# Bayesian Generative Models for Capturing Uncertainties in Driver Behavior Prediction

**Sheng Li and Arec Jamgochian**
Department of Aeronautics and Astronautics, Stanford University
{lisheng, arec}@stanford.edu

## 1 Introduction

Conventionally, deep forecasting models only provide point forecasts. In safety-critical applications such as autonomous vehicles, supply chain planning, and medical planning, understanding uncertainty is essential for decision-making. Therefore, incorporating model uncertainty becomes paramount. Scarcity of data usually impedes the training of complex parametric models that can accurately capture forecast uncertainty without overfitting. Traditional approaches for capturing model uncertainty with non-parametric models (e.g. Gaussian processes) typically do not scale well to high feature dimensions. Recent advancements in Bayesian deep learning alleviate these issues by allowing us to simultaneously capture deep model parameter uncertainty while predicting distributions over outputs.

In this project[1], we examine the effectiveness of Bayesian deep learning versus traditional autoregressive models for probabilistic, low-data, time series forecasting. Specifically, we investigate driver behavior modeling and how Bayesian autoregressive models can be used to forecast probabilistic driver behavior.

We first baseline fixed-parameter feed-forward and recurrent autoregressive neural networks for predicting future highway driver trajectory distributions given past trajectory and headway information. We then relax the fixed-parameter assumption and determine distributions over recurrent neural network parameter weights that when sampled from, yield an ensemble of networks that can be used to form a distribution over future trajectories. We apply variational Bayesian methods (namely Bayes-by-backprop and Bayes-by-backprop with posterior sharpening) that learn posterior weight distributions by minimizing the variational free energy (the latter introducing latent variable dependencies). Using Monte Carlo estimation, we observe that Bayesian deep learning significantly reduces the trajectory root-weighted-squared-error (RWSE), though all models produce qualitatively accurate trajectory predictions.

The rest of this paper is organized as follows: related work is described in Section 2, the problem statement and dataset are described in Section 3, our technical approach is described in Section 4, experiments and results are shown in Section 5, and we conclude in Section 6.

## 2 Related Work

Safe planning in an autonomous system requires having an accurate understanding of the system's environment. In an autonomous vehicle, this translates to modeling an occupancy grid of space in which a vehicle can safely travel. To enable better planning, it is necessary to accurately model the behavior of surrounding objects in order to predict their future trajectories and possible interactions with the *ego* (controlled) vehicle. In general, single expected trajectories do not capture the range of possibilities to safely anticipate, and it is therefore essential to plan distributions over possible trajectories.

[1] Code of this project is available at https://github.com/parachutel/bayes-by-backprop

An upcoming survey on human driver behavior modeling, including the intention and motion prediction of surrounding objects, is done by Brown et al. [2019]. In general, the field of generative modeling for trajectory forecasting is still evolving. Gupta et al. [2018] use Generative Adversarial Networks to forecast multi-modal, interactive, socially acceptable pedestrian trajectories. Jain et al. [2019] introduce the discrete residual flow network, a variant of ResNet motivated by autoregressive generative modeling that captures uncertainty in pedestrian behavior prediction in the occupancy grid. Morton et al. [2017] use feedforward and recurrent neural networks to build probabilistic models of driver behavior, outputting parameterized distributions over future accelerations given inputs of distance headway, relative vehicle speeds, and ego vehicle speed and acceleration. This approach is closest to our autoregressive fixed-parameter benchmarks (feed forward neural network with past states and long short-term memory as the gate for a recurrent neural network), and also motivates our use of root-weighted-squared error for evaluating the trajectory distributions.

In this project, we use the HighD dataset[2] (Krajewski et al. [2018a]), an aggregate of 110,500 labeled vehicle trajectories on German highways. Krajewski et al. [2018b] train $\beta$-VAE and InfoGAN on this dataset to generate realistic trajectories for vehicles changing lanes. Rather than focusing on generating new realistic trajectories, we focus on capturing uncertainty in future trajectories given past data. We do this by incorporating uncertainty *within* a prediction model.

