Bayesian Deep Learning and Uncertainty in Computer Vision

by

Buu Phan

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2019

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Visual data contains rich information about the operating environment of an intelligent robotic system. Extracting this information allows intelligent systems to reason and decide their future actions. Erroneous visual information, therefore, can lead to poor decisions, causing accidents and casualties, especially in a safety-critical application such as automated driving. One way to prevent this is by measuring the level of uncertainty in the visual information interpretation, so that the system knows the reliability degree of the extracted information.

Deep neural networks are now being used in many vision tasks due to their superior accuracy compared to traditional machine learning methods. However, their estimated uncertainties have been shown to be unreliable. To mitigate this issue, researchers have developed methods and tools to apply Bayesian modeling to deep neural networks. This results in a class of models known as Bayesian neural networks, whose uncertainty estimates are more reliable and informative. In this thesis, we make the following contributions in the context of Bayesian Neural Network applied to vision tasks. In particular:

- We improve the understanding of visual uncertainty estimates from Bayesian deep models. Specifically, we study the behavior of Bayesian deep models applied to road-scene image segmentation under different factors, such as varying weather, depth, and occlusion levels.
- We show the importance of model calibration technique in the context of autonomous driving, which strengthens the reliability of the estimated uncertainty. We demonstrate its effectiveness in a simple object localization task.
- We address the high run-time cost of the current Bayesian deep learning techniques. We develop a distillation technique based on the Dirichlet distribution, which allows us to estimate the uncertainties in real-time.[1]

[1] We found several similar published works on arXiv [60, 11] at around the same time we were working on this idea.

Acknowledgements

I would like to thank my advisors Prof.Krzysztof Czarnecki and Dr.Rick Salay, for their guidance, support and feedback they have provided throughout my studies. I would also like to thank Dr.Vahdat Abdelzad, Dr.Jaeyoung Lee, Samin Khan, Sachin Vernekar and Taylor Denouden for many useful discussions and collaborations.

Dedication

This thesis is dedicated to my family and friends.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Recent advances in deep neural networks (DNNs) have created breakthroughs in many fields such as computer vision [50, 55], natural language processing [79, 16] and control [64, 54]. Unlike traditional machine learning methods, whose performance relies on the quality of engineered features, deep models are able to progressively build and extract features automatically. This property makes deep models more adaptive to various challenging tasks. For example, consider the human recognition task in computer vision- specifying and engineering the features can be challenging. By building deep models, we let the systems automatically figure out the rules and features by observing the data.

There is a price for the adaptive property, however: we do not understand what is happening inside the model. The model may give us an erroneous prediction because it is based on an incorrect rule. In a safety-critical application such as automated driving, this can cause injuries or loss of life. In fact, detection failure in perception system is the main reason behind recent reported self-driving accidents [1, 80, 37] (all involved a loss of human life). Addressing this problem turns out to be extremely challenging, researchers are currently focusing on two aspects of DNNs that can help us mitigate this problem: robustness and reliability.

Although DNNs have been reported to be surpassing human-level performance (in terms of accuracy) in vision tasks [32, 33], they are not robust compared to the human vision [47]. By robustness, we mean the ability of the model to predict accurately in different operating conditions, such as low-light, blurring, and noisy images. Despite being adaptive,

DNN's performance still depends on the data distribution and thus, changing the operating conditions will degrade the performance. This robustness property is very desirable for any perception system in autonomous driving [88], since the outdoor environment naturally possesses multiple variations that could negatively affect the visual data. This thesis, however, does not tempt to address this problem.

The second property is that the model should be reliable and able to tell how likely its prediction is going to be correct. This can be done by incorporating uncertainty estimation into the model. Recently, the field of Bayesian Deep Learning (BDL) has been actively creating tools and methods to apply the Bayesian framework to DNNs, so that the obtained uncertainty estimates are much more informative. Specifically, the resulted class of models, known as Bayesian neural networks (BNNs), has three characteristics:

- It can be used in small-data scenarios since Bayesian modelling is a well-established approach to prevent over-fitting in machine learning.
- It allows us to estimate two different types of uncertainty, namely aleatoric (data) and epistemic (model) uncertainty. Each of them corresponds to a different source of uncertainty.
- It gives us a good heuristic for data collection in order to improve model's performance.

The aleatoric uncertainty represents the intrinsic uncertainty in our data, and it cannot be reduced even if we have an infinite amount of such data [39]. Noise, lack of visual information and occlusions in the images are a few of several factors that increase the aleatoric uncertainty in our data. For example, if the object is completely occluded, then it would be impossible for the model to classify correctly all the time and the model reflects this by outputting a high aleatoric uncertainty. The epistemic uncertainty, on the other hand, represents the weakness of our model due to the insufficiency of data. Data in the high-density regions will have a lower epistemic uncertainty than those in the low-density one. Furthermore, unlike aleatoric uncertainty, epistemic uncertainty can be reduced by collecting more data. For example, if our training dataset lacks pedestrian images, then the model will not generalize well for the pedestrian class and output a high epistemic uncertainty when it encounters such images. Once we show the model more pedestrian data, the model generalizes better and reduces its epistemic uncertainty. As this thesis focuses on BNNs, the main objective is to explore and understand the behavior of these two types of uncertainty so that we can have a deeper understanding of how we should apply and expect them to behave in our perception system.

Remarks (Practical applications of uncertainty measures). As mentioned, uncertainty estimates (or measures) can allow us to avoid hazardous situations in safety-critical systems. In the case of image segmentation, for example, segmented regions with high uncertainty estimates (either aleatoric and epistemic) indicate that there is a high chance that the predictions are inaccurate and the system should try to avoid the potential risk arises from those regions. Besides this, other applications of uncertainty measures are:

- Active learning: in this task, the model interactively chooses (or queries) the data points and asks the user to label them. In many cases, labeling is a costly and laborious process, and active learning aims to reduce the amount of labeled data without sacrificing the performance. There are many querying strategies for active learning and in general, they should outperform the random strategy (choose any data point randomly). Gal, et al. [24] and Mackowiak, et al. [59] show that iteratively choosing data with high uncertainty measure saves a significant amount of labeled data while keeping the performance intact.
- Sensor and model fusion: with visual uncertainty estimates, we can combine the ocular information with other sensory or model's information to obtain more precise information. Ganti and Waslander [25] leverage BNN's uncertainty estimates to improve the feature selection process for their visual simultaneous localization and mapping (SLAM) algorithm. Fang, et al. [19] combine visual uncertainty from spatial and temporal estimation model in their video visual saliency detection algorithm. By this way, they achieved a significantly better result than other methods.
- Model improvement: in many situations, we would like to ensure that the model has received a sufficient amount of data and reached a maximum performance. Later in Chapter 3, our experiment shows that the epistemic uncertainty estimate can potentially be used to help us with this task.

1.2 Overview

Researchers have started to work on BNNs in the late 1980s [14, 82, 58, 67, 4] and developed various methods and models before the 2000s. For instance, in his thesis, N.Radford [67] showed the relationship between BNNs and Gaussian processes, and introduced a

Hamiltonian Monte-Carlo (HMC) method for Bayesian inference in BNNs. D. Mackay [58] also developed methods and derived several theoretical results for BNNs. Many of these methods, however, are computational-expensive for many practical problems.

Recent advances in BNNs [85, 23, 43] and computing resources allow us to apply Bayesian modelling to complex deep architectures for challenging vision tasks, such as image semantic segmentation and object detection. Specifically in road-scene semantic segmentation, the work by A.Kendall and Y.Gal [39] has shown the benefits of BNNs in terms of accuracy and reliability. However, the experiments in these works do not investigate the effects of uncertainty influence factors [12], which are factors influencing the perceptual uncertainty, on the uncertainty estimates. A possible reason could be the lack of datasets with real-world images with varying influence factors and the associated factor labels. Yet understanding how the uncertainty estimates behave under these factors is needed for assuring the system performance. This thesis aims to narrow this gap, studying those effects on uncertainty by using the ProcSy synthetic dataset. Besides, we also introduce and suggest several improvements in calibration and distillation techniques for BNNs. This thesis is structured as follow:

- Chapter 2 provides a necessary background. Specifically, we describe the Bayesian approach for deep models, different types of uncertainty estimates and their associated metrics.

- Chapter 3 studies how different influence factors, such as occlusion, depth, and inclement weather, affects the uncertainty estimates in the image segmentation task. We study this by using the ProcSy synthetic dataset [44]. Surprisingly, we find that the *estimated aleatoric uncertainty* from Bayesian deep models can be reduced with more training data. This chapter is adapted from B.Phan, et al. [71].

- Chapter 4 discusses the topic of model calibration, which is a desirable property for decision-making under uncertainty. We also show how to apply calibration techniques in the single object classification and localization task. This chapter is adapted from B.Phan, et al. [72].

- Chapter 5 introduces a distillation method for estimating the epistemic uncertainty in real-time. This chapter is my unpublished independent work. After having finished the experiment for this chapter, we found several similar published works on arXiv [60, 11], published at around the same time we were working on this project.

- Chapter 6 provides a conclusion and several future directions.

Chapter 2

Background

We begin this chapter with a brief overview of the parameter estimation problem. We will describe two different approaches to this problem, from the frequentist and Bayesian point of view. Then we describe methods to incorporate uncertainty estimation into deep models in both Bayesian and non-Bayesian case. We finalize the chapter with a section about the Kullback–Leibler divergence.

2.1 The Parameter Estimation Problem

The field of machine learning centers around the problem of parameter estimation. Given a training dataset, we define a model, use a learning algorithm to estimate the model's parameters so that it would generalize well for unseen data. In this section, instead of focusing on a particular machine learning problem, we show a general way to do modeling for parameter estimation problems. In Chapter 1.1, we say that Bayesian modeling is useful for small-data scenarios, which can raise a doubt to people who have been practicing classical modeling (frequentist). The main goal of this section is to show that the Bayesian approach is a better candidate for the problem of our interest.

We first give an overview of two approaches, namely Bayesian and classical (frequentist) inference, to the parameter estimation problem. Then, we give an example of why we should choose Bayesian modeling for this problem.

2.1.1 Statistical Modeling

Consider that we have an independent and identically distributed (i.i.d) dataset $D = \{x_1, x_2, x_3, ..., x_N\}$ and a generative process: $p(x|\theta)$ (given or assumed), we would like to estimate the parameter θ. We note that in this thesis, unless specified otherwise, we will assume that the dataset we are dealing with are i.i.d.

Frequentism considers θ as *a parameter* and addresses this problem by constructing an unbiased estimator $\hat{\theta}$ for θ and build also a confidence interval for this estimator. When N is large, then this estimator is very close to the true parameter. For a 95% confidence interval, it means that if we repeat this experiment many times, 95% of these confidence intervals will contain the true value. For small N, this interval tends to be large (since the variance of the estimator is high when N is low).

Bayesianism considers θ as a *random variable* and addresses this estimation problem as follow. We begin by defining a reasonable prior $p(\theta)$, and then apply the Bayes theorem and the probability marginalization rule to find the posterior $p(\theta|D)$:

$$\begin{aligned} p(\theta|D) &= \frac{p(\theta)p(D|\theta)}{p(D)} \qquad \text{(Bayes' theorem)} \\ &= \frac{p(\theta)p(D|\theta)}{\int p(\theta)p(D|\theta)d\theta} \text{ (Marginalization)} \end{aligned} \tag{2.1}$$

The credible interval is then constructed based on $p(\theta|D)$. A 95% credible interval means that there is a 95% probability that the true parameter lies within this interval. When the number of data N increases, the posterior will also converge to a true parameter. Otherwise, the credible interval will be large when N is small.

2.1.2 Why being Bayesian but not Frequentist?

So far, we have seen that the estimation from Bayesian and frequentist approaches have similar characteristics. For a small amount of data, the confidence and credible interval tend to be large whereas with large N, it converges to the true value. Also, on many occasions, their results are similar [84]. So what makes Bayesian special? It turns out that

the subtle difference in definition between confidence interval and the credible interval is one of the reasons.

We use an example shown in [84, 36] to motivate the need for Bayesian modeling. In this example, we consider a device that is guaranteed to operate without failure before the time θ. After the time θ, the device will be randomly malfunctioning according to an exponential distribution. This can be modeled by considering the following model $p(x|\theta)$ where x is the living life time of a device:

$$p(x|\theta) = \begin{cases} \exp(\theta - x), & \text{if } x > \theta \\ 0, & \text{otherwise} \end{cases} \tag{2.2}$$

Suppose now that we have several samples of failure data $D = \{10, 12, 15\}$ from these devices, we would like to estimate the parameter θ. Intuitively, from this data, we know that $\theta < 10$ based on the definition (θ is the same for all devices). However, by using the frequentist's approach, we end up having a 95% confidence interval as $[10.2, 12.2]$ for the unbiased estimator. On the other hand, the Bayesian's approach gives us the 95% credible interval as $[9.0, 10.0]$ (see [36] for more details of the solution). While the Bayesian's perspective agrees with our knowledge, the frequentist's does not, so is the frequentist's view wrong?

Actually, frequentists are not wrong. In their point of view, 95% of *such intervals* will contain the true parameter θ, while the Bayesian's result says that *this interval* is 95% likely to contain the true θ. By saying "such intervals", it means the following: suppose that for 100 days, each day we collect three samples and compute the confidence interval for each day (based on those three samples), then around 95 of these intervals will contain the true θ. This means that the frequentist bound does not give us the information that we are interested in, the observed samples do not give us much detail about the parameters. If the amount of samples increases a lot more, their estimations will undoubtedly converge to a true θ[66], but for now, this is not our concern. This example motivates the use of Bayesian modeling in parameter estimation problems. We illustrate this in Figure 2.1. The estimated Bayesian 95% credible interval contains the information where our true parameter is likely to lie in, while frequentist 95% confidence interval does not.

