}_0, \boldsymbol{x}, \boldsymbol{y})$. Prediction intervals can also easily be obtained from the predictive densities. Usually, symmetric prediction intervals are used where the lower bound is the response value at which the predictive CDF is smaller than $\alpha/2$ and the upper bound is the response value at which the CDF is larger than $1 - \alpha/2$.

4. Exact and approximate posterior estimation.

In this section we present a stable MCMC scheme for posterior estimation of the IC-NLM which is applicable for moderately sized datasets. It is based on HMC, since the MCMC sampler of Klein, Nott and Smith (2021), based on Gibbs and Metropolis-Hastings updates, was too sticky to produce reliable results in our application. Next, we develop a VI approach as an approximate alternative for posterior estimation with large-scale datasets and highly parameterized models. Even though MCMC delivers estimates in both of our data scenarios, it is plagued by slow convergence and long runtimes as n grows large. This is an issue for autonomous driving applications in which sample sizes can become extremely large. The VI approach is much faster than MCMC and sufficiently accurate, as we reveal in Section 5. In the following we denote $\boldsymbol{\vartheta} = \{\boldsymbol{\beta}, \boldsymbol{\theta}\}$ as the set of all model parameters and set $p_{\boldsymbol{\vartheta}} = \dim(\boldsymbol{\vartheta})$. For convenience, we also transform the parameters for VI and MCMC so that $\boldsymbol{\vartheta} = \{\boldsymbol{\beta}, \log(\tau^2)\}$ for the ridge prior and $\boldsymbol{\vartheta} = \{\boldsymbol{\beta}, \log(\lambda_1^2), \log(\lambda_2^2), \dots, \log(\lambda_p^2), \log(\tau)\}$ for the horseshoe prior. We first introduce the HMC scheme in Section 4.1 before we develop the scalable VI approach in Section 4.2.

4.1. *Exact estimation using MCMC.* Since the sampler of Klein, Nott and Smith (2021) was too sticky to produce reliable estimates in this application, we suggest an alternative approach based on HMC to generate from $\boldsymbol{\vartheta} \mid \boldsymbol{x}, \boldsymbol{y}$. HMC augments $\boldsymbol{\vartheta}$ by momentum variables and draws samples from an extended target distribution that is proportional to the exponential of the Hamiltonian function. The dynamics specify how the Hamiltonian function evolves, and its volume-conserving property results in high acceptance rates of the proposed iterates when tuned properly. To do so, we employ the leapfrog integrator which involves the posterior $\log p(\boldsymbol{\vartheta} \mid \boldsymbol{x}, \boldsymbol{y})$ and its gradient; see Web Appendix A.1 for full details on HMC settings and the algorithm.

4.2. *Approximate estimation using VI.* The general idea of VI is to turn sampling from the posterior into an optimization problem. In VI the posterior $p(\boldsymbol{\vartheta} \mid \boldsymbol{x}, \boldsymbol{y})$ is approximated by a member of some tractable density family $q_{\boldsymbol{\lambda}}(\boldsymbol{\vartheta})$ that depends on a vector of variational parameters $\boldsymbol{\lambda}$. Proximity between $p(\boldsymbol{\vartheta} \mid \boldsymbol{x}, \boldsymbol{y})$ and $q_{\boldsymbol{\lambda}}$ is measured by some measure of closeness. When this measure is the Kullback–Leibler divergence (KLD) from $q_{\boldsymbol{\lambda}}$ to $p(\boldsymbol{\vartheta} \mid \boldsymbol{x}, \boldsymbol{y})$

$$\mathrm{KLD}\big(q_{\boldsymbol{\lambda}}(\boldsymbol{\vartheta}) \parallel p(\boldsymbol{\vartheta} \mid \boldsymbol{x}, \boldsymbol{y})\big) = \int \log\left(\frac{q_{\boldsymbol{\lambda}}(\boldsymbol{\vartheta})}{p(\boldsymbol{\vartheta} \mid \boldsymbol{x}, \boldsymbol{y})}\right) q_{\boldsymbol{\lambda}}(\boldsymbol{\vartheta})\, d\boldsymbol{\vartheta},$$

the optimal approximation maximizes the evidence lower bound (ELBO; Ormerod and Wand (2010)), given by

$$\mathcal{L}(\boldsymbol{\lambda}) = \int q_{\boldsymbol{\lambda}}(\boldsymbol{\vartheta}) \log\left(\frac{p(\boldsymbol{y}, \boldsymbol{\vartheta} | \boldsymbol{x})}{q_{\boldsymbol{\lambda}}(\boldsymbol{\vartheta})}\right) d\boldsymbol{\vartheta}, \tag{8}$$

with respect to $\boldsymbol{\lambda}$. Setting $h(\boldsymbol{\vartheta}) = p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\vartheta}) p(\boldsymbol{\vartheta})$, we note that (8) is an expectation with respect to $q_{\boldsymbol{\lambda}}(\boldsymbol{\vartheta})$, that is,

$$\mathcal{L}(\boldsymbol{\lambda}) = \mathbb{E}_{q_{\boldsymbol{\lambda}}}\big[\log h(\boldsymbol{\vartheta}) - \log q_{\boldsymbol{\lambda}}(\boldsymbol{\vartheta})\big]. \tag{9}$$

This observation enables an unbiased MC estimation of the gradient of $\mathcal{L}(\boldsymbol{\lambda})$ after differentiating under the integral sign. Doing so, the resulting expression for the gradient $\nabla_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\lambda})$ leads to an expectation with respect to $q_{\boldsymbol{\lambda}}$,

$$\nabla_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\lambda}) = \mathbb{E}_{q_{\boldsymbol{\lambda}}}\big[\nabla_{\boldsymbol{\lambda}} \log q_{\boldsymbol{\lambda}}(\boldsymbol{\vartheta}) \big(\log h(\boldsymbol{\vartheta}) - \log q_{\boldsymbol{\lambda}}(\boldsymbol{\vartheta})\big)\big], \tag{10}$$

where the so-called log-derivative trick $\mathbb{E}_{q_{\boldsymbol{\lambda}}}[\nabla_{\boldsymbol{\lambda}} \log q_{\boldsymbol{\lambda}}(\boldsymbol{\vartheta})] = 0$ has been used. This expression is often used in stochastic gradient ascent (SGA) to optimize the ELBO (also known as stochastic VI; see, e.g., Hoffman et al. (2013), Nott et al. (2012), Titsias and Lázaro-Gredilla (2014)). Denoting with $\widehat{\nabla_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\lambda})}$ an unbiased MC estimate of the gradient $\nabla_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\lambda})$ and with $\boldsymbol{\lambda}^{(0)}$ an initial value for $\boldsymbol{\lambda}$, SGA performs the update

$$\boldsymbol{\lambda}^{(t+1)} = \boldsymbol{\lambda}^{(t)} + \boldsymbol{\rho}^{(t)} \circ \widehat{\nabla_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\lambda}^{(t)})} \tag{11}$$

recursively. In (11), $\circ$ denotes the Hadamard (element-by-element) product of two random vectors and $\{\boldsymbol{\rho}^{(t)}\}_{t \geq 0}$ is a sequence of vector-valued learning rates with dimension $\dim(\boldsymbol{\lambda})$. In practice, it is important to consider adaptive learning rates, and we do so using the ADADELTA method of Zeiler (2012) as successfully adopted in, for example, Ong, Nott and Smith (2018).