A few papers have investigated incorporating model uncertainty and approximate Bayesian inference in neural networks. Gal and Ghahramani [2016] proposed using Monte Carlo dropout on weights to approximate Bayesian inference. They showed Monte Carlo dropout as an interpretation of Gaussian processes. Kendall and Gal [2017] used this weight dropout technique as a variational Bayesian approximation in feedforward neural networks for computer vision applications to model epistemic uncertainties.

Another more "generative" way of encoding model uncertainty into neural networks is assuming the weights of neural networks are sampled from distributions rather than remaining fixed. Multiple methods exist for training neural networks with this formulation. Graves [2011] proposed a variational Bayesian scheme with biased gradients for the variational parameters using the Fisher matrix. As an improvement, Blundell et al. [2015] proposed Bayesian by backpropagation (BBB), a method that learns the variational parameters of weights using unbiased gradient estimates to the free variational energy, and regularizing the weights using a Gaussian mixture prior.

Fortunato et al. [2017] efficiently formulate BBB for recurrent neural networks. They also proposed a technique called posterior sharpening to adapt the variational posterior to the current batch of data, reducing the variance and giving a better approximation. Similar to what Kingma and Welling [2013] did with the variational auto-encoder (VAE), posterior sharpening introduces latent distributions given by variational parameters upon which the model parameter distributions depend. This method yields state-of-the-art performance on language modeling and image caption generation tasks.

## 3 Problem Statement

Since safe decision-making in autonomous vehicles requires probabilistic trajectory forecasts of other objects in the scene, we wish to perform generative modeling for moving object trajectories. Formally, our models take past trajectory features and output distributions over future trajectories. Given a past observation trajectory $\boldsymbol{o}_{1:t} \in \mathbb{R}^n$ $(n \geq 2)$ which is an combination of positions $\boldsymbol{x}_{1:t} \in \mathbb{R}^2$ and other environmental observations such as distance headway $dh_{1:t} \in \mathbb{R}$ and time headway $th_{1:t} \in \mathbb{R}$, etc. With these features, the observation as time step $i$ is $\boldsymbol{o}_i = [\boldsymbol{x}_i, dh_i, th_i]$. Note that the features in observation trajectory is customizable, but using more features comes at the expense of training difficulty. In this project, we evaluate our methods on highway vehicle trajectories from the HighD dataset, described in more detail below.

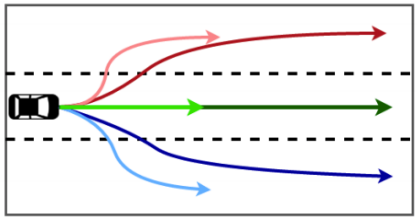
Mathematically, the model $M$ parameterized by $\theta$ should predict a distribution over future position trajectory $\hat{\boldsymbol{x}}_{t+1:T} \in \mathbb{R}^{2T}$ as described by Eq. (1).

$$p(\hat{\boldsymbol{x}}_{t+1:T}) = M(\boldsymbol{o}_{1:t}; \theta). \tag{1}$$

Fig. 1 shows illustrations of observation collection of the ego car (Fig. 1a) and the trajectory prediction task (Fig. 1b).

[2]HighD dataset is available at `https://www.highd-dataset.com`

(a) An illustration of ego car's observation $o_i$ to the environment.



(b) An illustration of possible trajectory predictions $\hat{x}_{t+1:T}$.

Figure 1: Illustrations of task of the model.

A recurrent neural network (RNN) model is a natural choice for performing this prediction task due to the discrete and consecutive structure of time series. We baseline our models using a simple feedforward network that map multiple past states to future trajectory distribution parameters, followed by a RNN that unrolls observations. We apply variational Bayes methods to train RNNs with model parameter distributions, showing how Bayesian autoregressive models improve probabilistic forecasts.