Remarks We note that in this section, we use the term "confidence" and "credible" interval to differentiate the concept between Frequenism and Bayesianism. The rest of this thesis will use the term "confidence interval" to describe the Bayesian interpretation.

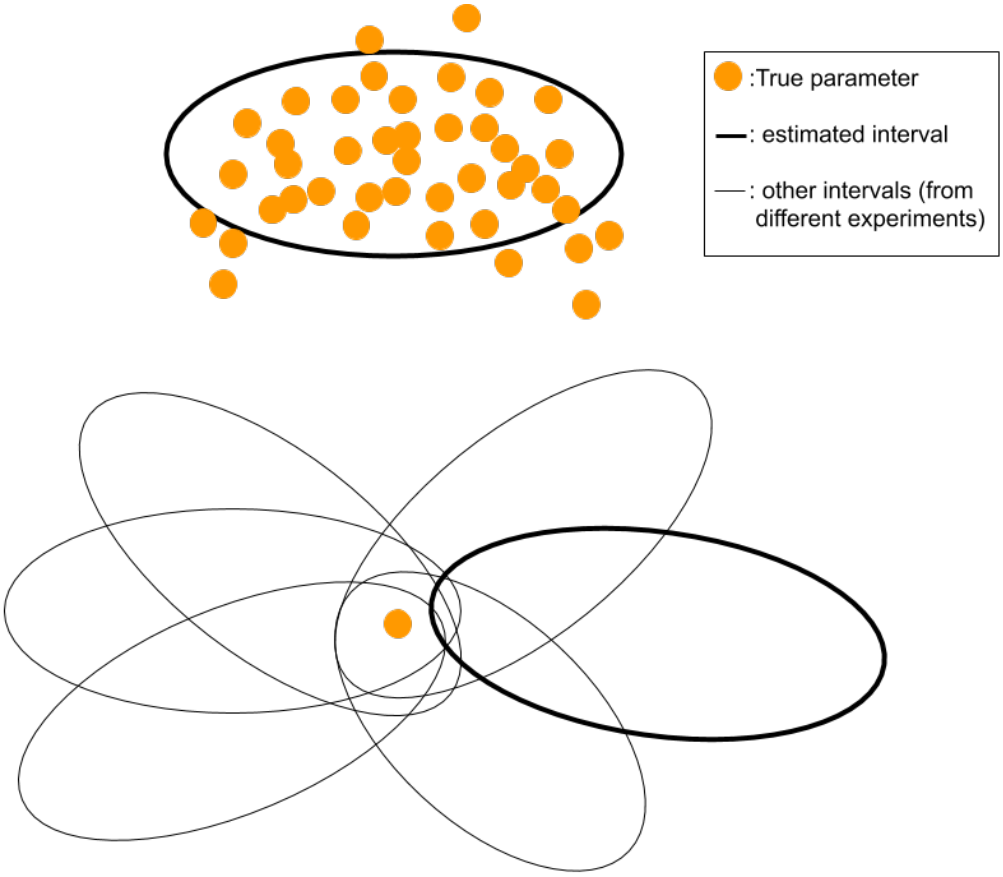

Figure 2.1: Interpretation for the example in Section.2.1.2. The top diagram shows the Bayesian credible interval and the bottom diagram shows the frequentist confidence interval.

.

2.2 Non-Bayesian Uncertainty Estimation

From now on, we will turn our focus to the supervised learning setting in DNN. In this section, we review methods for estimating the uncertainties for the traditional deep learning model. This type of uncertainty is often regarded as aleatoric uncertainty in the Bayesian context, which can be interpreted as an irreducible noise in the data. We note that none of the methods in this section is Bayesian and the examples assume that we have a lot of training data (so that there is no difference between being Bayesian and non-Bayesian).

2.2.1 Supervised-learning task

In a classification task, we are given a dataset $\mathbf{D} = \{\mathbf{X}, \mathbf{Y}\}$. We are given the input data $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ and the associated labels $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_N\}$, where each $\mathbf{y}_j$ belongs to one of the K classes $\{1, 2, ..., K\}$. For regression, the associated labels $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_N\}$ are real values.

For the non-Bayesian (or frequentist) approach, the model's weights are deterministic and we aim to find it by optimizing some objective function. The objective function function, in most of the cases, is derived based on the principle of maximum likelihood. At test time, given a new input $\hat{\mathbf{x}}$, we need our model to output an estimated value $\hat{\mathbf{y}}$. When uncertainty is considered, we wish to capture the predictive probability $p(\hat{\mathbf{y}}|\hat{\mathbf{x}})$.

2.2.2 Classification

In the classification task, we usually build a DNN with a softmax activation at the output layer and train the model by optimizing the softmax likelihood. At test time, the softmax probability can then be used as an estimate of $p(\hat{\mathbf{y}}|\hat{\mathbf{x}})$. Specifically, for each training pair $(\mathbf{x}_i, \mathbf{y}_i)$, the function $f_\omega^m(\mathbf{x}_i)$ gives a logit value for class m. The softmax likelihood for this pair is:

$$p(\mathbf{y}_i|\mathbf{x}_i) = \frac{\exp\left(f_\omega^{\mathbf{y}_i}(\mathbf{x}_i)\right)}{\sum \exp(f_\omega^m(\mathbf{x}_i)} \tag{2.3}$$

The negative log-likelihood (or cross-entropy) loss can then be defined as:

$$L(\omega) = -\sum_i \log p(\mathbf{y}_i|\mathbf{x}_i) = -\sum_i \log \frac{\exp\left(f_\omega^{\mathbf{y}_i}(\mathbf{x}_i)\right)}{\sum \exp(f_\omega^m(\mathbf{x}_i)} \tag{2.4}$$

Remarks In computer vision, data uncertainty can occur due to the lack of visual information. If we are given a completely dark image and asked to infer about the object in that image, we would have no clue. For the neural network, ideally, such images will be projected close to the decision boundary. As we will see in the next section, occlusion and depth are also sources of this aleatoric uncertainty.

2.2.3 Regression

In the regression task, we usually build and train a model to predict solely the labels given an input. However, in many cases, there is noise in the labels and its level varies differently

according to the input (as shown in the leftmost figure in Figure.2.2). So given input data, we might want to predict the amount of uncertainty in the labels. Nix, D.A. and Weigend, A.S [68] show that this can be done by assuming:

$$\mathbf{y}_i = f_\omega(\mathbf{x}_i) + \epsilon(\mathbf{x}_i), \text{ where } \epsilon(\mathbf{x}_i) \sim \mathcal{N}(0, \sigma_\omega^2(\mathbf{x}_i)) \tag{2.5}$$

This means that the labels is a sum of a function $f_\omega(\mathbf{x}_i)$ and an input-dependent noise $\epsilon(\mathbf{x}_i)$. The likelihood and log likelihood for each training data point $\{\mathbf{x}_i, \mathbf{y}_i\}$ are:

$$p(\mathbf{y}_i|\mathbf{x}_i;\omega) = \mathcal{N}(f_\omega(\mathbf{x}_i), \sigma_\omega^2(\mathbf{x}_i)) = \frac{1}{\sigma_\omega(\mathbf{x}_i)\sqrt{2\pi}} \exp\left[\frac{-(\mathbf{y}_i - f_\omega(\mathbf{x}_i))^2}{2\sigma^2(\mathbf{x}_i)}\right] \tag{2.6}$$

$$\log p(\mathbf{y_i}|\mathbf{x_i};\omega) = \frac{-(\mathbf{y_i} - f_\omega(\mathbf{x_i}))^2}{2\sigma_\omega^2(\mathbf{x_i})} - \frac{1}{2}\log \sigma_\omega^2(\mathbf{x_i}) \tag{2.7}$$

With this log likelihood function, we can formulate the loss function as follow:

$$L(\omega) = \sum_i^N \frac{(\mathbf{y_i} - f_\omega(\mathbf{x_i}))^2}{2\sigma_\omega^2(\mathbf{x_i})} + \frac{1}{2}\log \sigma_\omega^2(\mathbf{x_i}) \tag{2.8}$$

In neural network models, we add an additional output onto the final hidden layer to predict the variance (or log variance due to numerical issue). We illustrate the result in Figure 2.2.

2.3 Bayesian Neural Networks

With BNN models, instead of getting a single prediction from a learned set of weight values, we obtain the prediction by taking into account the outputs of multiple models, whose weight values are derived under the Bayesian framework. In regions that have a large amount of data, the predictions of these models are consistent with each other; on the other hand, in regions that lack data, this consistency does not tend to hold. Since deep models contain a large number of weights, applying the Bayesian framework to a deep model is computationally intractable; therefore, to obtain different sets of weight values, we need to use Bayesian approximation techniques.

We begin this section by describing the general Bayesian approach in machine learning. Then we introduce the dropout method [23] for Bayesian approximation and end with a literature review of Bayesian approximation techniques.

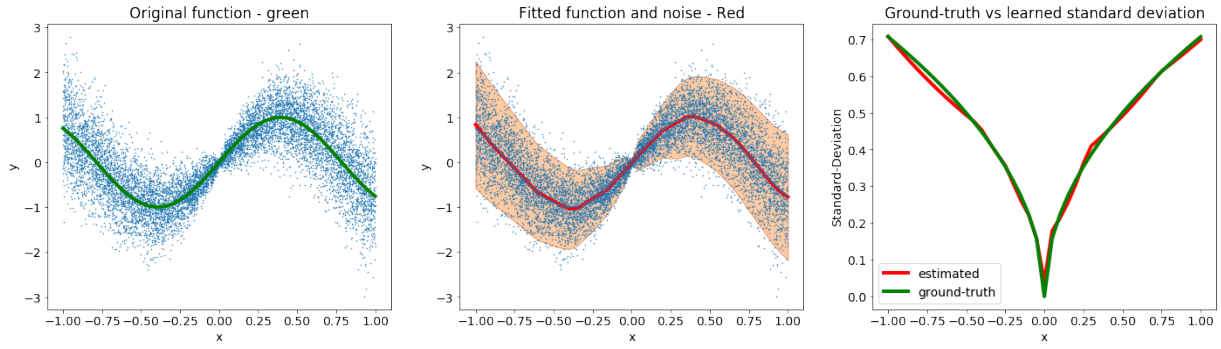

Figure 2.2: A simple example for heteroscedastic regression. In the leftmost figure, the blue dots are our training data which are produced by adding adaptive Gaussian noise to the function (green line). The middle figure shows the estimated mean (red line) and 95% confidence interval (orange region). The rightmost figure shows the estimated and ground-truth variance. We can see that the model is able to predict the variance accurately.

2.3.1 Bayesian Modeling for Machine Learning

While the non-Bayesian approach treats the model's weight ω as deterministic variables, the Bayesian approach considers them as random variables and thus assigns a prior distribution over these weight values:

$$\omega \sim p(\omega) \tag{2.9}$$

This prior distribution is updated when new data is given. When we have a dataset $\mathbf{D} = \{\mathbf{X}, \mathbf{Y}\}$, then the posterior is:

$$p(\omega|\mathbf{X}, \mathbf{Y}) = \frac{p(\mathbf{Y}|\mathbf{X}, \omega)p(\omega)}{p(\mathbf{Y}|\mathbf{X})} \tag{2.10}$$

For both classification and regression tasks, given the weight values ω, we can always estimate the likelihood $p(\mathbf{y}|\mathbf{x}, \omega)$. This can be done by using a softmax classifier (in classification) or heteroscedastic regression (in regression). Therefore, the term $p(\mathbf{Y}|\mathbf{X}, \omega)$ is easy to calculate. For the denominator in the Equation 2.10, we have:

$$p(\mathbf{Y}|\mathbf{X}) = \int p(\mathbf{Y}|\mathbf{X}, \omega)p(\omega)d\omega \tag{2.11}$$

In the case of DNNs, the integral in the Equation 2.11 does not have any closed-form solution. Indeed, to the best of our knowledge, the closed-form solution only exists in the case of Bayesian linear regression. Thus, it is also intractable to calculate the posterior

$p(\omega|\mathbf{X}, \mathbf{Y})$ according to the Equation 2.10. Therefore, the predictive distribution for a test input $\hat{\mathbf{x}}$:

$$p(\hat{\mathbf{y}}|\hat{\mathbf{x}}, \mathbf{X}, \mathbf{Y}) = \int p(\hat{\mathbf{y}}|\hat{\mathbf{x}}, \omega)p(\omega|\mathbf{X}, \mathbf{Y})d\omega \tag{2.12}$$

is also intractable. We note that any attempt to use the Monte-Carlo (MC) approximation method for computing the predictive distribution requires knowledge of $p(\omega|\mathbf{X}, \mathbf{Y})$ (so that we can draw samples from it). Therefore, estimating or approximating the posterior $p(\omega|\mathbf{X}, \mathbf{Y})$ is a major task for Bayesian machine learning research. Next, we will introduce one practical way to approximate the posterior by using the dropout method. For a more general understanding of the approximation technique in Bayesian machine learning, we suggest the readers to the work by Y.Gal [22].

2.3.2 Bayesian Neural Networks with MC-dropout

One approach to obtain an approximated BNN model from an existing DNN architecture is by inserting dropout layers and training the new model with dropout training [23]. After we have trained our model with dropout, if we keep the dropout layers activated, we are sampling from a distribution $q_\theta(\omega)$, which is an approximation of the posterior $p(\omega|\mathbf{X}, \mathbf{Y})$ that we have shown above. For more details about this method, we refer the reader to consult the original work by Gal, Yarin, and Zoubin Ghahramani [23].

At test time, specifically, for a given input, we perform multiple forward predictions in the network while keeping the dropout layers active. In other words, we remove a percentage of randomly-selected units (i.e., set the weight values of their connections to 0) from the trained model in order to obtain a sample prediction for the given input; then we repeat this process T times and calculate the average prediction. This technique at test time is referred to as Monte-Carlo (MC) dropout.