Algorithm 2: Variational approximation with a factored covariance structure

```
   Input  : Variational parameters $\boldsymbol{\lambda}^{(0)} = (\boldsymbol{\mu}^{(0)}, \text{vech}(\boldsymbol{\Upsilon})^{(0)}, \boldsymbol{d}^{(0)})$, iteration $t = 0$
   Output: Optimal variational parameters $\hat{\boldsymbol{\lambda}}$
1  while ELBO has not converged do
2      Generate for $m = 1, \dots, M$ samples $\boldsymbol{\xi}^{(t,m)}, \boldsymbol{\delta}^{(t,m)} \sim N(\mathbf{0}, \boldsymbol{I})$.
3      Compute the unbiased estimate of the gradient
       $\widehat{\nabla_{\boldsymbol{\lambda}}\mathcal{L}(\boldsymbol{\lambda}^{(t)})} \leftarrow (\widehat{\nabla_{\boldsymbol{\mu}}\mathcal{L}(\boldsymbol{\lambda}^{(t)})}, \widehat{\nabla_{\text{vech}(\boldsymbol{\Upsilon})}\mathcal{L}(\boldsymbol{\lambda}^{(t)})}, \widehat{\nabla_{\boldsymbol{d}}\mathcal{L}(\boldsymbol{\lambda}^{(t)})})$ using the $M$ samples
       $(\boldsymbol{\xi}^{(t,1)}, \boldsymbol{\delta}^{(t,1)}), \dots, (\boldsymbol{\xi}^{(t,M)}, \boldsymbol{\delta}^{(t,M)})$.
4      Update the learning rate $\boldsymbol{\rho}^{(t)}$ using ADADELTA.
5      Update the variational parameters via $\boldsymbol{\lambda}^{(t+1)} \leftarrow \boldsymbol{\lambda}^{(t)} + \boldsymbol{\rho}^{(t)} \circ \widehat{\nabla_{\boldsymbol{\lambda}}\mathcal{L}(\boldsymbol{\lambda}^{(t)})}$ and
       $t \leftarrow t + 1$.
6  end
```

4.2.1. *Choice for the variational approximation $q_{\boldsymbol{\lambda}}$.* Successful application of stochastic VI requires a numerically tractable yet flexible variational density $q_{\boldsymbol{\lambda}}$. A popular choice for the approximation family is the Gaussian distribution, which contains the mean vector and the covariance elements as variational parameters. It has been shown to approximate posteriors well with similar copula regression models (Smith and Klein (2021)). To reduce computational cost while remaining flexible enough, we follow Ong, Nott and Smith (2018), who use a factored representation of the covariance matrix. A typical member $q_{\boldsymbol{\lambda}}(\boldsymbol{\vartheta})$ of this approximating family takes on the form

$$q_{\boldsymbol{\lambda}}(\boldsymbol{\vartheta}) = N\big(\boldsymbol{\mu}, \boldsymbol{\Upsilon}\boldsymbol{\Upsilon}^T + \boldsymbol{\Delta}^2\big),$$

with $\boldsymbol{\Upsilon}$ being a lower triangular, full rank $p_{\boldsymbol{\vartheta}} \times k$ matrix with $k \ll p_{\boldsymbol{\vartheta}}$, and $\boldsymbol{\Delta}$ a real-valued diagonal matrix with diagonal elements $\boldsymbol{d} = (d_1, \dots, d_{p_{\boldsymbol{\vartheta}}})$. The elements above the diagonal in $\boldsymbol{\Upsilon}$ are fixed to zero. Computing the elements of a $p_{\boldsymbol{\vartheta}} \times k$ matrix is much faster than computing the elements of the $p_{\boldsymbol{\vartheta}} \times p_{\boldsymbol{\vartheta}}$ covariance matrix directly.

For this variational density, Ong, Nott and Smith (2018) employ the re-parameterization trick (Kingma and Welling (2014), Rezende, Mohamed and Wierstra (2014)) to reduce variance of the MC estimates required for estimating the gradients in an unbiased fashion. In our case, this leads to re-writing the model parameters as $\boldsymbol{\vartheta} = \boldsymbol{\mu} + \boldsymbol{\Upsilon}\boldsymbol{\xi} + \boldsymbol{d} \circ \boldsymbol{\delta} := g(\boldsymbol{\xi}, \boldsymbol{\delta})$, where $\boldsymbol{\xi} \in \mathbb{R}^k$, $\boldsymbol{\delta} \in \mathbb{R}^{p_{\boldsymbol{\vartheta}}}$ and $\boldsymbol{\xi}, \boldsymbol{\delta} \sim N(\mathbf{0}, \boldsymbol{I})$, $\boldsymbol{\lambda} = (\boldsymbol{\mu}^\top, \text{vech}(\boldsymbol{\Upsilon})^\top, \boldsymbol{d}^\top)^\top$ and vech$(\boldsymbol{\Upsilon})$, stacks the elements of the lower triangle of $\boldsymbol{\Upsilon}$ (including the diagonal) columnwise into a vector. We have found $M = 50$ iterates of $(\boldsymbol{\xi}, \boldsymbol{\delta})$ to be sufficient to provide unbiased gradient estimates. To implement the SGA (11), we need the gradients

$$\nabla_{\boldsymbol{\lambda}}\mathcal{L}(\boldsymbol{\lambda}) = \big(\nabla_{\boldsymbol{\mu}}\mathcal{L}(\boldsymbol{\lambda})^\top, \nabla_{\text{vech}(\boldsymbol{\Upsilon})}\mathcal{L}(\boldsymbol{\lambda})^\top, \nabla_{\boldsymbol{d}}\mathcal{L}(\boldsymbol{\lambda})^\top\big)^\top$$

which involve the gradients of $h(\boldsymbol{\vartheta})$ with respect to the elements of $\boldsymbol{\vartheta}$. The gradients can be obtained analytically; see Supplementary Material A.2 (Hoffmann and Klein (2023)). The complete VI algorithm is summarized in Algorithm 2.

5. Analysis in autonomous driving. The basis of the IC-NLM is a deep neural network that maps images of the street ahead to the transformed steering angles (see Step 2 of Algorithm 1). We will first introduce the end-to-end learning architecture and details on the training process in Section 5.1. Afterward, our analysis in autonomous driving consists of three parts. First, in Section 5.2 we verify that VI is an accurate and scalable alternative to MCMC in the IC-NLM based on the two data scenarios introduced in Section 2.3. Second,

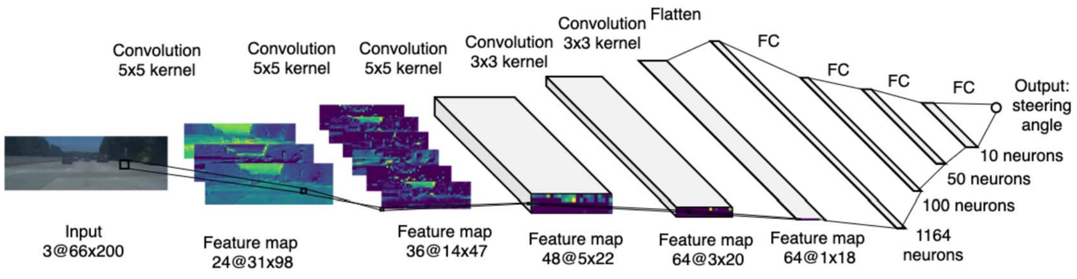


FIG. 4. *PilotNet architecture by Bojarski et al. (2016) with activations from the comma2k19 driving data.*

in Section 5.3 we benchmark the IC-NLM with two other competitive methods to obtain predictive densities for end-to-end learners with regard to calibration, coverage rates of the prediction intervals, and the ability to identify high-risk predictions. Third, we explore how predictive densities for the steering angle can help to enhance the explainability of end-to-end learners in Section 5.4.