### 3.1 Dataset and Preprocessing

The HighD dataset (Krajewski et al. [2018a]) consists of 60 drone-made recordings of traffic in 6 locations on German highways. Computer vision algorithms have been used to detect extract vehicle trajectories for all 110,500 vehicle tracks at a data rate of 25 frames per second.

For our tasks, we use $x$ position, $y$ position, distance headway, and time headway features. We sample batches of data that have more proportional instances of lane changes to straight paths, or are exclusive lane changes. We remove tracks that last less than 8 seconds, and truncate tracks that last longer than 14 seconds. Similar to what was done by Krajewski et al. [2018b], we maintain a uniform input sequence length by extrapolating the $x$ positions for tracks between 8 and 14 seconds while keeping the other features fixed. We flip vehicle tracks going that go from right to left, initialize all tracks at the same position, normalize our input features, and sub-sample our data to a lower frame rate of 5 frames per second. A reduced sample of the processed data can be seen in Fig. 2.

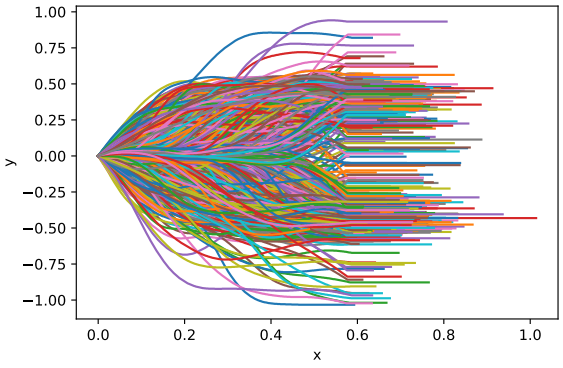



Figure 2: Processed trajectories for lane changes in a reduced dataset.

## 4 Technical Approach

To benchmark probabilistic trajectory performance, we implement simple autoregressive feedforward and recurrent neural network models that map past-observed features to means and standard deviations which parameterize a Gaussian distribution over future trajectories. We train these models by maximum liklihood estimation. More details on these benchmarks is provided in Section 5. The following subsections introduce the technical details of the Bayesian generative modeling approaches

we implement: Bayes-by-backprop (BBB, Blundell et al. [2015]) and Bayes-by-backprop with posterior sharpening (Fortunato et al. [2017]).

### 4.1 Bayes-by-backprop

In BBB, we assume the weights $\theta \in \mathbb{R}^d$ of a neural network are sampled from a learned posterior distribution, for example Gaussian $\mathcal{N}(\theta \mid \mu, \sigma^2 I)$, with mean $\mu \in \mathbb{R}^d$ and standard deviation $\sigma \in \mathbb{R}^d$. Fig. 3 gives a visual interpretation of applying BBB to an RNN.

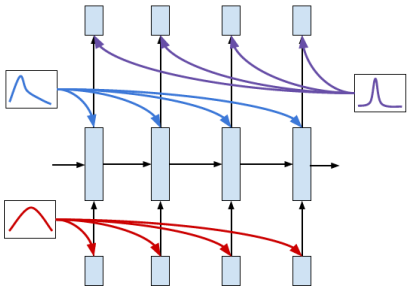


Figure 3: Applying BBB to an RNN. The model weights are sampled from a learned distribution (assuming data also have their own distribution).

In conventional maximum likelihood framework, we want to maximize the probability of our data $\mathcal{D}$ with neural network parameterized by $\theta$, i.e. maximizing $p(\mathcal{D}; \theta)$. In Bayesian maximum likelihood framework, instead, we want to maximize $p(\mathcal{D})$ with weights $\theta$ as latent variables. Equivalently we want to maximize the log-likelihood $\log p(\mathcal{D})$ where