In classification We use $p(\hat{\mathbf{y}} = k|\hat{\mathbf{x}}, \omega_i)$ to denote the probability that the label $\hat{\mathbf{y}}$ of the test input $\hat{\mathbf{x}}$ is the kth class, which is given by a model (trained with dropout on $\mathbf{X}, \mathbf{Y}$) with a set of sampled weight values ω_i as its softmax output. Then, we wish to capture the mean probability $p(\hat{\mathbf{y}} = k|\hat{\mathbf{x}})$ for a test point $\hat{\mathbf{x}}$, which can be calculated as:

$$p(\hat{\mathbf{y}} = k|\hat{\mathbf{x}}) \approx \frac{1}{T}\sum_{i=1}^{T} p(\hat{\mathbf{y}} = k|\hat{\mathbf{x}}, \omega_i) \tag{2.13}$$

where T is the number of MC-dropout samples and ω_i is a set of weight values for each MC-dropout sample. We note that calculating the probability for every class will give us a categorical distribution over classes for the input $\hat{\mathbf{x}}$.

In regression Similar to the case of classification, for each sampled model, we obtain a Gaussian distribution $\mathcal{N}(\mu_i, \sigma_i^2|\hat{\mathbf{x}})$. The mean distribution $p(\hat{\mathbf{y}}|\hat{\mathbf{x}})$ can be obtained as follow:

$$p(\hat{\mathbf{y}}|\hat{\mathbf{x}}) \approx \frac{1}{T}\sum_{i=1}^{T}\mathcal{N}(\mu_i, \sigma_i^2|\hat{\mathbf{x}}) \tag{2.14}$$

In practice [39, 69], we can further approximate $p(\hat{\mathbf{y}}|\hat{\mathbf{x}})$ with a Gaussian distribution:

$$p(\hat{\mathbf{y}}|\hat{\mathbf{x}}) \approx \mathcal{N}(\hat{\mu}, \hat{\sigma}^2|\hat{\mathbf{x}}) \tag{2.15}$$

where $\hat{\mu} = \frac{1}{T}\sum_{i=1}^{T}\mu_i$ and $\hat{\sigma}^2 = \frac{1}{T}\sum_{i=1}^{T}\sigma_i^2 + \frac{1}{T}\sum_{i=1}^{T}\mu_i^2 - \hat{\mu}^2$.

2.3.3 Other methods

For the rest of this thesis, unless specified, we will use MC-Dropout as a method to estimate the model uncertainty. We outline here several methods that can be used as an alternative for MC-Dropout.

In their work, Blundell, et al.[6] introduce a method to estimate the weight uncertainty by using the backpropagation algorithm. The idea is to parameterize each weight value by a mean and standard deviation of a Gaussian distribution. Each mean and standard deviation is then learned by using the backpropagation algorithm. At test time, we sample the weight values and compute the uncertainty estimates. Unlike MC-Dropout, this method requires a twice amount of weight parameters.

Teye, et al.[81] show that training deep models with batch-normalization layer can also be considered as training a Bayesian Neural Network. This approach is very interesting because it eliminates the need for dropout layers (and tuning the dropout rate), however, this method requires accessing the training dataset at test time, which means that we need additional space (and possibly time) to access the data.

Lakshminarayanan, et.al [52] provide a simple method for obtaining the model uncertainty by training different deep models with the same architecture but different initialization. In their experiment, by using 5 deep networks, they can achieve good accuracy and

uncertainty estimates in terms of outlier detection. We note that except for the ensemble idea (similar to MC-Dropout), this method has yet to be shown to have any connection to Bayesian modeling in terms of parameter estimation.

2.4 Bayesian Uncertainty Quantification

In this section, we describe the two types of uncertainty (aleatoric and epistemic), and how to extract measures of these types of uncertainty from a BNN's predictions.

2.4.1 Types of uncertainty

Aleatoric uncertainty represents the irreducible noise in the data and cannot be reduced even when we gather more data [39]. For example, in the binary classification problem where we have a large amount of data, the data that lie within the intersection region of the two class distributions will have higher aleatoric uncertainty than the data in either distribution but outside the intersection.

Epistemic uncertainty captures the model's lack of knowledge due to the limitation in the training data (such as bias, scarcity, novelty, etc.) [22]. It can be reduced by gathering more training data. Bayesian modeling allows us to quantify both types of uncertainty. Although approaches exist for estimating aleatoric uncertainty with non-Bayesian approaches [39] (as we have shown earlier), they cannot be used to estimate epistemic uncertainty. Furthermore, non-Bayesian models tend to perform poorly and give overconfident predictions in regions that lack data [22]. Bayesian approaches to neural networks, on the other hand, allow us to capture the epistemic uncertainty [39] and, thus, BNN models tend to give predictions with high uncertainty in low-density regions. Unlike aleatoric uncertainty, this uncertainty can be reduced by increasing the amount of data. We illustrate these properties of the two uncertainties in a regression task in Figure 2.3.

2.4.2 Uncertainty Estimation - Classification

There are three metrics that we will use in the experiments, namely: predictive entropy $\mathbb{H}(\hat{\mathbf{y}}|\hat{\mathbf{x}})$ (captures total uncertainty), mutual information $\mathbb{MI}(\hat{\mathbf{y}}|\hat{\mathbf{x}})$ (captures epistemic uncertainty), and aleatoric entropy $\mathbb{AE}(\hat{\mathbf{y}}|\hat{\mathbf{x}})$ (captures aleatoric uncertainty). These uncertainty estimates can be calculated as follows [15, 65].

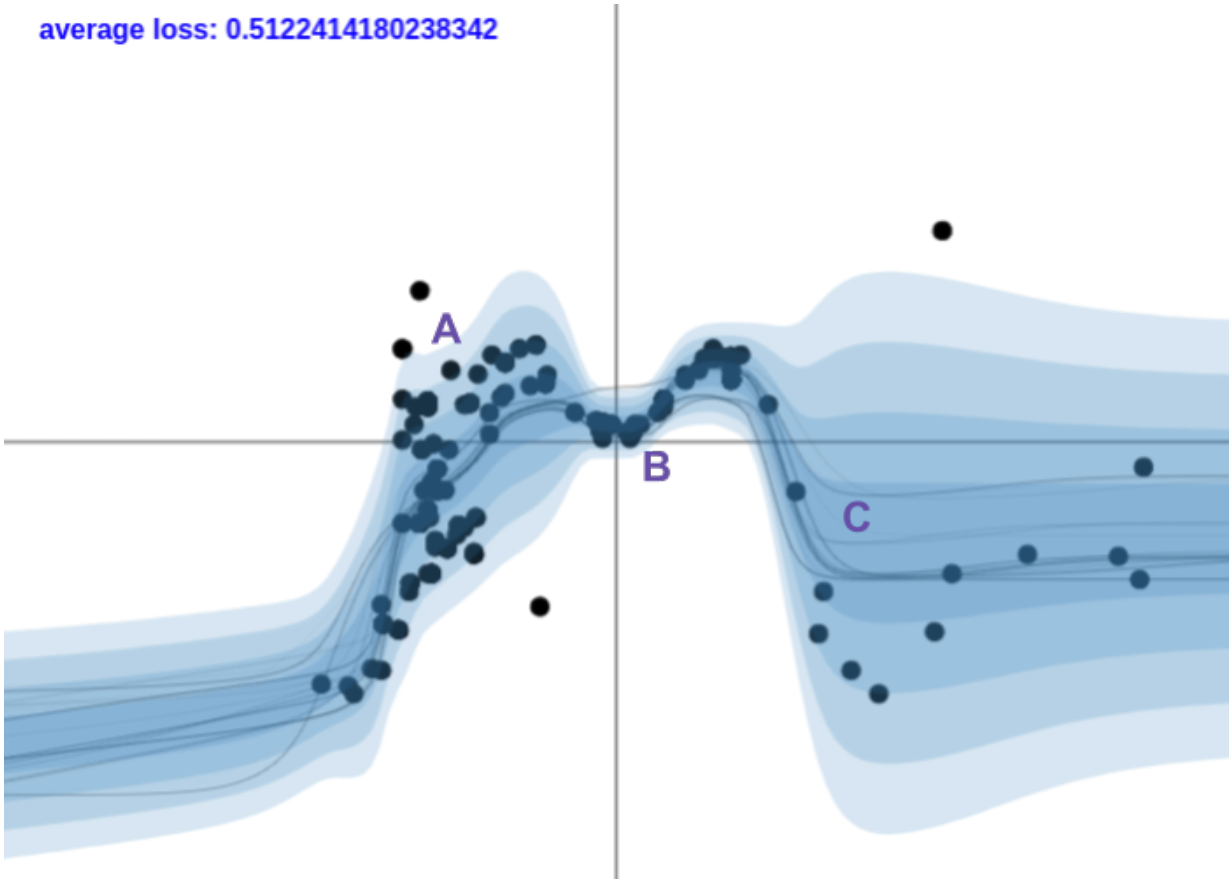

Figure 2.3: A simple example for BNN regression. The dark line is a sampled model mean while the blue region is the confidence interval region for the distribution $p(y|x)$. In the dense but noisy data region in the left (A), the sampled models from $p(\theta|D)$ stay close together (low epistemic uncertainty) and the aleatoric uncertainty is high. The middle region B shows high density and a little noisy region, which is reflected in the epistemic uncertainty (the drawn model stays close to each other) and low aleatoric uncertainty. The right region C shows a sparse data region. The sampled models are very different and the uncertainty estimate overall is high. This figure was created using the tool in this website: http://mlg.eng.cam.ac.uk/yarin/blog_2248.html

Predictive Entropy captures the total amount of uncertainty (epistemic and aleatoric) and is equal to the sum of mutual information and aleatoric entropy. It is calculated as the entropy of the mean categorical distribution:

$$\begin{aligned}\mathbb{H}(\hat{\mathbf{y}}|\hat{\mathbf{x}}) &= \mathbb{MI}(\hat{\mathbf{y}}|\hat{\mathbf{x}}) + \mathbb{AE}(\hat{\mathbf{y}}|\hat{\mathbf{x}}) \\ &= -\sum_{k} p(\hat{\mathbf{y}} = k|\hat{\mathbf{x}}) \log p(\hat{\mathbf{y}} = k|\hat{\mathbf{x}})\end{aligned} \tag{2.16}$$

Aleatoric Entropy captures the aleatoric uncertainty and is calculated by averaging the entropy of each sampled categorical distribution. This type of uncertainty should converge to the predictive uncertainty when the amount of training data increases.

$$\mathbb{AE}(\hat{\mathbf{y}}|\hat{\mathbf{x}}) = -\frac{1}{T}\sum_{k,i} p(\hat{\mathbf{y}} = k|\hat{\mathbf{x}}, \omega_i) \log p(\hat{\mathbf{y}} = k|\hat{\mathbf{x}}, \omega_i) \tag{2.17}$$

Mutual Information captures the epistemic uncertainty and is calculated as:

$$\mathbb{MI}(\hat{\mathbf{y}}|\hat{\mathbf{x}}) = \mathbb{H}(\hat{\mathbf{y}}|\hat{\mathbf{x}}) - \mathbb{AE}(\hat{\mathbf{y}}|\hat{\mathbf{x}}) \tag{2.18}$$

Intuitively, it measures the disagreement between different sampled models and should converge to 0 when the amount of data increases.

2.4.3 Uncertainty Estimation - Regression

In the case of regression, variance can be used as a measure of uncertainty. We can extract the uncertainty as follows [15]:

Epistemic Variance is calculated by measuring the variance of the mean over the predicted samples.

$$\hat{\sigma}^2 = \frac{1}{T}\sum_{i=1}^{T} \mu_i^2 - \hat{\mu}^2 \tag{2.19}$$

Aleatoric Variance is calculated by measuring the mean of the predicted variances.

$$\hat{\sigma}^2 = \frac{1}{T}\sum_{i=1}^{T} \sigma_i^2 \tag{2.20}$$

Predictive (Total) Variance is calculated by summing both aleatoric and epistemic variance.

$$\hat{\sigma}^2 = \frac{1}{T}\sum_{i=1}^{T}\sigma_i^2 + \frac{1}{T}\sum_{i=1}^{T}\mu_i^2 - \hat{\mu}^2 \tag{2.21}$$

2.5 Kullback–Leibler Divergence

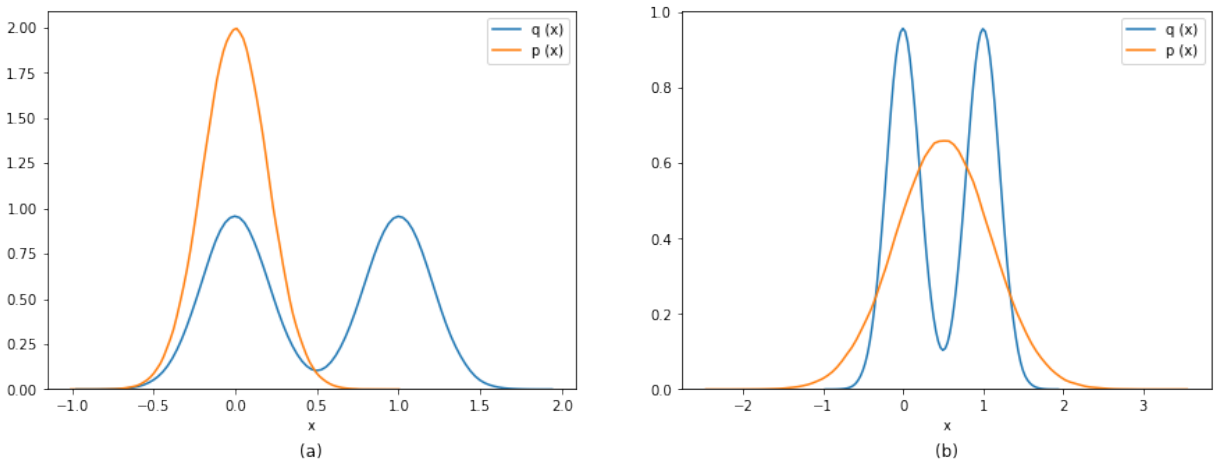

Figure 2.4: An example that illustrates the asymmetry property of the KL divergence. Figure 2.2(a) shows the solution for $\arg\min_q \mathbb{D}_{\mathrm{KL}}(p||q)$. Figure 2.2(b) shows the solution for $\arg\min_q .\mathbb{D}_{\mathrm{KL}}(q||p)$

.