5.1. *End-to-end learner architecture and training.* As an end-to-end learner, we use a CNN based on the pioneering PilotNet architecture by Bojarski et al. (2016) with additional regularization to prevent overfitting. We deliberately decide against exploring more complex network architectures (such as LSTMs) to ensure comparability with the existing literature. Figure 4 depicts the PilotNet architecture. PilotNet is a CNN with five convolutional layers followed by four fully connected layers. CNNs are particularly suitable to deal with image data, since they possess a "weight-sharing" property. This allows CNNs to learn features independently of their position in an image and reduces the number of training parameters. Each convolutional layer is followed by batch normalization, and we use dropout on the fully connected layers to avoid overfitting. We use ReLU (Rectified Linear Unit; Nair and Hinton (2010)) activation functions for all layers but the last layer, which has a linear activation function and dimension $p_{\vartheta} = 10$. For Step 2 of Algorithm 1, the networks are trained on the transformed steering angles $z_i = \Phi^{-1}(\hat{F}_Y(y_i))$ to minimize the MSE with the Adam optimizer. Details on training settings and the preprocessing steps to obtain a good model can be found in Supplementary Material B.1 (Hoffmann and Klein (2023)).

To estimate the marginal density of the steering angle in Step 1 of Algorithm 1, we use a nonparametric KDE with a Gaussian kernel (Racine (2008)). We found that this estimator captures the shape of the density sufficiently well, while producing no numerical issues on the edges of the data distributions where observations are sparse. Driving on highways involves relatively little steering action. Most steering angles, therefore, lie within the interval of $[-15°, +15°]$, and we limit the steering angles to $[-50°, +50°]$. Angles outside this interval are associated with complex steering actions, such as U-turns or parking. These actions are beyond the scope of present-day end-to-end learners. We fit the IC-NLM to the end-to-end learner trained on both datasets once, using both the ridge and horseshoe prior, respectively.

5.2. *Estimation accuracy of VI vs. HMC.* To assess the pure estimation accuracy of VI across the two data scenarios and priors, we compare the VI results with those of HMC. For the horseshoe prior, this is not entirely possible, since the HMC chain already partly suffers from convergence issues. This again highlights the importance of using VI as an alternative in large n scenarios.

Figure 5 depicts the posterior estimates for the model parameters using a ridge prior. Panels (a) and (b) portray the lower bound from VI with $k = 3$ factors and trace plots from HMC, respectively. Posterior means and standard deviations of model parameters can be found in

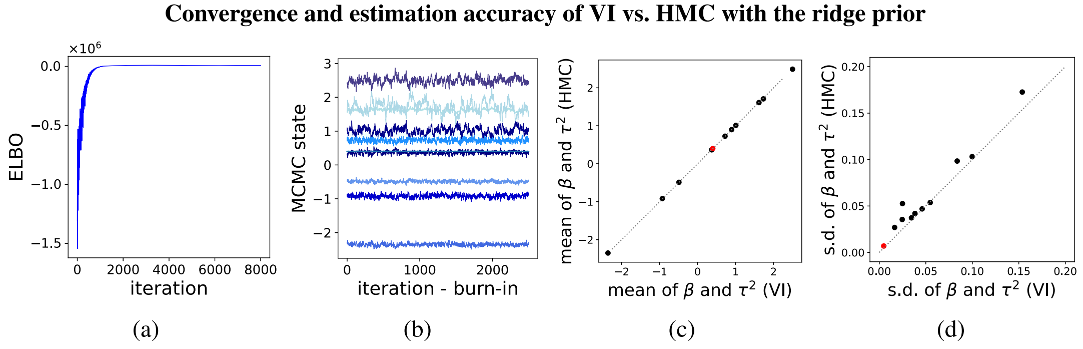


FIG. 5. *Panel* (*a*) *depicts the ELBO for VI with* $k = 3$ *factors. Panel* (*b*) *portrays trace plots of HMC, where the x-axis denotes the number of iterations minus the number of burn-in samples. Panels* (*c*) *and* (*d*) *depict posterior means and standard deviations of* $\boldsymbol{\beta}$ (*black*) *and* $\boldsymbol{\theta} = \{\tau^2\}$ (*red*) *for VI and HMC. The number of factors* $k = 3$ *was found to be sufficient to achieve fast runtimes while maintaining the same estimation accuracy as with a larger number of factors.*

panels (c) and (d), respectively. In these two right-hand plots, points on the diagonal indicate that the respective posterior means or standard deviations using VI and HMC coincide.

Figure 6 presents the same results but for the horseshoe prior. With both priors, VI converges fast, while HMC chains exhibit obvious autocorrelation when the horseshoe prior is used. For the ridge prior, VI estimates the posterior means very accurately, while the standard deviations suffer from slight over and underestimation only in a very few cases. When using a horseshoe prior, differences in standard deviations are more pronounced, but the posterior means using VI and HMC still coincide. This over- and underestimation did not translate to deviations between the final predictive densities for the steering angles obtained by VI and HMC, as can be seen when inspecting the predictive accuracy in the following section. That is why we adopted no further measures to improve the estimation of the standard deviations.

The results for the large data scenario are presented in Supplementary Material C (Hoffmann and Klein (2023)) and are similar to those presented here apart from the expected performance loss in terms of point metrics owing to the aforementioned distribution shift between the training and validation data. Table 1 reports the runtimes for the HMC samplers and VI schemes in all data scenarios as well as for the ridge and horseshoe priors.

VI is much faster than HMC, especially as n grows larger. This becomes more evident in the computationally more intensive case with the horseshoe prior, where VI is about 18 (small scenario) and 11 (large scenario) times faster.

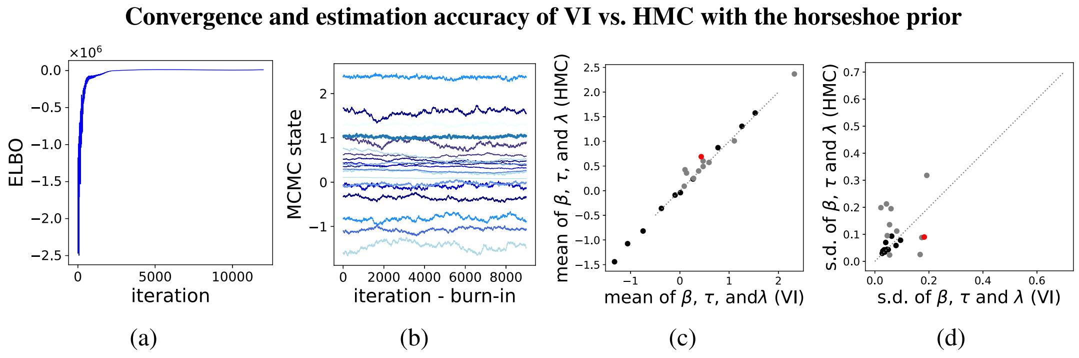


FIG. 6. *Panel* (*a*) *portrays the ELBO for VI with* $k = 3$ *factors. Panel* (*b*) *depicts trace plots of HMC. Panels* (*c*) *and* (*d*) *portray posterior means and standard deviations of* $\boldsymbol{\beta}$ (*black*), $\boldsymbol{\lambda}$ (*grey*) *and* τ (*red*).