$$
\begin{aligned}
\log p(\mathcal{D}) &= \log \int_\theta p(\mathcal{D} \mid \theta) p(\theta) d\theta \\
&= \log \int_\theta \frac{q(\theta)}{q(\theta)} p(\mathcal{D} \mid \theta) p(\theta) d\theta \\
&= \log \mathbb{E}_{q(\theta)} \left[ \frac{p(\mathcal{D} \mid \theta) p(\theta)}{q(\theta)} \right] \\
&\geq \mathbb{E}_{q(\theta)} \left[ \log \frac{p(\mathcal{D} \mid \theta) p(\theta)}{q(\theta)} \right] \\
&= \mathbb{E}_{q(\theta)} \left[ \log p(\mathcal{D} \mid \theta) \right] - D_{\mathrm{KL}}(q(\theta) \mid\mid p(\theta)).
\end{aligned}
\tag{2}
$$

The log-likelihood of data $\log p(\mathcal{D})$ is intractable to compute directly. We seek to maximize the lower bound as shown in Eq. (2). The equality only holds when $q(\theta)$ is the true posterior $p(\theta \mid \mathcal{D})$. Since the true posterior is intractable to approximate, we perform *variational inference* for variational parameters $\mu$ and $\sigma$ so that $q(\theta) = q(\theta; \mu, \sigma) = \mathcal{N}(\theta \mid \mu, \sigma^2 I)$ is as close as possible to $p(\theta \mid \mathcal{D})$. Following Eq. (2), the loss or the variational free energy is defined as

$$
\mathcal{L}(\mathcal{D}, \mu, \sigma) = -\mathbb{E}_{q(\theta;\mu,\sigma)} \left[ \log p(\mathcal{D} \mid \theta) \right] + D_{\mathrm{KL}}(q(\theta; \mu, \sigma) \mid\mid p(\theta)). \tag{3}
$$

We can apply this technique to RNNs (see Fig. 3). An RNN can be trained on a sequence by backpropagation through unrolling time steps into a feedforward network. By computing gradients $\nabla_\mu \mathcal{L}$ and $\nabla_\sigma \mathcal{L}$, we can update the variational parameters by backpropagation. And therefore, the algorithm is called Bayes-by-backprop (BBB) by Fortunato et al. [2017]. The algorithm is outlined in Algorithm 1.

**Algorithm 1** BBB

1: Sample a minibatch of data $\mathcal{D}_i$
2: Sample $\theta \sim q(\theta) = \mathcal{N}(\theta \mid \mu, \sigma^2)$
3: Compute the gradients of loss in Eq. (3) with respect to $\mu$ and $\sigma$
4: Update $\mu$ and $\sigma$

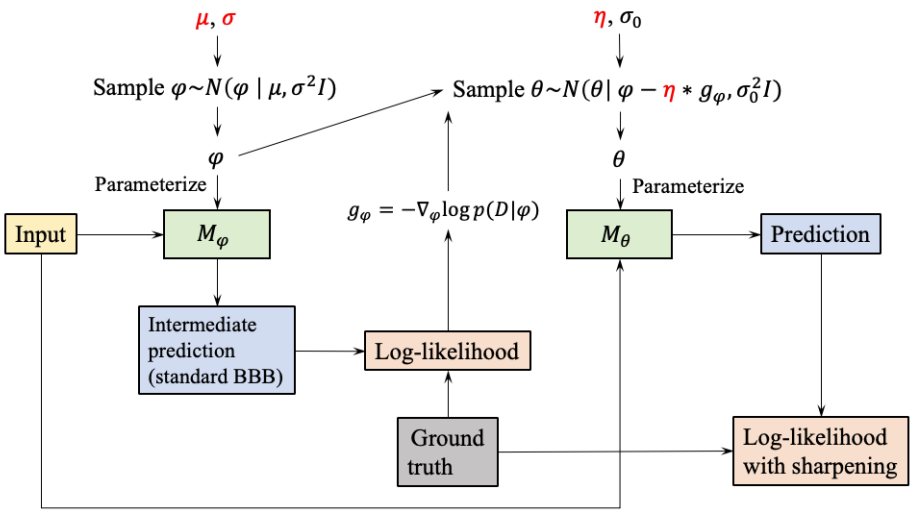



Figure 4: Prediction (inference) in training stage using BBB with posterior sharpening. Parameters to be learned are shown in red. The hierarchical parameterization scheme for model weights $\theta$ is illustrated.