We end this chapter with a definition of Kullback–Leibler (KL) divergence, a measurement of how two distributions are different from each other. Given two probability distribution $p(x)$ and $q(x)$, the KL divergence is defined as:

$$\mathbb{D}_{\mathrm{KL}}(p||q) = \int_{-\infty}^{\infty} p(x)\log\left(\frac{p(x)}{q(x)}\right)dx \tag{2.22}$$

One property of this quantity is that it is always non-negative, that is:

$$\mathbb{D}_{\mathrm{KL}}(p||q) \geq 0 \tag{2.23}$$

It is equal to 0 if and only if $p = q$ almost everywhere. This can be interpreted as a distance between the two distributions. However, we note that unlike normal distance metric, it is asymmetric:

$$\mathbb{D}_{\mathrm{KL}}(p||q) \neq \mathbb{D}_{\mathrm{KL}}(q||p) \tag{2.24}$$

This implies that if we have a distribution p and we want to fit a distribution $q(x) = Z_\theta(x)$ that is "close" to p, then:

$$\arg\min_\theta \mathbb{D}_{\mathrm{KL}}(Z_\theta(x)||p) \neq \arg\min_\theta \mathbb{D}_{\mathrm{KL}}(p||Z_\theta(x)) \tag{2.25}$$

This can be illustrated through the example in Figure 2.4. In this example, $p(x)$ is a Gaussian mixture model and we want to find $q(x) = \mathcal{N}(\mu, \sigma^2)$ such that it is "similar" to $p(x)$. We can see that if we optimize based on the $\arg\min_q \mathbb{D}_{\mathrm{KL}}(p||q)$ objective, $q(x)$ fits one mode of the mixture distribution. On the other hand, the result based on the other objective tends to fit the whole coverage of $p(x)$.

Chapter 3

Bayesian Uncertainty Quantification with Synthetic Data

Image semantic segmentation systems based on deep learning are prone to making erroneous predictions for images affected by uncertainty influence factors such as occlusions or inclement weather. Bayesian deep learning applies the Bayesian framework to deep models and allows estimating so-called epistemic and aleatoric uncertainties as part of the prediction. Such estimates can indicate the likelihood of prediction errors due to the influence factors. However, because of lack of suitable experimental data, the effectiveness of Bayesian uncertainty estimation when segmenting images with varying levels of influence factors has not yet been systematically studied.

In this chapter, we propose using a synthetic dataset to address this gap. We conduct two sets of experiments to investigate the influence of distance, occlusion, clouds, rain, and puddles on the estimated uncertainty in the segmentation of road scenes. The experiments confirm the expected correlation between the influence factors, the estimated uncertainty, and accuracy. Contrary to expectation, we also find that the estimated aleatoric uncertainty from Bayesian deep models can be reduced with more training data. We hope that these findings will help improve methods for assuring machine-learning-based systems.

3.1 Introduction

Deep neural network (DNN) models, although having achieved many state-of-the-art results on a variety of tasks in computer vision [33, 73, 8], are not perfect as their performance

much depends on the input images. Since errors in prediction due to the input characteristics are inevitable, several uncertainty metrics have been proposed as failure indicators for potentially faulty predictions [23, 17, 30]. Before deploying these metrics as failure indicators into safety-critical applications such as autonomous driving (AD) and medical diagnosis, their behaviour should be studied extensively under different scenarios in order to improve safety assurance.

Image semantic segmentation systems based on deep learning are prone to making erroneous predictions for images affected by uncertainty influence factors such as occlusions or inclement weather. Bayesian deep learning applies the Bayesian framework to deep models and allows estimating so-called epistemic and aleatoric uncertainties as part of the prediction. Such estimates can indicate the likelihood of prediction errors due to the influence factors. Aleatoric uncertainty represents the irreducible source of errors in the data (e.g., noise), and thus cannot be reduced by providing more data. In contrast, epistemic uncertainty represents the model's "lack of knowledge" about the problem and can be reduced with more training data (more details in Sec. 2.4.1). Inputs with high aleatoric uncertainty can be thought as inherently ambiguous, whereas inputs with high epistemic uncertainty can be viewed as "unexpected", i.e., far from the training dataset. However, because of lack of experimental data, the effectiveness of Bayesian uncertainty estimation when segmenting images with varying levels of influence factors has not yet been systematically studied.

In this chapter, we propose using synthetic data to investigate and study the effects of selected factors on BNN's uncertainty in the task of image semantic segmentation. In particular, we consider scene-specific factors: depth, occlusion, clouds, rain, and puddles. These factors can lead to ambiguous inputs, e.g., due to reduced information about the underlying objects in the scene because of their far distance or high occlusion level, but also unexpected inputs because of changed appearance, e.g., due to rain or puddles, if not trained for. We perform two sets of experiments, with the following results:

1. We study and quantify the uncertainty estimates of a state-of-the-art BNN under different levels of occlusion and depth for vehicles. As expected, we find a correlation between the influence factors, the estimated uncertainty, and accuracy, which is desirable for a failure indicator. Contrary to expectation, we find that the estimated aleatoric uncertainty from a BNN *can* change with more training data.

2. We explore and report on the behavior of BNN's uncertainty estimates under different weather effects. We find that cloud level has much more significant impact on the uncertainty estimates than rain and puddles—which correlates with the higher

negative impact of clouds on the network performance than that of rain or puddles, as previously reported for the same dataset and network [44].

The chapter is structured as follows. Section 3.2 briefly describes the ProcSy dataset, which we use in this research. Section 3.3 describes the modification in the Deeplab v3+ architecture so that we can estimate the uncertainties in the image segmentation task. Section 3.4 presents two experiments using ProcSy to study the effects of influence factor variations on the uncertainty estimates. Finally, we conclude the chapter and suggest future research directions in Section 3.5.

3.2 The ProcSy Dataset

This section briefly describes the ProcSy synthetic dataset, which we have developed in previous work [44], and use for our experiments. Being synthetically generated, ProcSy holds several benefits for studying uncertainty estimation. Section 3.2.1 explains these benefits in detail. Section 3.2.2 summarizes the content of the ProcSy dataset.

3.2.1 Why Synthetic Data?

In the context of autonomous driving, factors such as depth, amount of occlusion, and weather effects can produce ambiguous and unexpected inputs to the model. Studying effects of these factors with a real-world dataset is difficult, although this is desirable. Current segmentation datasets with weather effects such as Raincouver [83] have a limited quantity of data to work with. Raincouver, for instance, is meant to be used as an addendum to an existing dataset such as Cityscapes rather than by itself, as it only contains 326 finely annotated images.

Berkeley Deep Drive [87] is a more recent dataset that shows more promise in the quantity of data availability (5683 finely annotated images). However, this dataset suffers from labeling inconsistency issues. These limitations make the currently available real-world segmentation datasets impractical for model uncertainty analysis. Using these datasets to supplement a high-quality segmentation dataset such as Cityscapes [10] (which has no weather effects) is also problematic, since there is a qualitative difference among the datasets.

It is also very often the case that data acquired in the real world is not repeatable under different conditions. For instance, a scene captured in ideal conditions may not be

reproducible during a day with heavy rain, because a car that was originally parked in the scene is no longer there. This sort of logistical issues can prove to be very expensive to overcome in generating a real-world dataset. It is also not an easy task to annotate effects such as amount of occlusion in a real-world dataset. On the other hand, synthetic data rendered by recent computer graphics technology can provide various influence factor effects with minimal labour cost. Due to these reasons, in this chapter, we use our ProcSy synthetic dataset to analyze the effect of influence factors on the uncertainty estimates of a model.

3.2.2 Dataset Summary

Our ProcSy dataset is comprised of road scenes captured from a virtual render of a 3 km^2 map region of urban Canada. From this environment, 11,000 scene frames were curated to contain no visible artifacts such as clipping of the camera through vehicles and pedestrians. These curated frames were then split into 8000 training frames, 2000 validation frames, and 1000 test frames.

For each frame of the ProcSy dataset, along with the base RGB image, we have generated corresponding ground truth annotation labels, depth map data, and occlusion maps of the different vehicle types present in the scene.

We generate weather variations in the categories of rain, puddle, and cloud coverage. For each of these factors, we consider five different intensity levels. These are 0%, 25%, 50%, 75%, and 100%. Figure 3.1 shows an example frame with different intensity levels for each of the three weather factors. For our training set of 8000 images, we first consider three equal subsets for the weather categories. Within each subset, we further divide into intensity levels and render RGB images reflecting variations in these subsets. This allows us to carry out experiments without having to generate every combination of influence factor variations.

3.3 Bayesian Neural Networks for Image Segmentation

Semantic segmentation is a task that assigns a class label to each pixel of an image. For autonomous driving, the image segmentation system enables the vehicle to perceive the visual state of the world. Since deep convolutional architectures consider this task as

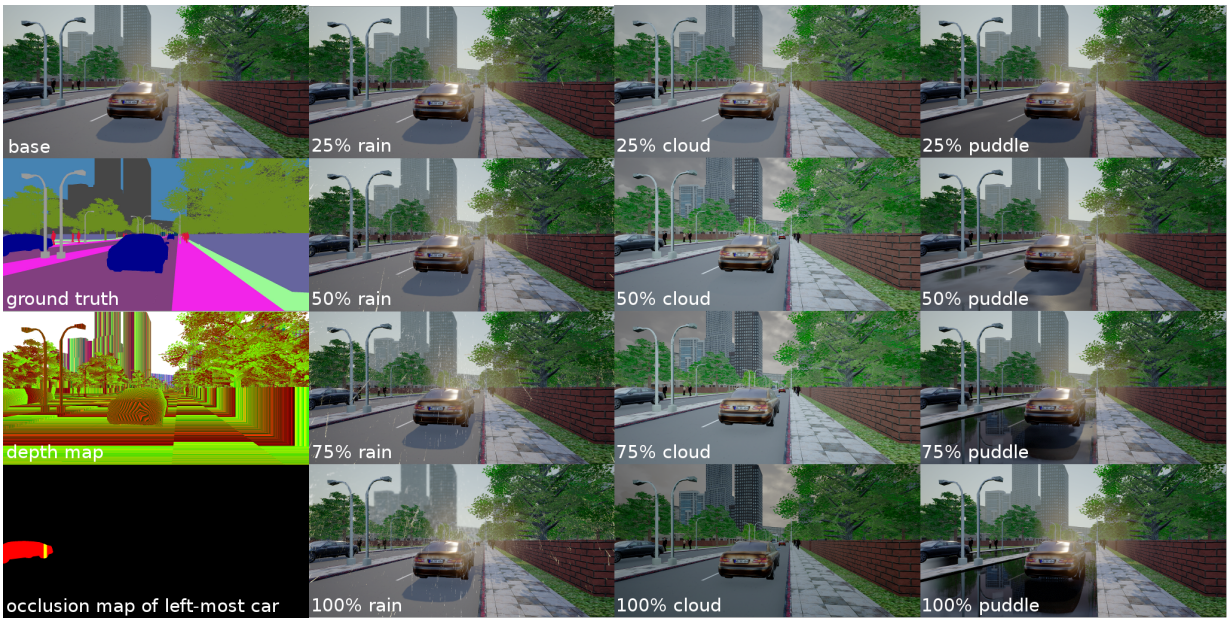

Figure 3.1: Weather variations visualizing intensity level differences in the three weather categories — rain, cloud, and puddle; ground truth, depth map, and occlusion map for one vehicle type are also shown along with the base RGB image

classifying each pixel independently using the same network [57], the BNN approach can be applied to this family of architectures to estimate the uncertainty per pixel.

For the experiments in this chapter, we use Deeplab v3+ [9], one of the state-of-the-art models for segmentation, with ResNet 50 backbone architecture [33]. We inserted dropout layers with rate of 0.5 at four blocks in the middle of the backbone (specifically, at the end of the 8^{th} till 11^{th} block). The basis for this setup is based on the studies by Kendall et al. [38] and Mukhoti et al. [65], from which they empirically determined that inserting dropout layers in the middle flow yields a better predictive performance than for other positions in a network. Figure 3.2 shows examples of the three uncertainty types estimated by a BNN.

3.4 Experiments with Synthetic Data

In this section, we perform two experiments with two set of influence factors. The first experiment involves the amount of depth and occlusion as factors, whereas the second experiment involves different weather effects, specifically: clouds, rain, and puddles. The reason why we treat them separately is that occluded and distant objects necessarily occur in the training set, whereas the latter factors do not.

We train two Bayesian Deeplab v3+ models: model A with 3000 clean images and model B with 8000 clean images. We note that the set of 3000 clean images is a subset of the set of 8000 images. Model A and model B are trained with $75,000$ and $180,000$ iterations, respectively, with a batch size of 16 and crop size of 512x512. The uncertainty estimates are calculated based on 50 MC-dropout samples.