TABLE 1

Computing times for the HMC samplers and VI. Further performance metrics for the $n = 355{,}543$ scenario can be found in Supplementary Material C (Hoffmann and Klein (2023))

		Runtime	
Shrinkage prior	Estimation type	$n = 43{,}736$	$n = 355{,}543$
ridge	HMC	4 h 14 min	15 h 5 min
ridge	VI	2 h 21 min	10 h 50 min
horseshoe	HMC	3 d 18 h 24 min	11 d 8 h 28 min
horseshoe	VI	4 h 7 min	1 d 2 h 16 min

In the most extreme case (horseshoe prior, $n = 355{,}543$), using VI reduces the computation time by several days when compared to HMC (both using two 12-core CPUs).

5.3. *Benchmark study.* Having ensured the scalability and accuracy of the VI approach, we will now benchmark the predictive accuracy, calibration, and uncertainty estimates obtained by the IC-NLM. We compare the performance on each of the two validation sets and both priors with two nonprobabilistic and two probabilistic end-to-end learners. Overall, we compare the following models:

- *Naive learner*: Driving model that always drives straight, regardless of the input.
- *Uncalibrated learner*: PilotNet trained directly on the response $\boldsymbol{y}$.
- *IC-NLM with ridge/horseshoe prior*: IC-NLM based on PilotNet trained to predict the transformed responses $z_i = \Phi_1^{-1}(\hat{F}_Y(y_i))$, as described in Algorithm 1. We fit two versions, one using VI and the other using HMC.
- *MC dropout*: PilotNet using MC dropout (Gal and Ghahramani (2016)) with dropout probabilities from Michelmore, Kwiatkowska and Gal (2018) and using 1000 dropout masks at prediction time.
- *Mixture density network (MDN)*: Mixture density network (Bishop (1994)) with 50 mixture components. An MDN is a DNN where the output is a Gaussian mixture, and the model is trained to predict the means, variances, and weights of the mixture components by maximizing the likelihood of the response.

All models are implemented using TensorFlow (Abadi et al. (2015)) and Keras (Chollet et al. (2015)). We implement MC dropout ourselves, and for the MDN we use the Keras-compatible keras-mdn-layer (Martin and Duhaime (2020)). Note that the first two models do not produce predictive densities, so they will only be used in the comparison of the predictive point accuracy, which we will measure via the mean absolute error (MAE) and the corresponding mean squared error (MSE) on the validation set. In order to compare predictive densities for the IC-NLM, MC dropout, and MDN, we follow the literature and investigate the deviation between the true and mean predictive angle. Specifically, we refer to "$E < x^\circ$ (%)" as the percentage of predictions for which the true and predicted steering angles differ by fewer than x degrees.

Results for these measures are presented first, before we investigate the calibration and uncertainty estimates in more detail. For the reasons explained in Section 2, results in the rest of this subsection will be based on the small data scenario, while respective results for the large scenario are documented in detail in Supplementary Material C (Hoffmann and Klein (2023)).

Predictive accuracy. Table 2 lists the MAE, MSE, and the percentage of predictions where the absolute deviation between the predicted and the observed steering angle is less than 5°, 2°, 1°, and 0.5°.

TABLE 2
Benchmark study. Predictive performance as measured by the MAE, MSE, and $E < x$ (%) with $x \in \{5^\circ, 2^\circ, 1^\circ, 0.5^\circ\}$ (from left to right) on the validation set. The first column details the employed models

Model	MAE	MSE	$E < 5^\circ$ (%)	$E < 2^\circ$ (%)	$E < 1^\circ$ (%)	$E < 0.5^\circ$ (%)
Naive learner	2.16	14.61	93.21	65.12	52.12	33.08
Deterministic DNN	1.42	6.43	98.19	79.61	52.92	29.26
IC-NLM + ridge + HMC	1.34	6.28	98.08	82.22	57.14	32.12
IC-NLM + ridge + VI	1.34	6.28	98.08	82.21	57.18	32.16
IC-NLM + horseshoe + HMC	**1.29**	5.67	98.21	**82.75**	**58.22**	**33.55**
IC-NLM + horseshoe + VI	1.29	**5.66**	98.21	82.73	58.16	33.51
MC dropout	1.41	6.35	**98.37**	80.08	53.13	29.38
MDN	1.36	6.63	98.16	81.91	54.55	30.37

The IC-NLM using a horseshoe prior performs best on all metrics but $E < 5^\circ$, where MC dropout performs best. Performance for VI is very similar to that of HMC which again underpins the good accuracy of our approximate scalable method. The choice of the prior for the IC-NLM is relevant, and the more complex horseshoe prior results in more accurate point predictions. Hence, it is worth the additional computational cost.

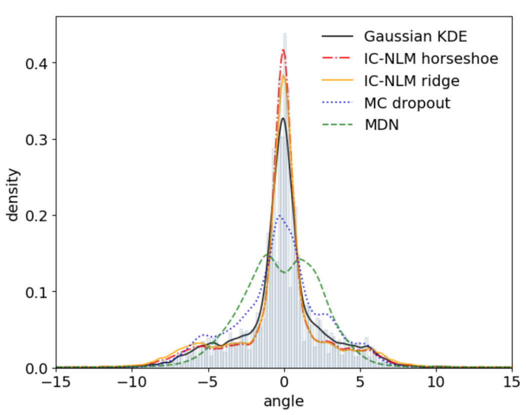
Training and prediction speed. For all models in the benchmark study, training PilotNet takes around 6.2 hours on a 12 GB NVIDIA Tesla K80 GPU ($n = 43{,}736$). For the IC-NLM, there is the additional step of fitting VI which adds two hours ($n = 43{,}736$) to the training time.

With regard to prediction, IC-NLM is more readily scalable compared to the MDN and MC dropout. From our view this is a crucial advantage, since training only has to be performed once, but prediction occurs many times during driving. MC dropout requires several forward passes through the DNN. These can be evaluated in parallel, but this is not infinitely scalable for autonomous driving applications. There is only limited space for computational resources in a vehicle, since hardware generally needs to be placed in the vehicle cabin (Lin et al. (2018)). The MDN requires sampling from several normal distributions during prediction which can be computationally intensive. The IC-NLM, on the other hand, can be evaluated in closed form without sampling, requires only a single forward pass, and evaluating the predictive density at arbitrary values is readily parallelizable.