### 4.2 Posterior Sharpening

As proposed by Fortunato et al. [2017], posterior sharpening can used to reduce variance of the learning process by adding side information. The idea of posterior sharpening is similar to what is done for VAE by Kingma and Welling [2013]. In VAE, a variational posterior $q(z \mid \mathcal{D})$ is used to improve the gradient estimates for $p(D)$. Similarly, in BBB, we can introduce a set of latent variables $\varphi$ upon which the model parameters $\theta$ depend, and construct a variational posterior $q(\theta \mid \mathcal{D})$ that incorporates data information. Fortunato et al. [2017] proposed constructing such a posterior in a hierarchical way:

$$q(\theta \mid \mathcal{D}) = \int q(\theta \mid \varphi, \mathcal{D}) q(\varphi) d\varphi, \tag{4}$$

with $\mu, \sigma \in \mathbb{R}^d$, and $q(\varphi) = \mathcal{N}(\varphi \mid \mu, \sigma^2)$ (data independent component). Then the authors construct a data dependent component

$$q(\theta \mid \varphi, \mathcal{D}) = \mathcal{N}(\theta \mid \varphi - \eta * g_\varphi, \sigma_0^2 I), \tag{5}$$

where $g_\varphi$ is $-\nabla_\varphi \log p(\mathcal{D} \mid \varphi)$ brings data information. Parameters $\eta \in \mathbb{R}^d$ are to be learned, and $\sigma_0^2$ is a scalar hyperparameter. This is essentially hierarchical parameterization for $\theta$.

During training, we sample $\theta$ via ancestral sampling (first sample $\varphi$, then sample $\theta$). The loss is therefore given by

$$\mathcal{L}(\mathcal{D}, \mu, \sigma, \eta) = -\mathbb{E}_{q(\varphi)}\left[\mathbb{E}_{q(\theta|\varphi,\mathcal{D})}[\log p(\mathcal{D} \mid \theta)]\right] + D_{\mathrm{KL}}(q(\theta \mid \varphi, \mathcal{D}) \mid\mid p(\theta \mid \varphi)) + D_{\mathrm{KL}}(q(\varphi) \mid\mid p(\varphi)). \tag{6}$$

BBB with posterior sharpening is outlined in Algorithm 2.

**Algorithm 2** BBB with posterior sharpening

1: Sample a minibatch of data $\mathcal{D}_i$
2: Sample $\varphi \sim q(\varphi) = \mathcal{N}(\varphi \mid \mu, \sigma^2)$
3: Let $g_\varphi = -\nabla_\varphi \log p(\mathcal{D}_i \mid \varphi)$
4: Sample $\theta \sim q(\theta \mid \varphi, \mathcal{D}_i) = \mathcal{N}(\theta \mid \varphi - \eta * g_\varphi, \sigma_0^2 I)$ //ancestral sampling
5: Compute the gradients of loss Eq. (6) with respect to $\mu$, $\sigma$, and $\eta$
6: Update $\mu$, $\sigma$, and $\eta$

Fig. 4 illustrates the training stage using BBB with posterior sharpening. Note that if we do not apply posterior sharpening (either in the testing stage or with standard BBB), then the intermediate prediction is the final outputted prediction.

## 5 Experiments

We first build a Bayesian recurrent neural network model using long short-term memory (LSTM), and run experiments with standard Bayes-by-backprop on the highD dataset (LSTM-BBB). The LSTM model $M$ has one hidden layer with a feature size of 100, followed by a single layer decoder. The input observation has four features at each frame: positions $(x, y)$, distance headway $dh$ and time headway $th$, i.e. $\boldsymbol{o} = [x, y, dh, th]$. The prediction has two features which indicate the position $(x, y)$. The decoder maps the hidden state of LSTM to a Gaussian distribution parameterized by prediction mean and variance, i.e. the model outputs $\hat{\boldsymbol{x}} = [x, y, v]$. In our structure, we assume a shared variance $v$ for $x$ and $y$ coordinates at each time step. We input 50 steps of past trajectories and predict 20 future steps.