3.4.1 Occlusion and Depth

In this experiment, we study how different variations of occlusions and depth factors affect the uncertainty estimates. We test and measure the uncertainty estimates (aleatoric, epistemic, and predictive) of model A and B using a test set consisting of 270 clean images containing a total of 1,200 vehicles. The amount of occlusion for each vehicle is determined by calculating the fraction of the number of occluded pixels over the total number of the vehicle's pixels. We then assign this occlusion level to each visible pixel of the vehicle. For each model, we partition the pixels into subsets based on amount of occlusion and distance (each discretized into five intervals of 20%). Then we calculate the mean accuracy and uncertainty estimates for the model predictions of the pixels in each subset. Finally, we

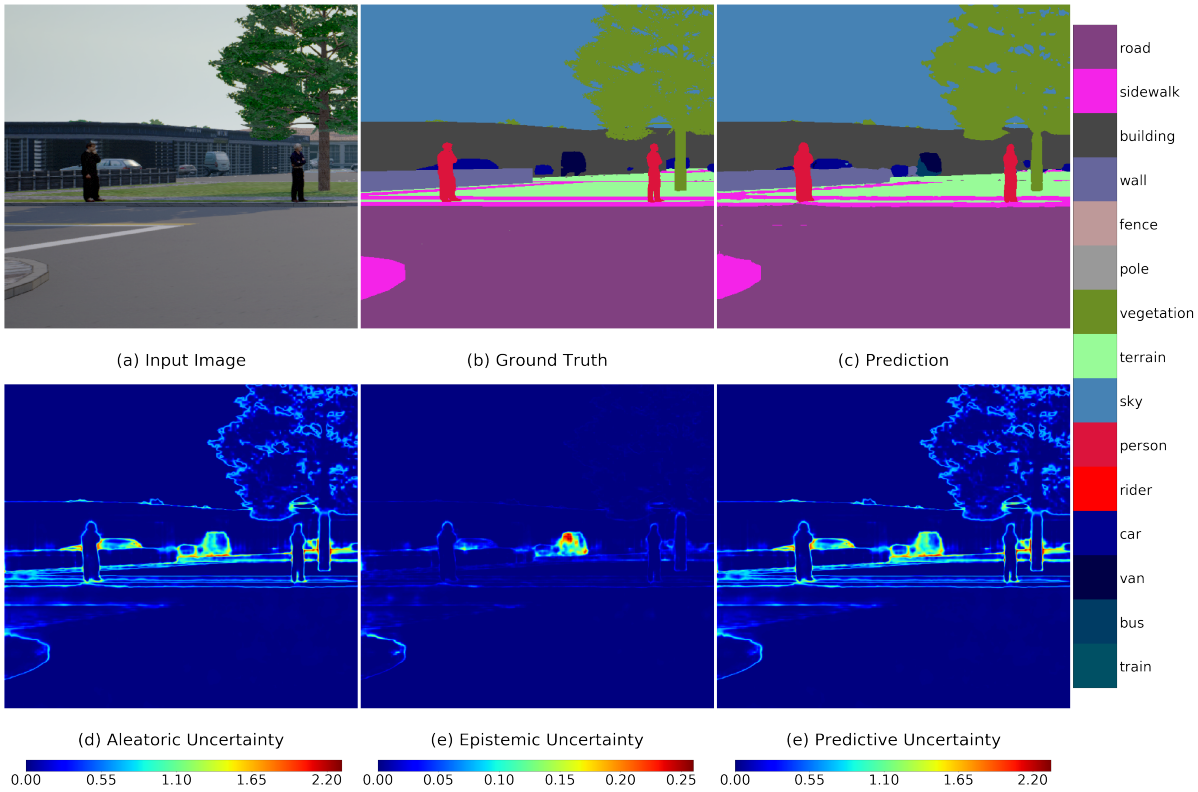

Figure 3.2: Illustration for different types of uncertainty estimates in semantic segmentation. (a),(b),(c) show the input image, ground truth and prediction, respectively. (d),(e),(f) show the estimated aleatoric, epistemic and predictive uncertainty from our model. It can be visually observed that the aleatoric uncertainty is high at the class boundary (e.g., the tree). On the other hand, the epistemic uncertainty estimates are high only at several specific regions, such as the cluster in the middle of (e). This model is trained with 3000 clean images (model A in Section 3.4).

use cubic spline interpolation to obtain a contour plot. The results are shown in Figure 3.3.

According to the definition of aleatoric and epistemic uncertainty, we expect that model B's epistemic uncertainty estimates should be lower than model A's, whereas the aleatoric estimates of the two models should stay similar. Based on the results in Figure 3.3, we make the following observations:

1. Model B has higher accuracy and lower epistemic uncertainty than model A in general. This fits with expectations since model B is trained with more data than model A.

2. The predictive uncertainty can be observed to be correlated well with the accuracy, which is a desirable characteristic for a failure indicator. The Pearson correlation coefficient between predictive uncertainty and accuracy for model A is -0.89 and for model B is -0.90.

3. For both models A and B, the aleatoric estimates increase for objects that are more occluded and further away from the camera as expected. The same behavior can also be observed for the predictive uncertainty estimates.

4. There is a difference between the aleatoric estimates of the two models, which is surprising. Specifically, the difference between aleatoric estimates seems to increase with the epistemic difference. To validate this observation, we plot in Figure 3.4 the difference between those two uncertainty estimates according to the amount of occlusion and depth, then we calculate the Pearson correlation coefficient of these quantities. The results reflect this observation as the Pearson coefficient, which values is 0.579, implies that there is a relation between the two quantities.

We hypothesize that the reason why we observe the aleatoric uncertainty estimate changing when more training data is provided is that the estimate is only reliable where we have sufficient data. For regions with a sufficient amount of data, the decision boundary of model A and B are similar to each other. Thus, adding more data would not likely change the estimated aleatoric uncertainty (unless the data we already have is biased). On the other hand, the decision boundaries in regions that lack data tend to be inconsistent making the aleatoric estimate unreliable.

Finally, we note that for model B, high aleatoric estimates still occur for regions that have relatively low epistemic uncertainty, such as the middle top region in Figure 3.3f,g where the objects are occluded around 50% and far away from the camera. This suggests

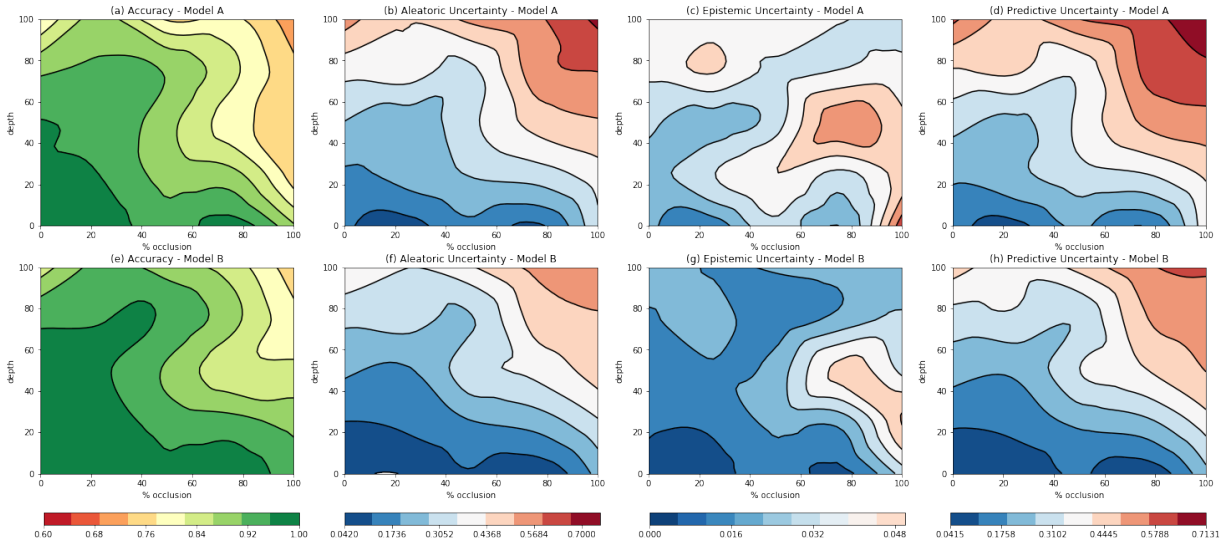

Figure 3.3: The two rows show the accuracy, aleatoric, epistemic and predictive uncertainty estimates according to level of depth and occlusion of model A and B, respectively. Each color bar reflects the metric values of the plots in the corresponding column.

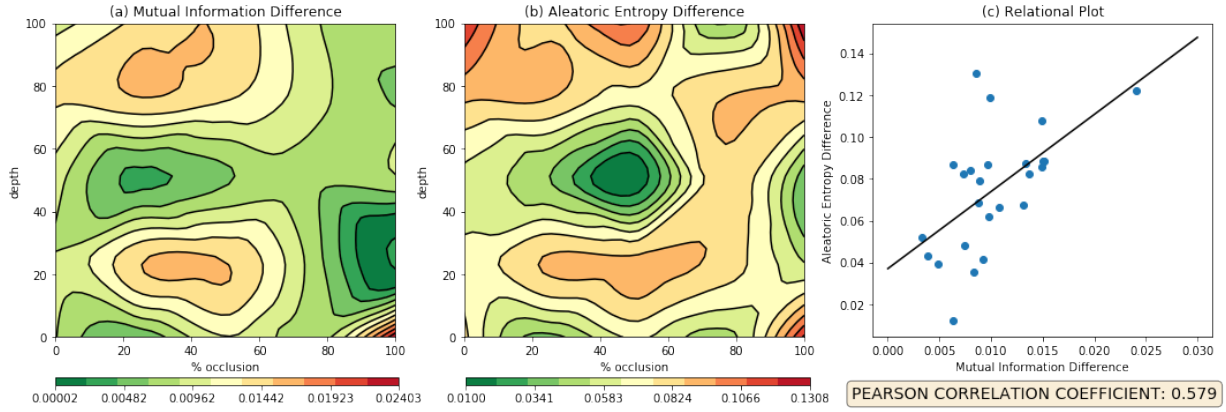

Figure 3.4: (a) shows the difference between the estimated mutual information for model B and A according to occlusion and depth. (b) shows the equivalent difference for aleatoric entropy.(c) shows the relational plot between the two differences. Each blue dot represents the difference between the aleatoric and epistemic estimates in a certain subset of occlusion and depth. The black line, which shows the relation between the two variables, is fitted by using linear regression.

that depth and occlusion are sources of aleatoric uncertainty in the image segmentation task.

To ensure perceptual performance in safety-critical application, developers must make sure that the training data satisfies the scenario coverage condition properly [12]. This experiment and analysis suggests that we can use the measure of epistemic uncertainty to determine the optimal amount of data to collect for occlusion and depth factors. Specifically, we should collect enough data to make the epistemic uncertainty map blue.

Extended Results We show the extension of this experiment in Figure 3.5. We note that the model's name in this extension results is different from the previous one (e.g, results in Figure 3.3). Model A, B, C, D is trained on 500, 3000, 8000 and 131000 images respectively. Similar to the previous results, the aleatoric and epistemic uncertainty estimates decrease as we increase the amount of training data. It is also more clear to see the interesting trend in the aleatoric estimates, where the uncertainty estimates keep being higher for more distant and occluded vehicles.

3.4.2 Weather

In this experiment, we study how the uncertainty estimates vary with respect to different intensity levels of weather effects, namely: clouds, rain, and puddles. We expect that as we increase the effect's intensity, the BNN model should have worse performance and output higher uncertainty estimates. This is because high intensity effects will introduce more artifacts that would cause misclassification in the model, thus the uncertainty estimate should increase to indicate this.

For each model and effect, we calculate the mIoU (mean Intersection over Union) and the mean aleatoric, epistemic, and predictive uncertainty estimates per pixel at every intensity level. Each effect's intensity level contains 150 images. The reason why we compare the performance of model A and B is that we want to observe how the uncertainty estimates change when we train with more in-distribution data. The results are shown in Figure 3.6.

In terms of mIoU (Figure 3.6d), we see that model B, which is trained with more in-distribution data, has better performance than model A when there is no factor involved. Further, model B's mIoU is higher than model A's for different intensity levels of rain and puddles. However, surprisingly for cloudiness, there is a sharp degradation in terms of mIoU for model B. Critically, at the 100% cloud intensity level, model A outperforms

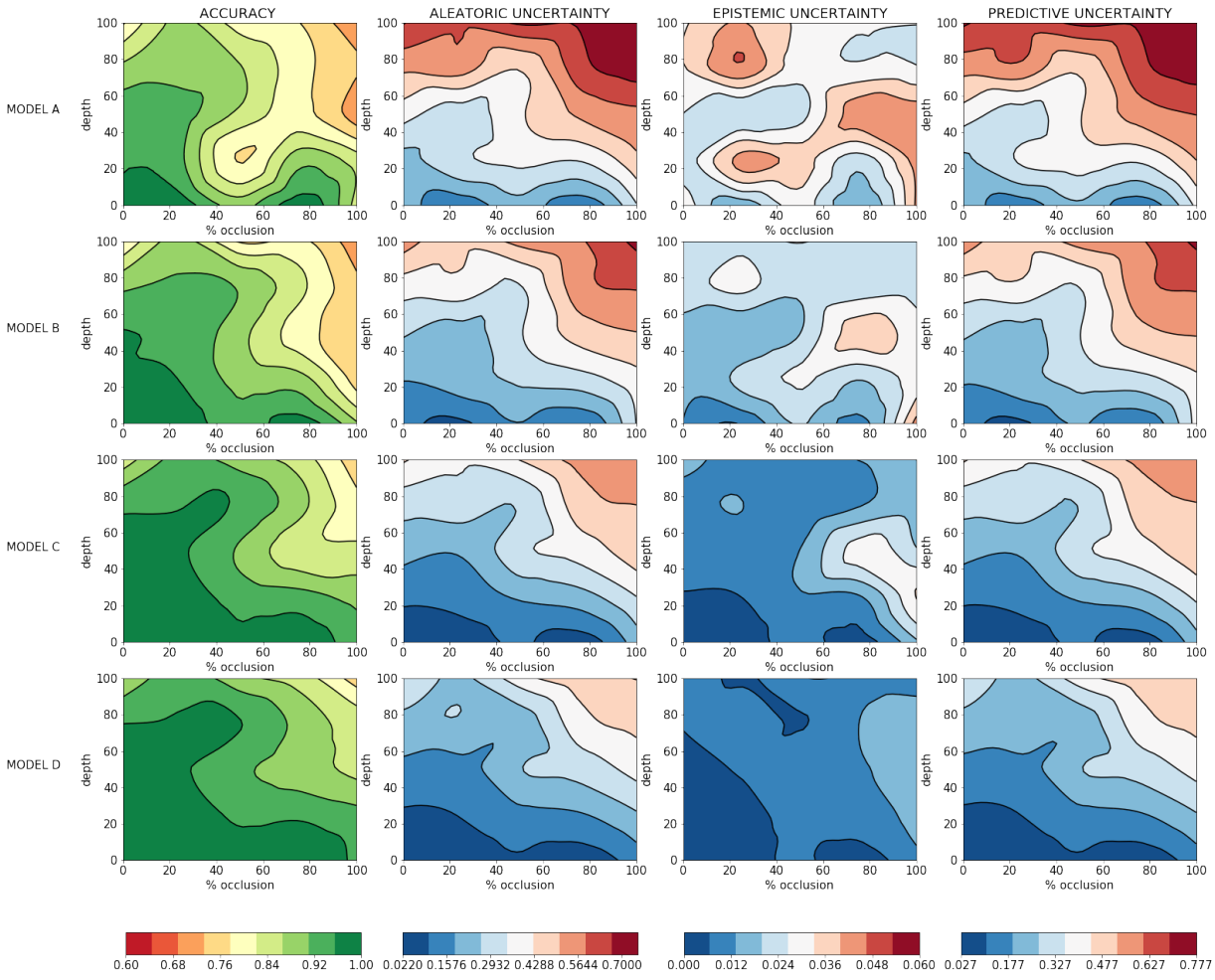

Figure 3.5: Each row shows the accuracy, aleatoric, epistemic and predictive uncertainty estimates according to level of depth and occlusion of the models.