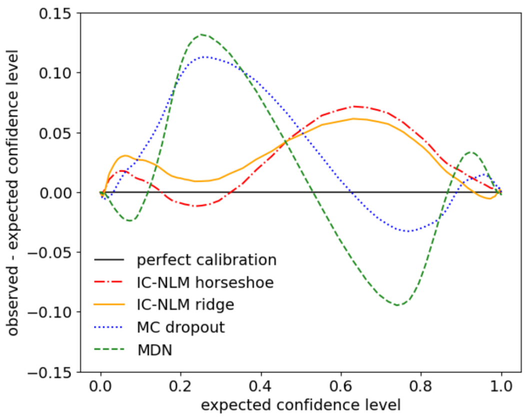
Calibration. Calibration is essential to the reliability of predictive densities, and one main motivation for the IC-NLM is to achieve marginal calibration (see Section 2.1). Figure 7(a) illustrates the marginal calibration plots of all probabilistic learners for the validation data along with the histogram and the KDE. For the IC-NLM we portray the results from VI only, since they are visually indistinguishable from the HMC results. Accurate marginal calibration occurs when the predictive densities coincide or are close to the respective KDE. Even though the IC-NLM is by construction only marginally calibrated for the training data, it demonstrates the most accurate marginal calibration also for the validation set. In contrast, MC dropout and MDN are not marginally calibrated. Figure 7(b) illustrates the probabilistic calibration of the end-to-end learners (see Section 2.1). Probabilistic calibration is measured by the difference between the observed and expected confidence level plotted over the confidence level. Even though probabilistic calibration is not guaranteed with the IC-NLM, it performs relatively well for lower expected confidence levels. In general, no method is perfectly probabilistically calibrated, as Figure 7(b) depicts.

Prediction intervals, coverage rates and high-error predictions. Predictive densities should not only be calibrated but also provide reliable and ideally sharp prediction intervals. They

Marginal and probabilistic calibration



(a) Marginal calibration

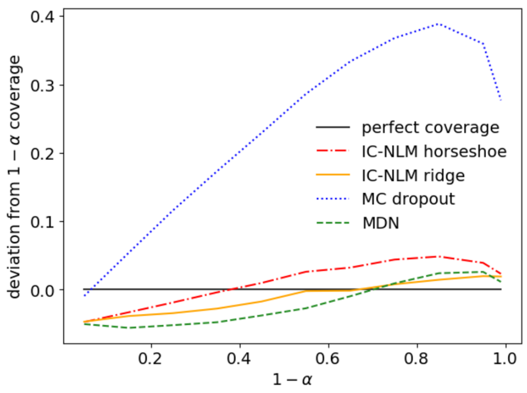


(b) Probabilistic calibration

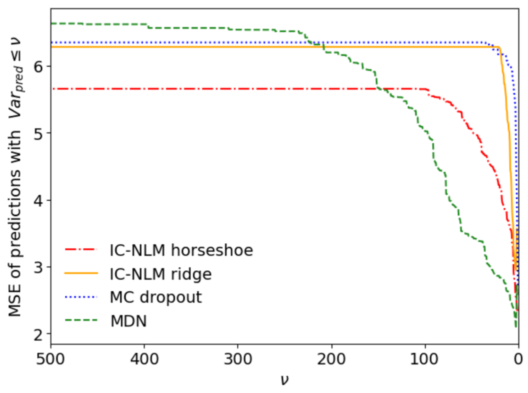
FIG. 7. *Benchmark study. Marginal (panel (a)) and probabilistic (panel (b)) calibration for the validation data with the IC-NLM with ridge (solid yellow line), IC-NLM with horseshoe (dot-dashed red line), MC dropout (dotted blue line), and MDN (dashed green line).*

should be reliable in the sense that their coverage rates are accurate and sharp, in the sense that, if accurate, the width of prediction intervals is tight. The first property can be measured through the deviation between the expected and observed coverage rates of the prediction intervals over the confidence level, as shown in Figure 8(a). The IC-NLM with a ridge prior demonstrates the most accurate prediction intervals, closely followed by the MDN and IC-NLM with a horseshoe prior. MC dropout produces unreliable prediction intervals, likely owing to the use of only 1000 dropout masks at prediction time.[2]

Coverage rates and high-error predictions



(a) Prediction interval reliability



(b) Prediction error vs. predictive variance

FIG. 8. *Benchmark study. Coverage rates and error-variance relation for the validation data with the IC-NLM with ridge (solid yellow line), IC-NLM with horseshoe (dot-dashed red line), MC dropout (dotted blue line), and MDN (dashed green line).*

[2]Using more dropout masks, however, is associated with increased computational cost and is unlikely to be an efficient solution in the context of autonomous driving.

Another usage of predictive densities is to identify overconfident learners. Optimally, high variances should point to predictions that are associated with incorrect steering actions such that a high predictive variance can act as an early warning of wrong predictions (He, Lakshminarayanan and Teh (2020)). Predictive densities can then be used to identify predictions that are potentially wrong and initiate a switch to human steering. In overconfident learners the opposite relation holds, and these learners can be dangerous when deployed in real road traffic.

Overconfident learners can be identified by plotting the MSE for all observations carrying a predictive variance under a certain threshold ν against this threshold, as done in Figure 8(b). A perfect uncertainty quantifier would produce a line from the lower-right corner to the upper-left corner so that the prediction error increases linearly as the predicted variance increases. Figure 8(b) illustrates that no end-to-end learner exhibits this perfect relation between predictive variance and predictive error. But the MDN comes quite close and is by far the most reliable uncertainty quantifier, followed by the IC-NLM with a horseshoe prior. For MC dropout and the IC-NLM with a ridge prior, hardly any relation exists between prediction error and predictive variance. This indicates that both of these learners are quite overconfident.

Overall, the IC-NLM performs best in terms of point predictions, calibration, and accuracy of the coverage rates. Only for the relation between predictive variance and predictive error does the MDN perform better. This demonstrates that the IC-NLM is a competitive option to obtain reliable predictive densities for the steering angle in autonomous driving applications.

5.4. *Understanding end-to-end learners.* End-to-end learners are generally black-box models which means that their predictions cannot easily be explained. In the context of autonomous driving, it is desirable to know how much of its surroundings an end-to-end learner actually understands so that its safety can be assessed precisely.

Predictive densities can be used to gain better insight into how an end-to-end learner sees its surroundings. A well-suited scenario to check whether an end-to-end learner understands its environment comprises situations in which multiple steering actions are valid, for example, at intersections. In this case, one can check whether the predictive density for the steering angle has several modes, where each mode should correspond to a valid steering action (compare, e.g., Xu et al. (2017)). This is also important for combining end-to-end learners with route planning.

Beyond that, we are often interested in the behavior of end-to-end learners in new environments, for example, novel lane patterns or road conditions that were not part of the training set. Figure 9 and Figure 10 depict two example images of the comma2k19 data with the predictive densities from the probabilistic end-to-end learners.

Figure 10 contains a distribution-shifted road pattern in the large, uncleaned validation set which allows the exploration of how the end-to-end learners react in situations of high uncertainty. All probabilistic learners correctly associate high uncertainty with this image and are not overconfident. Especially the MDN does not favor any particular steering angle. These examples also illustrate that it is not irrelevant which probabilistic learner is used, since, for a single input image, the predictive densities of the learners can differ notably.

Figure 9 features a parting road with two valid steering options (keep straight or turn right) in the small training set. Here, the two IC-NLMs correctly assign probability mass to the alternative steering action. MC dropout and MDN, however, both deliver almost unimodal predictions. Hence, MC dropout and MDN seem to have a limited understanding of the meaning of a parting lane and fail to indicate a second valid steering action.