We further run experiments for Bayesian LSTM using BBB with posterior sharpening (LSTM-BBB-Sharpen) with the same network architecture as LSTM-BBB. The hyperparameter $\sigma_0$ for the hierarchical parameterization is set to 0.02.

The prior for model parameters $\theta$ is selected to be a Gaussian mixture with two zero mean Gaussians (Blundell et al. [2015], Fortunato et al. [2017]) as given in Eq. (7).

$$p(\theta) = \prod_j \left(\pi \mathcal{N}(\theta_j \mid 0, \sigma_1^2) + (1-\pi)\mathcal{N}(\theta_j \mid 0, \sigma_2^2)\right), \tag{7}$$

where $\theta_j$ is the $j$-th weight of the model. We set $\pi = 0.25$, $\log \sigma_1 = -1$ and $\log \sigma_2 = -6$ after hyperparameter searching. This mixture of a heavy-tailed prior with a sharply peaked one resembles a spike-and-slab prior and purportedly is more amenable to use during optimization (Blundell et al. [2015]).

The baseline methods we compare against are

1. Feed forward (FF) network with two layers of 100 neurons and ELU activation. It takes in flattened past trajectory and outputs flattened predictions (means and variances);
2. Standard LSTM with one hidden layer of feature size of 100. It takes in past trajectory and outputs means and variances at prediction steps.