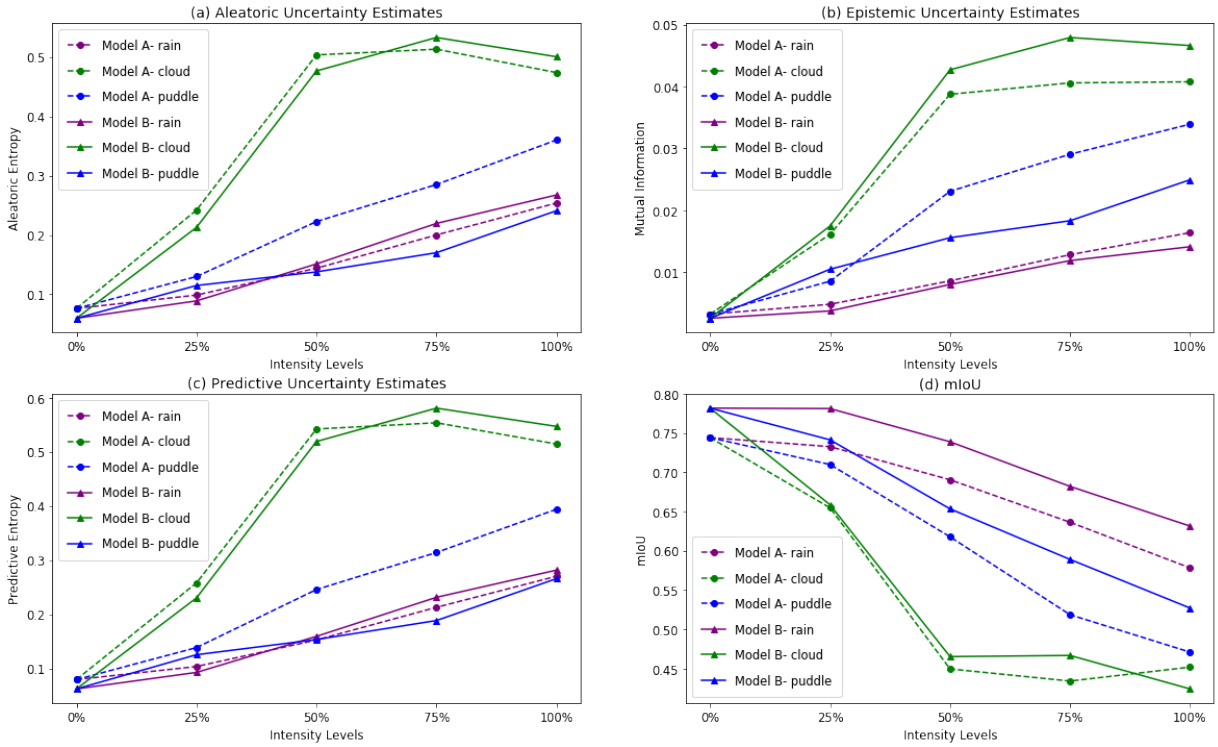

Figure 3.6: (a-d) show the estimated aleatoric, epistemic, and predictive uncertainties and mIoU values for different variations of weather, factors respectively. The x-axes represent different intensity levels.

model B (although with just a small margin). We find that in 100% cloud conditions, the two models fail to predict the following classes: pole, traffic lights, sky, bicycle, car and truck.

In terms of uncertainty, the three types of uncertainty estimates, in general, increase with the intensity levels for every factor. This behavior meets our expectation for the uncertainty and it applies to both model A and B. We notice that for clouds, there is a small decrease for the 50% to 100% levels, which requires further investigation.

We also make two following observations. First, for model B, we see that the epistemic uncertainty corresponds to the mIoU better than the other two uncertainties. Specifically, at every intensity level for each effects, the ascending order for epistemic estimates are rain, puddle and cloud, which corresponds to the descending of that order in mIoU. For model A, on the other hand, all the uncertainties reflect the mIoU well. Second, there is an inconsistency in terms of the difference between model A and B's uncertainty estimates for all the factors. For example, at 100% intensity level for rain and puddle, model B has lower epistemic uncertainty estimates than model A, yet it is higher in the case of cloudy images. This is unexpected since we assumed that training with more in-distribution data can only make the epistemic uncertainty lower or intact. This observation requires further investigation.

Extended Results We show the extension of this experiment in Figure 3.7 and 3.8 (simplified version of Figure 3.7). We note that the models name in this extension results is different from the previous one (e.g, results in Figure 3.6). Model A, B, C, D is trained on 500, 3000, 8000 and 131000 images respectively (similar to the previous extended results).

In the original experiment, we observe that the epistemic uncertainty corresponds to the mIoU better than the other two uncertainties. We would like to see if this still holds with other amount of training data. For both model A and D, we see that the epistemic uncertainty still reflects the mIoU well. However, we observe that for model D, the epistemic estimates at 50% puddle is lower than that at 75% puddle while the mIoU is decreasing. Finally, similar to model B (trained with 3000 images), all the uncertainty estimates of model A (trained with 500 images) corresponds well with the mIoU, while it is not the case for model C and D.

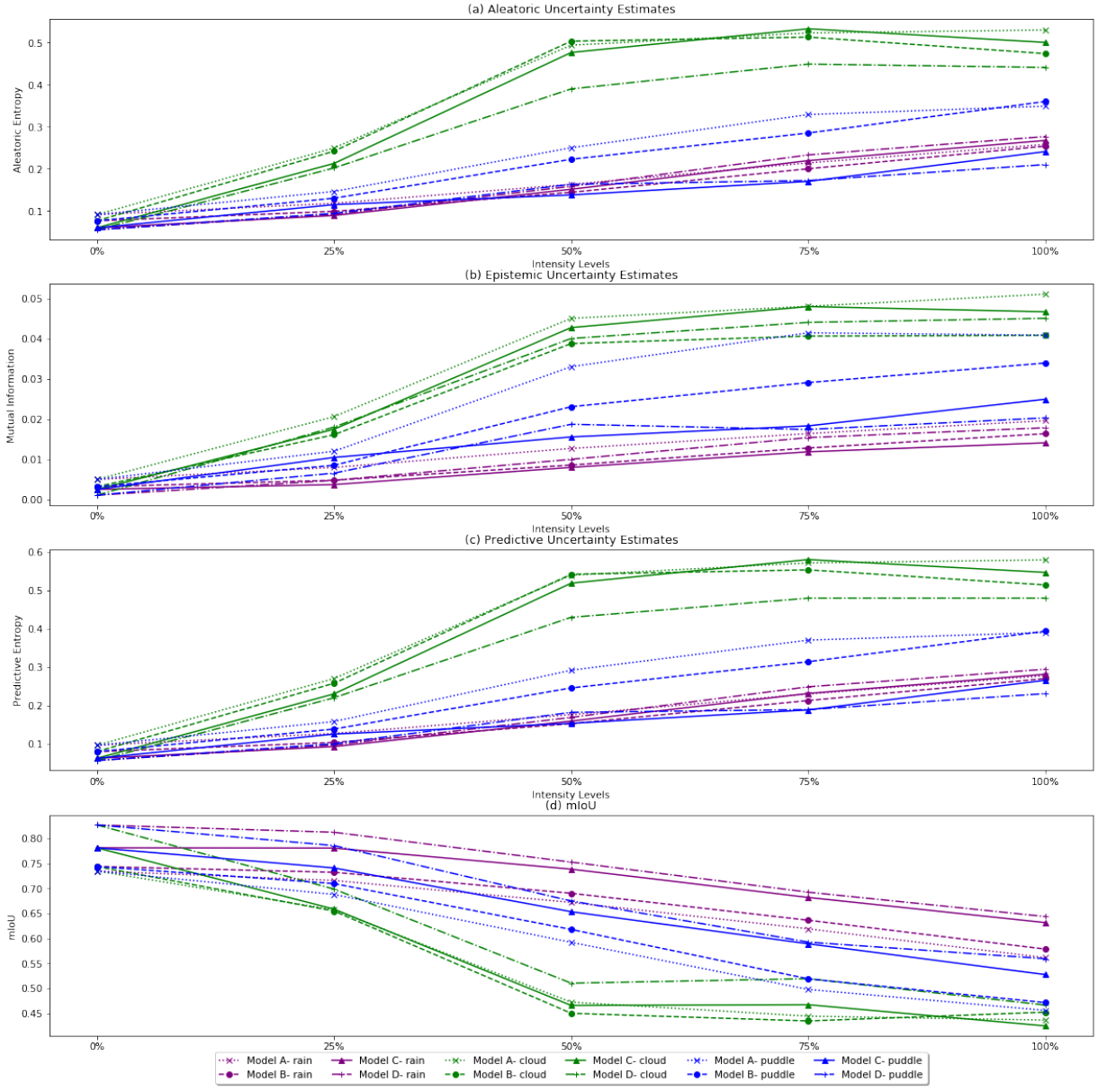

Figure 3.7: (a-d) show the estimated aleatoric, epistemic, and predictive uncertainties and mIoU values for different variations of weather, factors respectively. The x-axes represent different intensity levels.

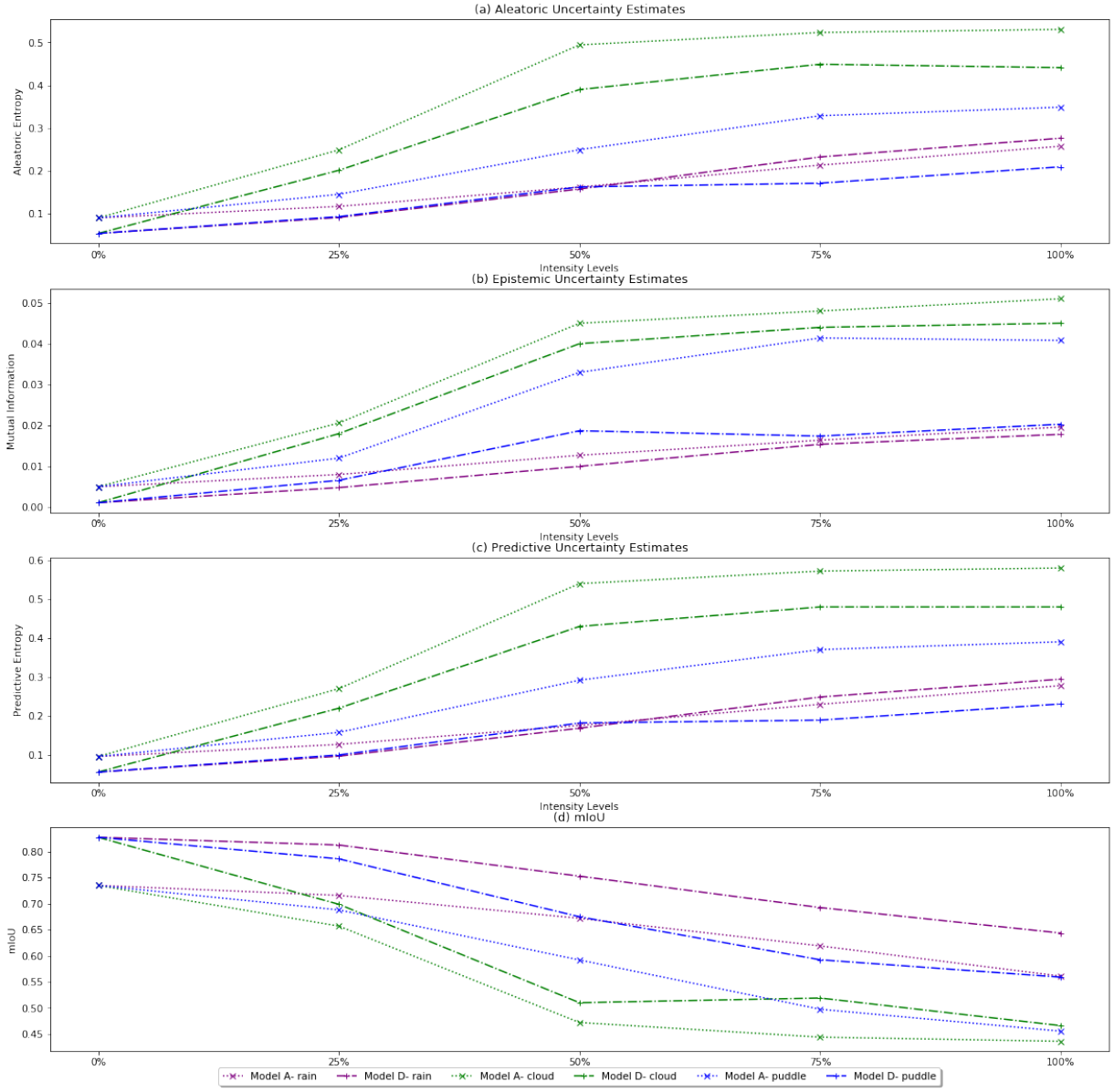

Figure 3.8: Simplified version of Figure 3.7, where we only show the behavior of model A and D.