Identification of multiple steering options (small scenario, training)

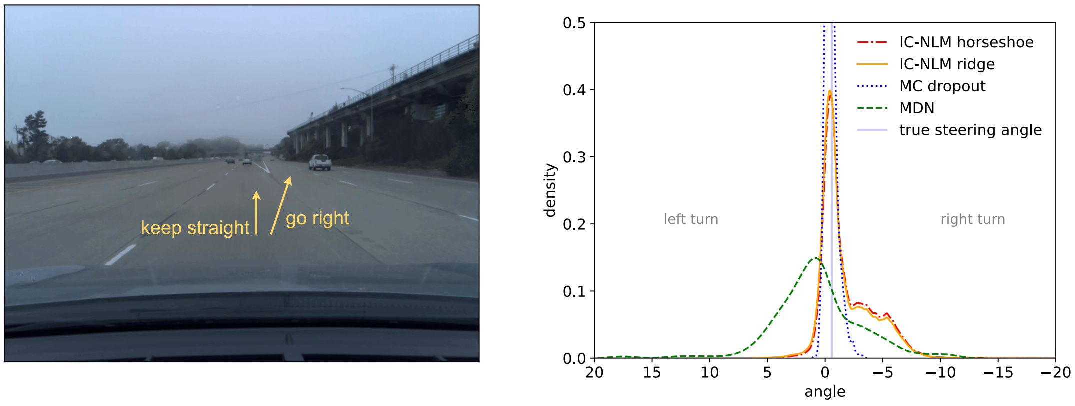


FIG. 9. *Exemplary predictive densities for a training image with multiple valid steering options. Shown are IC-NLM with ridge (solid yellow line), IC-NLM with horseshoe (dot-dashed red line), MC dropout (dotted blue line), and MDN (dashed green line), along with the true steering angle (blue vertical line).*

6. Discussion. We have expanded the IC-NLM of Klein, Nott and Smith (2021) to a scalable version using VI to enable fast uncertainty quantification for the steering angle with end-to-end learners. A detailed case study using the comma2k19 highway driving dataset reveals that the proposed VI approach is as accurate as exact inference based on MCMC and much faster in large n scenarios. The scalable IC-NLM is competitive with other uncertainty quantification methods, namely, MC dropout and MDNs. In terms of calibration, the IC-NLM is even superior. The choice of the prior in the IC-NLM has an impact on the predictive performance and reliable uncertainty quantification. Thus, we recommend using priors with more complex shrinkage schemes, even though they generally lead to longer runtimes. Priors that allow for even more flexibility could potentially improve calibration further. In fact, one disadvantage of our recommended prior, the horseshoe prior, is that the shrinkage of larger coefficients can be hard to control, thus resulting in potential undershrinking (Piironen and Vehtari (2017)). The regularized horseshoe prior (Piironen and Vehtari (2017)) solves

Uncertainty in the presence of an unknown lane pattern (large scenario, validation)

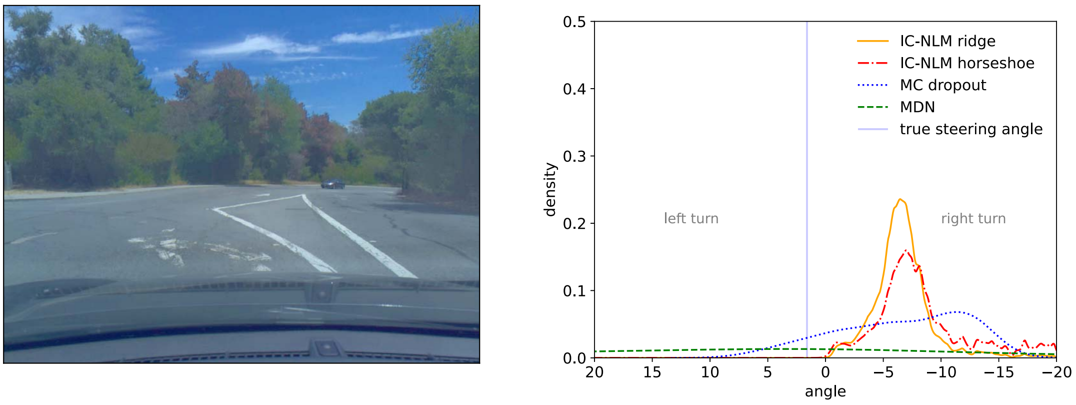


FIG. 10. *Exemplary predictive densities for a validation image with an unknown lane pattern. Shown are IC-NLM with ridge (solid yellow line), IC-NLM with horseshoe (dot-dashed red line), MC dropout (dotted blue line), and MDN (dashed green line), along with the true steering angle (blue vertical line).*

this problem by shrinking the larger coefficients like a slab and the smaller coefficients like a horseshoe prior. Future research could investigate whether the regularized horseshoe can achieve better calibration.

Future research should also address the over- and underestimation of the standard deviations in the posterior of the model parameters when using VI. A first possible source of this issue could be gradient noise which also translates to noise in the ELBO. To improve the accuracy of the standard deviations further, one could adapt methods in the spirit of Miller et al. (2017), who introduce a control variate to reduce gradient noise in VI and reach better convergence without having to increase the number of samples M in the reparameterization trick to very large numbers. A second potential source may be the assumption of a Gaussian VI. This assumption could be relaxed for instance by resorting to VIs based on Gaussian copulas (Smith, Loaiza-Maya and Nott (2020)), mixtures thereof (Gunawan, Kohn and Nott (2021)), or VI based on implicit copulas (Smith and Loaiza-Maya (2021)).

Another relevant research question is to quantify the information loss by performing Bayesian inference only in the last layer of the DNN, compared to Bayesian inference on the weights in all layers. As has been demonstrated in this work, the posterior densities of the NLM provide useful and comprehensible information on predictive uncertainty. Nonetheless, the negligence of uncertainty of all preceding layers should be investigated further, also with respect to computational feasibility in large n scenarios. A promising route to overcome any limitations resulting from this could be to build on the fully Bayesian framework of deep generalized mixed models of Tran et al. (2020).

The IC-NLM can also be extended to DNNs that take sequential data (e.g., video snippets) as an input. Three classes of model are relevant here for autonomous driving: recurrent neural networks (RNNs), LSTMs, and vision transformers (ViTs). The IC-NLM can be used without further adjustments for any of these models, as long as the output layer is a dense layer. In ViTs a multilayer perceptron (MLP) head is often used to produce an output. The IC-NLM could still be used here to quantify uncertainty of predictions by replacing the MLP head with a dense layer during fine-tuning, as proposed in Dosovitskiy et al. (2021). Extending the IC-NLM to time-series prediction requires more effort. Klein, Smith and Nott (2021) propose a more sophisticated extension for probabilistic time-series prediction using a modified IC-NLM based on RNNs in the context of electricity price forecasting.

Acknowledgements. The authors would like to thank Cornelius Hoffmann for providing storage and server infrastructure for the data and computations. The authors thank Andrew Entwistle for language proofreading.