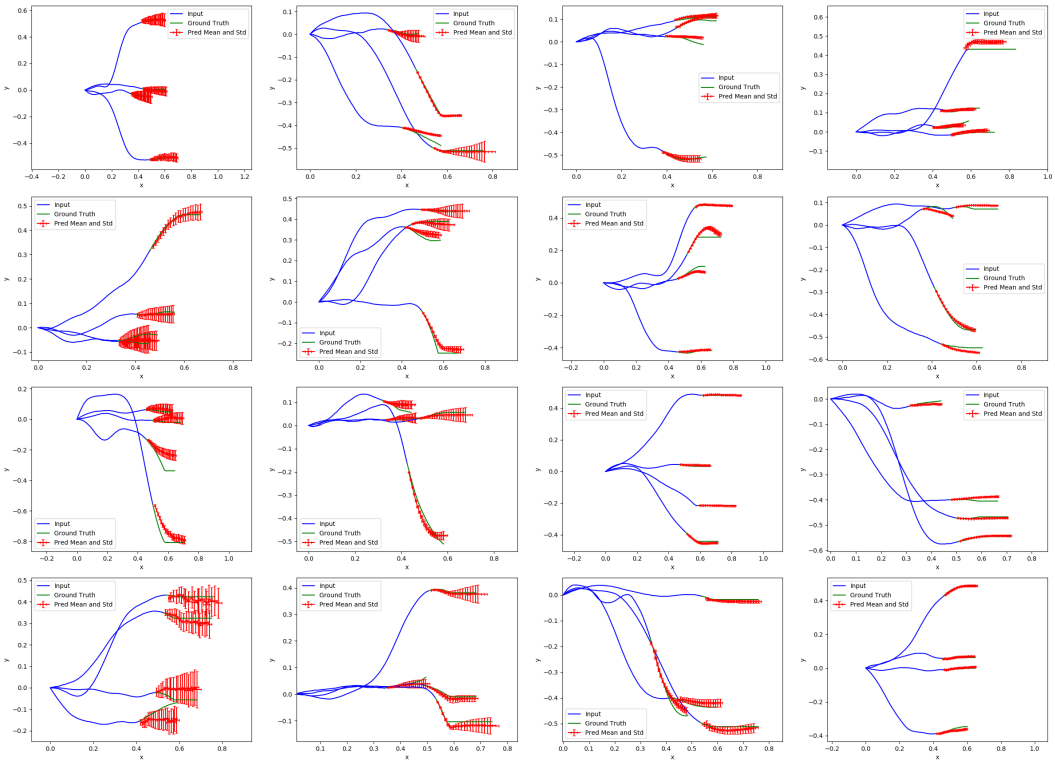
### 5.1 Results and Analysis

Test set trajectory distributions generated from all learning methods are shown in in Fig. 5. Predictions for non-BBB deterministic baseline methods (FF in Fig. 5a and LSTM in Fig. 5b) are generated by performing one pass through the models, which output means means and variances that parameterize trajectory Gaussians (standard deviations are plotted). Predictive distributions for BBB methods (LSTM-BBB in Fig. 5c and LSTM-BBB-Sharpen in Fig. 5d) are generated through averaging over Monte Carlo sampling (the average of 10 samples are plotted). Since model weights are sampled from learnt posterior distributions, we perform multiple passes through the models each time with newly sampled weights to finally obtain a distribution over the prediction.

The qualitative results in Fig. 5 show that each method can predict the trends and means of future trajectories fairly well, but there is a discrepancy in the description of uncertainties between deterministic baselines and BBB methods. Baseline methods tend to show noisier predictions than BBB methods. This observation qualitatively implies BBB methods have higher confidence in the predictions.

A good prediction should be both accurate and confident. We quantitatively measure the accuracy of prediction using root mean square error (RMSE). RMSE is root square of the expected error of mean prediction. A model can obtain low RMSE if the means of its predictions are accurate. To calculate RMSE for the benchmark feedforward and recurrent networks, we simply evaluate how the predicted mean trajectories perform against the true trajectories on a test set. To calculate RMSE for the Bayesian networks, we sample 100 networks, compute their forward passes, and compute an average expected trajectory to compare to truth.

We measure the confidence in prediction by using root weighted square error (RWSE), as was done by Morton et al. [2017] to evaluate their probabilistic driver behavior models. RWSE is the root of the expected squared error, evaluated across the whole predicted probability distribution. RWSE is low when large error has low probability and small error has high probability, and penalizes for

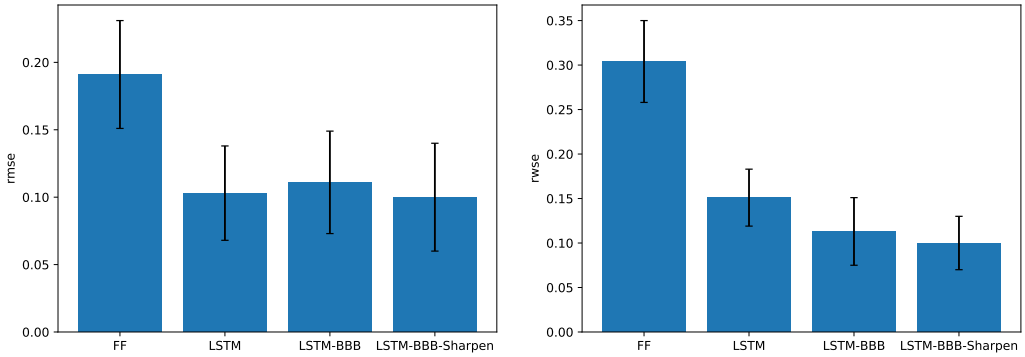



(a) FF (b) LSTM (c) LSTM-BBB (d) LSTM-BBB-Sharpen

Figure 5: Sample predictions of various methods (past trajectories are in blue, ground truths are in green, predicted means and stds are in red).

confidently predicting the wrong trajectory. For all models, we use 100 Monte Carlo samples to compute expected square error. RMSE and RWSE are formally defined in Eq. (8). Two-norm is used in computing errors since the models predict 2-D spatial trajectories.