3.5 Conclusions and Future Work

Reliable uncertainty estimation of ML predictions is important for the safe use of ML-based components. In this chapter, we use the ProcSy dataset to study the effects of different influence factors, namely: depth, occlusion, rain, clouds and puddles, on the uncertainty estimates of the BNN model for image segmentation.

In the experiments with occlusion and depth factors, our results show that the aleatoric uncertainty estimates is dependent on the epistemic uncertainty estimates. When given enough data, the epistemic uncertainty estimates reduce but the aleatoric estimates remain high for distant and occluded objects. Furthermore, we find that cloud affects the uncertainty estimates and mIoU values of the BNN more profoundly than rain and puddle, even when we have more in-distribution training data.

As we have mentioned in Sec. 3.4a, the experiment results suggest one potential application that requires further investigation: the possibility to use the epistemic uncertainty estimates to find an optimal amount of data for the occlusion and depth factors. Furthermore, it would be beneficial to extend this work to understanding the risk potential of these factors. Finally, while this chapter studies how the influence factors affect the Bayesian uncertainties, future work should address how we can use this synthetic framework to evaluate the reliability of other uncertainty estimation methods.

Chapter 4

Model Calibration

Until this point, we have described methods to extract uncertainty estimates from deep models. However, how we can verify that the predictive uncertainty estimates are reliable or not has not yet been mentioned. Calibration is a concept that enables us to address this question.

In this chapter, we will discuss the concept of uncertainty calibration in classification and regression. We give a miscalibration example of DNNs and describe the calibration method. Finally, we demonstrate how these methods mitigate the miscalibration issue through the object localization task. Our case study shows that BNNs can be miscalibrated.

4.1 Motivation

In safety-critical systems such as self-driving cars, it is desirable to have an object detection system that provides accurate predictions and reliable, or well-calibrated, associated uncertainties. The uncertainties in this case come from two sources: the bounding box regressor and the object classifier. For the bounding box regressor, a $p\%$ confidence interval for each coordinate (estimated from the calibrated uncertainties) should contain the true value $p\%$ of the time. Similarly, in classification, calibration means that predictions with $p\%$ confidence are accurate $p\%$ of the time. Miscalibration, in either case, can lead to dangerous situations in autonomous driving. For instance, if the detection model indicates the 95% confidence interval that the location of a pedestrian is within the sidewalk, but it is actually a 50% confidence interval, then the vehicle may make hazardous movements. Model-recalibration, therefore, is one of the important procedure to make sure that our predictive uncertainty estimate is reliable.

4.2 Definitions and Methods

In this section, we describe the definition of model calibration and the recalibration methods, which are introduced in the work of C. Guo, et.al [30] (for classification) and V. Kuleshov, et.al [51] (for regression). Besides, we also suggest one minor improvement for the regression calibration algorithm suggested by V. Kuleshov.

4.2.1 Classification

In a classification problem, we are given the dataset $D = \{X, Y\}$ where $x_i \in X$ is the input and $y_i \in Y$ is the corresponding label (within K classes). As mentioned in Chapter 2, for an arbitary input x_i, a deep classification model (Bayesian or non-Bayesian) $f(x)$ trained on D gives a predictive probability vector $P_i = \{p_{i1}, p_{i2}, ..., p_{iK}\}$ that indicates how likely the input x_i belongs to a certain class. This probability vector can then be used to express the level of uncertainty through the entropy metric. This means that to evaluate the uncertainty estimates, instead of working with the entropy metric, we can work directly with the probability vector.

Calibration In the context of classification, calibration means that the expected confidence probability should match the corresponding actual accuracy. That is:

$$\mathbb{P}(y_i = k|p_{ik}) = p_{ik} \text{ for every class } k \tag{4.1}$$

This condition, however, is hard to achieve in the case of muti-class classification. A more relaxed and practical condition which we will follow is:

$$\mathbb{P}(y_i = k|p_{ik}) = p_{ik} \text{ where } k = \arg\max_m p_{im} \tag{4.2}$$

This means that we only consider the calibration for the predicted class probability.

Calibration Methods Guo, et al. [30] suggest that using temperature scaling is the simplest way to calibrate a model. Assume that the model has a softmax activation at the output layer, temperature scaling is a procedure that divides the logits vector by a hyper-parameter T. Denote the logit value for class k as σ_k. After temperature scaling, the predictive probability for class k is:

$$\mathbb{P}(y_i = k|p_{ik}) = \frac{\exp^{\sigma_k/T}}{\sum_j \exp^{\sigma_j/T}} \tag{4.3}$$

The hyper-parameter T should be chosen such that it minimizes the cross entropy loss on the calibration set (similar to the validation set but we use this one for calibration only). We note that using this method *does not affect the model's accuracy* since temperature scaling does not change the ranking of output probabilities.

Reliability Diagram A reliability diagram (Figure 4.1) shows the relationship between the confidence (x-axis) and accuracy (the y-axis). To plot the reliability diagram, we collect the confidence measures in the test set and partition the measures into M bins. We then calculate the accuracy for each bin and plot as a bar as shown. The model is calibrated when the bars follow the diagonal line (the blue line in Figure 4.1).

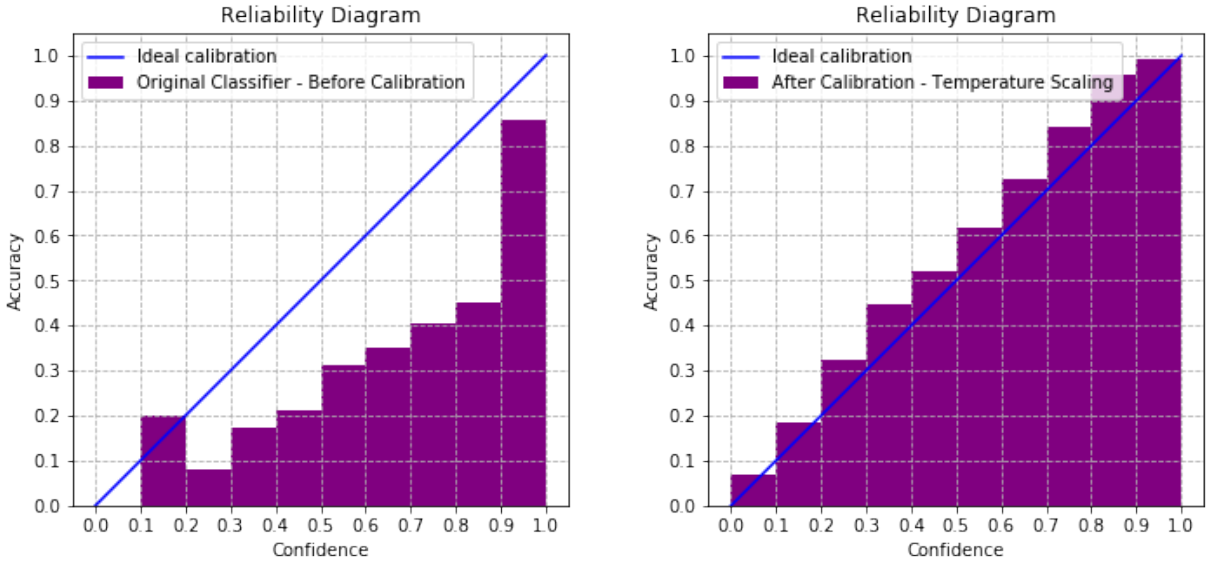

Figure 4.1: Model calibration example: reliability diagram before (left figure) and after calibration (right figure).

We illustrate this through a small classification experiment using the CIFAR-10 dataset. We train a VGG-16 classification network for this task. Figure 4.1 shows the reliability diagram on test set before (left figure) and after calibration with temperature scaling (right figure). As we can see, without calibration, the probability given by the model does not

reflect the actual outcome. For instance, when the model predicts a class with a confidence probability 0.8, the actual accuracy is only 0.4. This is a very misleading probability and should not be relied on for any safety-critical task. On the other hand, temperature scaling makes the confidence level align well with the actual accuracy and thus makes the model reliable.

4.2.2 Regression

In the regression problem, we are given the dataset $D = \{X, Y\}$ where $x_i \in X$ is the input and $y_i \in Y$ is the corresponding label that takes a continuous value. Similar to classification, for input x_i, our model gives the probability distribution, which we assume to be Gaussian, $p(y_i|x_i) = \mathcal{N}(\hat{\mu}_i, \hat{\sigma}_i^2|x_i)$. We denote the associated cumulative distribution function $P_{y_i|x_i}(z) = \Phi_{x_i}(z|\hat{\mu}_i, \hat{\sigma}_i^2)$. For real value $z \in \mathbb{R}$, $P_{y_i|x_i}(z)$ is the probability that the label y is in the $(-\infty, z]$ interval. Conversely, for a probability value q, we obtain the output z of the inverse CDF: $P^{-1}_{y_i|x_i}(q) = z$, which means that the interval $(-\infty, z]$ is a $100q\%$ interval for y_i.

Calibration Let $\mathbb{I}_{y_i|x_i}(q) := \mathbb{I}[y_i \leq P^{-1}_{y_i|x_i}(q)]$ be an indicator function that verifies the condition in the brackets. Then $P_{y_i|x_i}(z)$ is calibrated when:

$$\mathbb{E}[\mathbb{I}_{y_i|x_i}(q)] = q \tag{4.4}$$

This implies that we expect to see the $100q\%$ confidence interval to cover $100q\%$ of the label data. Calibrating a regression model means that we adjust $P_{y_i|x_i}(z)$ such that (4.4) holds. To obtain a reliable uncertainty estimate, we carry out two steps: validating the uncertainty estimate (our suggestion) and calibrating it (based on [51]).

Validating the Uncertainty Estimates In regression, higher estimated variance should correspond to higher expected square error. However, in practice, if the model lacks expressiveness or does not converge, the resulting uncertainty estimates may not be valid and the calibration process will not give desired results. Thus, we suggest using scatter plot (representing the variance $\hat{\sigma}_i^2$ and square error $(y_i - \hat{y}_i)^2$) as a visualization method to validate this attribute of the uncertainty estimate before calibrating it.

Calibration Method The main idea of the calibration method proposed by Kuleshov, et al. [51] is as follows: suppose that our model is mis-calibrated and gives $\mathbb{E}[\mathbb{I}_{y_i|x_i}(q)] = r_q$,

where $r_q \neq q$. This means that the $100q\%$ confidence interval covers $100r_q\%$ of actual data. In other words, the $100q\%$ interval is actually a $100r_q\%$ interval. So when the model gives us a CDF $P_{y_i|x_i}(z)$ and we query the $100q\%$ interval based on this CDF, the interval is actually a $100r_q\%$ interval.

Based on the above idea and supposing that we have a calibration set $C = \{X, Y\}$), the calibration process is as follows:

1. Iterating over all data pairs (x_i, y_i), we build a list of features $U = \{P_{y_i|x_i}(y_i)\}_{i=1}^{N}$. We then build a list of corresponding labels $V = \{|[y_t|P_{y_t|x_t}(y_t) < P_{y_i|x_i}(y_i)]_{t=1}^{N}|/N\}_{i=1}^{N}$ ($|\,.\,|$ is a cardinality of a set). Each value V_i shows how many actual labels are actually contained within the interval defined by U_i.

2. We train a regression model R on U,V, using isotonic regression.

3. At test time, the probability for the confidence interval given by $R \circ P_{y|x}(z)$ is calibrated.

Reliability Diagram In regression, the reliability diagram shows the relation between the expected confidence interval and the actual confidence interval. Similar to classification, perfect calibration corresponds to the identity line.

We illustrate in Figure 4.2 how calibration works in the case of regression. The original function is a sine function with an additive heteroscedastic Laplacian noise. We fit this data using a neural network with a Gaussian loss function described in Section 2.2. Since our assumption about the noise is incorrect (Gaussian versus Laplacian), the expected confidence interval does not match the actual confidence interval (as shown in the reliability diagram). By using the described calibration technique, we are able to adjust the confidence interval so that our expectation matches the reality.