Funding. Financial support from the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) through an Emmy Noether grant, KL 3037/1-1, is gratefully acknowledged.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J. et al. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Abbasi, S., Hajabdollahi, M., Karimi, N. and Samavi, S. (2020). Modeling teacher-student techniques in deep neural networks for knowledge distillation. In 2020 *International Conference on Machine Vision and Image Processing* (*MVIP*) 1–6. https://doi.org/10.1109/MVIP49855.2020.9116923

Amini, A., Soleimany, A., Karaman, S. and Rus, D. (2019). Spatial uncertainty sampling for end-to-end control. 1–5. arXiv:1805.04829.

Arnez, F., Espinoza, H., Radermacher, A. and Terrier, F. (2020). A Comparison of uncertainty estimation approaches in deep learning components for autonomous vehicle applications. arXiv:2006.15172.

Bishop, C. M. (1994). Mixture density networks. Unpublished.

Blundell, C., Cornebise, J., Kavukcuoglu, K. and Wierstra, D. (2015). Weight uncertainty in neural networks. In *Proceedings of the* 32*nd International Conference on Machine Learning* (F. Bach and D. Blei, eds.). *Proceedings of Machine Learning Research* **37** 1613–1622. PMLR, Lille, France.

Bojarski, M., Testa, D. D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U. et al. (2016). End to end learning for self-driving cars. 1–4,6. arXiv:1604.07316.

Carvalho, C. M., Polson, N. G. and Scott, J. G. (2010). The horseshoe estimator for sparse signals. *Biometrika* **97** 465–480. MR2650751 https://doi.org/10.1093/biomet/asq017

Chen, C., Seff, A., Kornhauser, A. and Xiao, J. (2015). DeepDriving: Learning affordance for direct perception in autonomous driving. 2722–2730.

Chi, L. and Mu, Y. (2017). Learning end-to-end autonomous steering model from spatial and temporal visual cues. In *Proceedings of the Workshop on Visual Analysis in Smart and Connected Communities*. *VSCC* '17 9–16. Association for Computing Machinery, New York. https://doi.org/10.1145/3132734.3132737

Chollet, F. et al. (2015). Keras. https://keras.io.

Craiu, V. R. and Sabeti, A. (2012). In mixed company: Bayesian inference for bivariate conditional copula models with discrete and continuous outcomes. *J. Multivariate Anal.* **110** 106–120. MR2927512 https://doi.org/10.1016/j.jmva.2012.03.010

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G. et al. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In 9*th International Conference on Learning Representations*, *ICLR* 2021.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the* 33*rd International Conference on Machine Learning* (M. F. Balcan and K. Q. Weinberger, eds.). *Proceedings of Machine Learning Research* **48** 1050–1059. PMLR, New York.

Gneiting, T., Balabdaoui, F. and Raftery, A. E. (2007a). Probabilistic forecasts, calibration and sharpness. *J. R. Stat. Soc. Ser. B. Stat. Methodol.* **69** 243–268. MR2325275 https://doi.org/10.1111/j.1467-9868.2007.00587.x

Gneiting, T., Balabdaoui, F. and Raftery, A. E. (2007b). Probabilistic forecasts, calibration and sharpness. *J. R. Stat. Soc. Ser. B. Stat. Methodol.* **69** 243–268. MR2325275 https://doi.org/10.1111/j.1467-9868.2007.00587.x

Gneiting, T. and Ranjan, R. (2013). Combining predictive distributions. *Electron. J. Stat.* **7** 1747–1782. MR3080409 https://doi.org/10.1214/13-EJS823

Goodfellow, I., Bengio, Y. and Courville, A. (2016). *Deep Learning*. *Adaptive Computation and Machine Learning*. MIT Press, Cambridge, MA. MR3617773

Gunawan, D., Kohn, R. and Nott, D. (2021). Flexible variational bayes based on a copula of a mixture of normals.

Guo, C., Pleiss, G., Sun, Y. and Weinberger, K. Q. (2017). On calibration of modern neural networks. In *Proceedings of the* 34*th International Conference on Machine Learning* (D. Precup and Y. W. Teh, eds.). *Proceedings of Machine Learning Research* **70** 1321–1330. PMLR.

He, B., Lakshminarayanan, B. and Teh, Y. W. (2020). Bayesian deep ensembles via the neural tangent kernel. In *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan and H. Lin, eds.) **33** 1010–1022. Curran Associates, Red Hook, NY.

Hoffman, M. D., Blei, D. M., Wang, C. and Paisley, J. (2013). Stochastic variational inference. *J. Mach. Learn. Res.* **14** 1303–1347. MR3081926

Hoffmann, C. and Klein, N. (2023). Supplement to "Marginally calibrated response distributions for end-to-end learning in autonomous driving." https://doi.org/10.1214/22-AOAS1693SUPPA, https://doi.org/10.1214/22-AOAS1693SUPPB

KENDALL, A. and GAL, Y. (2017). What uncertainties do we need in Bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett, eds.) **30** 5574–5584. Curran Associates, Red Hook, NY.

KINGMA, D. P. and WELLING, M. (2014). Auto-encoding variational Bayes. In *2nd International Conference on Learning Representations* (Y. Bengio and Y. LeCun, eds.).

KLEIN, N. and KNEIB, T. (2016a). Simultaneous inference in structured additive conditional copula regression models: A unifying Bayesian approach. *Stat. Comput.* **26** 841–860. MR3515025 https://doi.org/10.1007/s11222-015-9573-6

KLEIN, N. and KNEIB, T. (2016b). Scale-dependent priors for variance parameters in structured additive distributional regression. *Bayesian Anal.* **11** 1071–1106. MR3545474 https://doi.org/10.1214/15-BA983

KLEIN, N., NOTT, D. J. and SMITH, M. S. (2021). Marginally calibrated deep distributional regression. *J. Comput. Graph. Statist.* **30** 467–483. MR4270517 https://doi.org/10.1080/10618600.2020.1807996

KLEIN, N. and SMITH, M. S. (2019). Implicit copulas from Bayesian regularized regression smoothers. *Bayesian Anal.* **14** 1143–1171. MR4044849 https://doi.org/10.1214/18-BA1138

KLEIN, N., SMITH, M. S. and NOTT, D. J. (2021). Deep distributional time series models and the probabilistic forecasting of intraday electricity prices. arXiv:2010.0184.

KULESHOV, V., FENNER, N. and ERMON, S. (2018). Accurate uncertainties for deep learning using calibrated regression. In *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.). *Proceedings of Machine Learning Research* **80** 2796–2804. PMLR.

LAKSHMINARAYANAN, B., PRITZEL, A. and BLUNDELL, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett, eds.) **30**. Curran Associates, Red Hook, NY.

LIN, S.-C., ZHANG, Y., HSU, C.-H., SKACH, M., HAQUE, M. E., TANG, L. and MARS, J. (2018). The architectural implications of autonomous driving: Constraints and acceleration. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems. ASPLOS '18* 751–766. Association for Computing Machinery, New York.

MARTIN, C. and DUHAIME, D. (2020). keras-mdn-layer. Published with MIT license.

MICHELMORE, R., KWIATKOWSKA, M. and GAL, Y. (2018). Evaluating uncertainty quantification in end-to-end autonomous driving control. 1–5. arXiv:1811.06817.

MILLER, A., FOTI, N., D'AMOUR, A. and ADAMS, R. P. (2017). Reducing reparameterization gradient variance. In *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett, eds.) **30** 3–10. Curran Associates, Red Hook, NY.