$$\begin{aligned} \text{RMSE} &= \sqrt{\mathbb{E}_{(\boldsymbol{x},\boldsymbol{o})\sim\mathcal{D}}\left[||\boldsymbol{x} - \mathbb{E}_{\hat{\boldsymbol{x}}\sim p(\hat{\boldsymbol{x}}|\boldsymbol{o},\theta)}[\hat{\boldsymbol{x}}]||_2^2\right]}, \\ \text{RWSE} &= \sqrt{\mathbb{E}_{(\boldsymbol{x},\boldsymbol{o})\sim\mathcal{D}}\left[\mathbb{E}_{\hat{\boldsymbol{x}}\sim p(\hat{\boldsymbol{x}}|\boldsymbol{o},\theta)}\left[||\boldsymbol{x} - \hat{\boldsymbol{x}}||_2^2\right]\right]}. \end{aligned} \tag{8}$$




(a) RMSE (b) RWSE

Figure 6: Evaluation results of various methods on the test set.

The evaluation results for RMSE and RWSE are shown in Fig. 6. RNN models outperform FF model by a considerable margin in both metrics. Standard LSTM model perform about the same as LSTM-BBB and LSTM-BBB-Sharpen in terms of prediction accuracy (RMSE). However, the ensemble of models obtained with Bayesian deep learning reduces the variance of the predictive distributions, thereby reducing the RWSE. With posterior sharpening, RWSE is reduced further.

## 6 Conclusion

Making decisions in safety-critical systems requires reasoning about agents' uncertainty. For driving, it becomes essential to reason about what other actors could possibly do as opposed to what they are most likely to do. In this project, we used Bayesian deep learning to simultaneously capture model and output uncertainty by keeping distributions over deep model parameters.

We used two variational methods – Bayes-by-backprop and Bayes-by-backprop with posterior sharpening – to find posterior distributions over model weights that minimize variational free energy. We compare model performance with more standard autoregressive models, namely a feedforward and a recurrent neural network that map past states to parameters describing future state distributions. We observe that while all methods produce qualitatively good results, Bayesian methods noticeably reduce variance and RWSE of the predictions.

Future work includes embedding and including information about surrounding vehicles rather than just the leading vehicle, since driving decision are often dictated by the whole scene rather than just the leading vehicle and past trajectory. Future modeling work includes trying deeper architectures, implementing other Bayesian deep learning methods (e.g. Monte Carlo dropout by Gal and Ghahramani [2016]), and evaluating the wellness of calibration in the model forecasts and implementing recalibration techniques, such as the one proposed by Kuleshov et al. [2018].

## References


Kyle Brown, Katherine Driggs-Campbell, and Mykel Kochenderfer. A survey of modeling human driver behavior. *Under Review*, 2019.

Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2255–2264, 2018.

Ajay Jain, Sergio Casas, Renjie Liao, Yuwen Xiong, Song Feng, Sean Segal, and Raquel Urtasun. Discrete residual flow for probabilistic pedestrian behavior prediction. *arXiv preprint arXiv:1910.08041*, 2019.

Jeremy Morton, Tim A Wheeler, and Mykel J Kochenderfer. Analysis of recurrent neural networks for probabilistic modeling of driver behavior. *IEEE Transactions on Intelligent Transportation Systems*, 18(5):1289–1298, 2017.

Robert Krajewski, Julian Bock, Laurent Kloeker, and Lutz Eckstein. The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems. In *21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2118–2125. IEEE, 2018a.

Robert Krajewski, Tobias Moers, Dominik Nerger, and Lutz Eckstein. Data-driven maneuver modeling using generative adversarial networks and variational autoencoders for safety validation of highly automated vehicles. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2383–2390. IEEE, 2018b.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584, 2017.

Alex Graves. Practical variational inference for neural networks. In *Advances in neural information processing systems*, pages 2348–2356, 2011.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

Meire Fortunato, Charles Blundell, and Oriol Vinyals. Bayesian recurrent neural networks. *arXiv preprint arXiv:1704.02798*, 2017.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression. *arXiv preprint arXiv:1807.00263*, 2018.