NAIR, V. and HINTON, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Coference on International Conference on Machine Learning. ICML'*10 807–814. Omnipress, Madison, WI.

NEAL, R. M. (2011). MCMC using Hamiltonian dynamics. In *Handbook of Markov Chain Monte Carlo. Chapman & Hall/CRC Handb. Mod. Stat. Methods* (S. Brooks, A. Gelman, G. Jones and X.-L. Meng, eds.) 113–162. CRC Press, Boca Raton, FL. MR2858447

NELSEN, R. B. (2006). *An Introduction to Copulas*, 2nd ed. *Springer Series in Statistics*. Springer, New York. MR2197664 https://doi.org/10.1007/s11229-005-3715-x

NOTT, D. J., TAN, S. L., VILLANI, M. and KOHN, R. (2012). Regression density estimation with variational methods and stochastic approximation. *J. Comput. Graph. Statist.* **21** 797–820. MR2970920 https://doi.org/10.1080/10618600.2012.679897

OBER, S. W. and RASMUSSEN, C. E. (2019). Benchmarking the neural linear model for regression. arXiv:1912.08416.

ONG, V. M.-H., NOTT, D. J. and SMITH, M. S. (2018). Gaussian variational approximation with a factor covariance structure. *J. Comput. Graph. Statist.* **27** 465–478. MR3863750 https://doi.org/10.1080/10618600.2017.1390472

ORMEROD, J. T. and WAND, M. P. (2010). Explaining variational approximations. *Amer. Statist.* **64** 140–153. MR2757005 https://doi.org/10.1198/tast.2010.09058

PEARCE, T., BRINTRUP, A., ZAKI, M. and NEELY, A. (2018). High-quality prediction intervals for deep learning: A distribution-free, ensembled approach. In *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.). *Proceedings of Machine Learning Research* **80** 4088. PMLR.

PIIRONEN, J. and VEHTARI, A. (2017). Sparsity information and regularization in the horseshoe and other shrinkage priors. *Electron. J. Stat.* **11** 5018–5051. MR3738204 https://doi.org/10.1214/17-EJS1337SI

PINSLER, R., GORDON, J., NALISNICK, E. and HERNÁNDEZ-LOBATO, J. M. (2019). Bayesian batch active learning as sparse subset approximation. In *Advances in Neural Information Processing Systems* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox and R. Garnett, eds.) **32** 7–8. Curran Associates, Red Hook, NY.

PITT, M., CHAN, D. and KOHN, R. (2006). Efficient Bayesian inference for Gaussian copula regression models. *Biometrika* **93** 537–554. MR2261441 https://doi.org/10.1093/biomet/93.3.537

POLSON, N. G. and SOKOLOV, V. (2017). Deep learning: A Bayesian perspective. *Bayesian Anal.* **12** 1275–1304. MR3724986 https://doi.org/10.1214/17-BA1082

RACINE, J. S. (2008). Nonparametric econometrics: A primer. *Found Trends Econom.* **3** 1–88. https://doi.org/10.1561/0800000009

REZENDE, D. J., MOHAMED, S. and WIERSTRA, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the* 31*st International Conference on Machine Learning* (E. P. Xing and T. Jebara, eds.) **32** 1278–1286.

RIQUELME, C., TUCKER, G. and SNOEK, J. (2018). Deep Bayesian bandits showdown: An empirical comparison of Bayesian deep networks for Thompson sampling. 4. arXiv:1802.09127.

RODRIGUES, F. and PEREIRA, F. C. (2020). Beyond expectation: Deep joint mean and quantile regression for spatiotemporal problems. *IEEE Trans. Neural Netw. Learn. Syst.* **31** 5377–5389. https://doi.org/10.1109/TNNLS.2020.2966745

RUMELHART, D., HINTON, G. E. and WILLIAMS, R. J. (1986). Learning representations by back-propagating errors. *Nature* **323** 533–536.

SCHAFER, H., SANTANA, E., HADEN, A. and BIASINI, R. (2018). A commute in data: The comma2k19 dataset. 1. arXiv:1812.05752.

SKLAR, M. (1959). Fonctions de répartition à *n* dimensions et leurs marges. *Publ. Inst. Stat. Univ. Paris* **8** 229–231. MR0125600

SMITH, M. S. and KLEIN, N. (2021). Bayesian inference for regression copulas. *J. Bus. Econom. Statist.* **39** 712–728. MR4272930 https://doi.org/10.1080/07350015.2020.1721295

SMITH, M. S. and LOAIZA-MAYA, R. (2021). Implicit copula variational inference.

SMITH, M. S., LOAIZA-MAYA, R. and NOTT, D. J. (2020). High-dimensional copula variational approximation through transformation. *J. Comput. Graph. Statist.* **29** 729–743. MR4191239 https://doi.org/10.1080/10618600.2020.1740097

SNOEK, J., RIPPEL, O., SWERSKY, K., KIROS, R., SATISH, N., SUNDARAM, N., PATWARY, M., PRABHAT, M. and ADAMS, R. (2015). Scalable Bayesian optimization using deep neural networks. In *Proceedings of the* 32*nd International Conference on Machine Learning* (F. Bach and D. Blei, eds.). *Proceedings of Machine Learning Research* **37** 2171–2176. PMLR, Lille, France.

SONG, P. X.-K. (2000). Multivariate dispersion models generated from Gaussian copula. *Scand. J. Stat.* **27** 305–320. MR1777506 https://doi.org/10.1111/1467-9469.00191

SONG, P. X.-K., LI, M. and YUAN, Y. (2009). Joint regression analysis of correlated data using Gaussian copulas. *Biometrics* **65** 60–68. MR2665846 https://doi.org/10.1111/j.1541-0420.2008.01058.x

TITSIAS, M. and LÁZARO-GREDILLA, M. (2014). Doubly stochastic variational Bayes for non-conjugate inference. In *Proceedings of the* 31*st International Conference on Machine Learning* (E. P. Xing and T. Jebara, eds.) **32** 1971–1979.

TRAN, M.-N., NGUYEN, N., NOTT, D. and KOHN, R. (2020). Bayesian deep net GLM and GLMM. *J. Comput. Graph. Statist.* **29** 97–113. MR4085867 https://doi.org/10.1080/10618600.2019.1637747

URIA, B., MURRAY, I. and LAROCHELLE, H. (2013). RNADE: The real-valued neural autoregressive density-estimator. In *Advances in Neural Information Processing Systems* (C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani and K. Q. Weinberger, eds.) **26** 2175–2183. Curran Associates, Red Hook, NY.

WILSON, A. G. and GHAHRAMANI, Z. (2010). Copula processes. In *Advances in Neural Information Processing Systems* (J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel and A. Culotta, eds.) **23**. Curran Associates, Red Hook, NY.

XU, H., GAO, Y., YU, F. and DARRELL, T. (2017). End-to-end learning of driving models from large-scale video datasets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*) 1,7–8.

ZEILER, M. D. (2012). ADADELTA: An adaptive learning rate method. 3–4. arXiv:1212.5701.

ZHANG, Z., DALCA, A. V. and SABUNCU, M. R. (2019). Confidence calibration for convolutional neural networks using structured dropout. arXiv:1906.09551.