4.3 Case study: Calibrating Uncertainties in Object Localization Task

In this section, we address the miscalibration issue for the 2D single object classification and localization (SOCL) task and demonstrate its applicability on the Oxford-IIIT Pet Dataset ([70]). Although recent works by [62] and [21] have shown the benefits of modeling uncertainty for detection accuracy in 2D open-set conditions and the 3D Lidar object

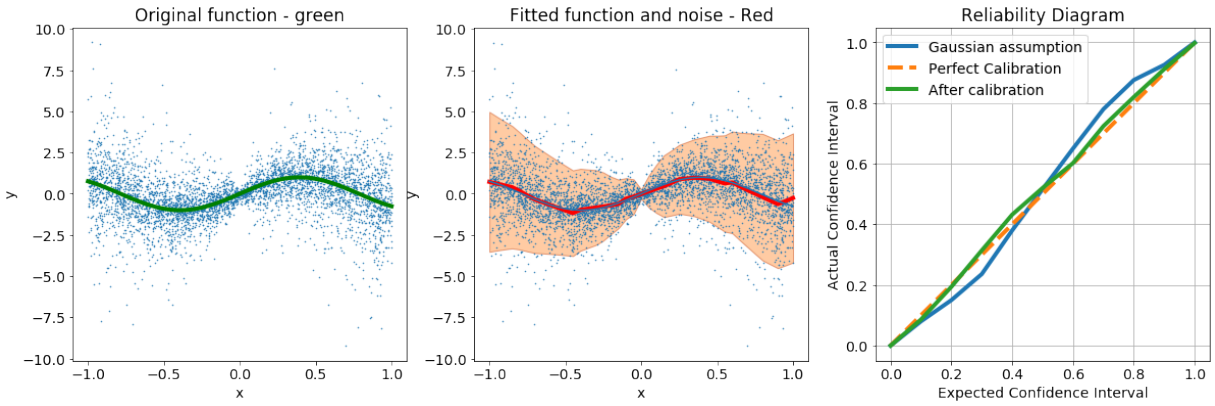

Figure 4.2: A simple example for calibration in heteroscedastic regression. The original function and data is shown in the leftmost figure. In the middle figure, we use heteroscedastic regression to fit the data, show the fitted function and the 95% expected confidence interval based on the Gaussian assumption. The rightmost shows the reliability diagram and how calibration fixes the problem.

detection task, respectively, neither of them has analyzed or focused on the reliability of the estimated localization uncertainties in terms of calibration. We show, experimentally, that the resulting calibrated model obtains more reliable uncertainty estimates.

We focus on the localization uncertainty estimates since our experiment shows that, while BNNs produce a calibrated classification uncertainty (similar to McClure and Kriegeskorte's results [61]), the estimated localization uncertainty is not calibrated. Specifically, our contributions are: (1) we show that the estimated localization uncertainties for the bounding box coordinates from the BNN model are not calibrated; (2) we adapt Kuleshov's et al method [51] for calibrating regression models and show improvements in this setting.

The remainder of this section is structured as follows. Subsection 4.3.1 gives the necessary background for the SOCL task. Subsection 4.3.2 describes the calibration procedure for localization. Subsection 4.3.3 shows experimental results demonstrating the method is effective.

4.3.1 Background: SOCL Task with Uncertainty Estimation

The goal of the SOCL task is to obtain a model that is able to predict the bounding box and class of an object in a given image. For an image dataset $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$, the associated labels are $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_N\}$, where each $\mathbf{y}_k = [\mathbf{c}_k, \mathbf{b}_k]$ consists of a one-hot

encoded class $\mathbf{c}_k$ and the bounding box coordinates $\mathbf{b}_k = \{b_{1k}, b_{2k}, b_{3k}, b_{4k}\}$. We define a DNN $\mathbf{f_W}(\mathbf{x}) = \hat{\mathbf{y}}$ with weights $\mathbf{W}$ such that it predicts both the class probability and coordinates.

Incorporating Uncertainty Based on Kendall and Gal [40]'s method, we incorporate the aleatoric and epistemic uncertainties into the model by optimizing the weights $\mathbf{W}$ with heteroscedastic loss and dropout training ([78]). At test time, we sample the predictions with MC-dropout and calculate the predictive mean and uncertainties. This results in a BNN model $\bar{\mathbf{f}}_{\mathbf{W}}(\mathbf{x}) = [\bar{\mathbf{y}}, \bar{\boldsymbol{\sigma}}^2]$, where $\bar{\mathbf{y}} = [\bar{\mathbf{c}}, \bar{\mathbf{b}}]$ is the predictive mean ($\bar{\mathbf{c}}$ is a predicted class probability) and $\bar{\boldsymbol{\sigma}}^2 = \{\bar{\sigma}_1^2, \bar{\sigma}_2^2, \bar{\sigma}_3^2, \bar{\sigma}_4^2\}$ is the predictive variance (sum of the epistemic and aleatoric variance) of the coordinates, assuming that they are mutually independent [1]. Similarly to [52], we estimate the probability $p(b_i|\mathbf{x})$ as a Gaussian: $p(b_i|\mathbf{x}) = \mathcal{N}(\bar{b}_i, \bar{\sigma}_i^2)$, for $i \in 1, 2, 3, 4$.

4.3.2 Calibration for Bounding Box Regressor

For each $p(b_i|\mathbf{x})$, similar to Section 4.3.1, we define its CDF and inverse CDF as $P_{b_i|\mathbf{x}}(z)$ and $P^{-1}_{b_i|\mathbf{x}}(q)$ respectively.

Calibrating the SOCL BNN We adapt the regression calibration method in Section 4.2.2 for the case of bounding box estimation. Given an uncalibrated probabilistic model $\bar{\mathbf{f}}_{\mathbf{W}}(\mathbf{x})$ for the SOCL task with $P_{b_i|\mathbf{x}}(z) = \Phi(z|\bar{b}_i, \bar{\sigma}_i^2)$ for each coordinate and a calibration dataset $\hat{\mathbf{X}}$, $\hat{\mathbf{Y}}$, the calibration process trains a calibration model R_i whose input is $P_{b_i|\mathbf{x}}(z)$ such that $\hat{P}_{b_i|\mathbf{x}}(z) = R_i \circ P_{b_i|\mathbf{x}}(z) = R_i \circ \Phi(z|\bar{b}_i, \bar{\sigma}_i^2)$ is calibrated. After obtaining R_i, we replace $P_{b_i|\mathbf{x}}(z)$ by $\hat{P}_{b_i|\mathbf{x}}(z)$ for localization uncertainty estimation.

We use $[x_{min}, y_{min}, x_{max}, y_{max}]$ for encoding the coordinates in our experiments, where x_{min},y_{min} is the top left and x_{max}, y_{max} is the bottom right corner of the bounding box. To estimate the $100q\%$ confidence interval around the mean, we determine the upper bound and lower bound for each b_i by calculating $\hat{P}^{-1}_{b_i|\mathbf{x}}(r + q/2)$ and $\hat{P}^{-1}_{b_i|\mathbf{x}}(r - q/2)$ accordingly for each coordinate, where $r = \hat{P}_{b_i|\mathbf{x}}(\bar{b}_i)$. These bounds define a confidence interval as a region in which the bounding box can occur (see blue region in Figure 4.3d). Since this bound requires $r + q/2 < 1$ and $r - q/2 > 0$, one can instead use the median by setting $r = 0.5$.

[1] This assumption gives an overapproximation of the bounding box extents, which is sufficient for obstacle avoidance.

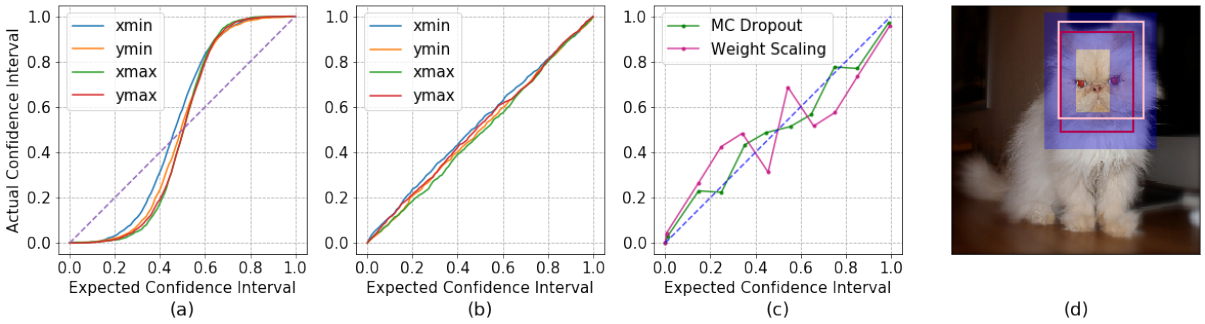

Figure 4.3: Reliability diagram and an example. Figure 4.3(a): reliability diagram for localization uncertainties before calibration. Figure 4.3(b): reliability diagram for localization uncertainties after calibration. Figure 4.3(c): reliability diagram for classification with MC-dropout and weight scaling. Figure 4.3(d) shows an example of bounding box localization with calibrated 95% confidence interval (blue region) centered around the mean (red). The ground truth is colored in pink

4.3.3 Experiments

In the following setting, we experimentally show the miscalibration problem of localization uncertainty of our model and the improvement after applying the calibration method on the model's output. For the completeness of the task, we also show the result for classification.

We use the Oxford-IIIT Pet Dataset ([70]), which consists of 3,686 annotated images in 2 classes depicting cats and dogs, for this task. The bounding box for localization covers the face of the pet. The dataset is split into 2:0.9:0.9 ratio for training, validation/calibration and testing respectively. For the model, the VGG-16 architecture ([76]) is used as a base network. The trained model obtained 94.85% classification accuracy and 14.03% localization error with 0.5 IOU threshold, based on the Imagenet evaluation method ([13]).We validated the uncertainty estimates (epistemic and aleatoric) and fit the calibration model R_i for each coordinate as described in Section 3.

Results Figure 4.3a -4.3c show the reliability diagrams for localization and classification uncertainties on the test set. Each reliability diagram shows the mapping between the expected confidence interval (from the model) and the actual one (i.e., how many labels are actually within that interval). Perfect calibration corresponds to the diagonal line in the diagram. Quantitatively, the calibration quality is evaluated by using the mean squared error (MSE) between the diagonal line and the calibration curve (calculated over

all buckets). Models with lower MSE are better calibrated.

Figure 4.3a and 4.3b show the reliability diagram for bounding box coordinates before and after calibration respectively. Consider the curve for x_{max} in Figure 4.3a, we can see that the expected 40% confidence interval corresponds to the 20% actual interval. In this case, the model has underestimated the uncertainty. On the other hand, the expected 60% confidence interval corresponds to the 80% actual interval, which implies that the model has overestimated the uncertainty in this range. After calibration, the estimated uncertainties are reliable, e.g, the expected 20% confidence interval contains approximately 20% of the true outcome. The calibration process reduces the average MSE from 2.7E-02 to 2.7E-04 for the four coordinates.

We also observed that, in Figure 4.3c, MC-dropout produces a calibrated classification confidence with MSE of 3.0E-03, compared to that of 1.6E-02 with the original weight scaling method for network trained with dropout ([78]), in which we scale the weights according to the dropout rate (instead of randomly setting the weight values to 0 like MC-dropout) at test time, and show the resulting softmax probabilities.

4.4 Calibration and the need for Bayesian Modelling

At this point, one may ask a valid question: given the existence of calibration technique, does Bayesian modelling matter since what we can do is to train a non-Bayesian uncertainty estimation model and calibrate it to fit this purpose?

We argue that Bayesian modelling is still relevant and should be used in parallel to the calibration process. This is because, in the calibration process, we use only the inlier data to calibrate the model. This means that the reliability curve only holds for the inlier data. For the outlier data, such as the images with weather effects in Chapter 4, the calibration property does not necessarily hold. We suggest the following procedure that combines Bayesian modelling and calibration:

- Train a BNN and choose a threshold ϵ value for the epistemic uncertainty. If the epistemic estimates (mutual information) is lower than ϵ, then we decide that input is an inlier data.
- Collect all the inlier data and calibrate the model based on the predictive probability vector of these data.

- At test time, we use the calibrated probability values for the inlier input to measure the potential risk and make decisions accordingly. For data that has high epistemic estimate, we should rely on other treatments.

4.5 Conclusion

In this chapter, we consider the reliability of estimated localization uncertainty for the 2D SOCL task in terms of calibration. Our experiment shows that without calibration, the estimated localization uncertainties are misleading. We adapted an existing method for calibrating regression to the uncertainty estimates of bounding box coordinates. The result shows that the new uncertainty estimates are well-calibrated.

In future work, we would like to extend this work to the general 2D and 3D multiple object detection task and use a more complex dataset such as KITTI ([26]). Furthermore, we want to address the calibration problem in the case the coordinates are not mutually independent.

Chapter 5

Real-Time Uncertainty Estimation with Dirichlet Distillation

In the field of autonomous driving, it is desirable to have a BNN-based computer vision system functioning during real-world operation. However, as we have seen so far, BNNs require multiple samples (around 50 to 100) at test time in order to estimate the uncertainty. This prevents the use of any complicated BNN in real-time applications. In this chapter, we introduce a distillation technique, focusing on the case of classification, to handle this issue. This technique allows us to estimate the uncertainty (both epistemic and aleatoric) without the need for dropout sampling. The general idea is to train a student network that learns from the BNN's output to estimate the uncertainty. Our experimental result shows that the student network are able to generalize and provide good epistemic estimates for both in-distribution and out of distribution data.

5.1 Method

Problem Formulation Let us suppose that we are given the dataset $D = \{X, Y\}$ where $x_i \in X$ is the input, $y_i \in Y$ is the corresponding label (within K classes) and a BNN model $\mathcal{M}(x)$ is "trained" on D. We also assume that model $\mathcal{M}$ is very complex such that the closed-form solution for the posterior is not achievable and Monte Carlo sampling is required to estimate this posterior. Specifically, given an input x_t at test time, model $\mathcal{M}$, through the sampling process, outputs multiple categorical distribution samples $p_{t1}, p_{t2}, ..., p_{tT}$. As shown before in Chapter 2, from these samples, we can estimate the

corresponding epistemic and aleatoric uncertainty. Due to the real-time constraint, we would like to obtain these uncertainty estimates without the sampling process.

Approach In a distillation technique, we want to train a model $\mathcal{S}(x)$ such that from the output $\mathcal{S}(x_t)$, we are able to derive the class prediction, aleatoric and epistemic uncertainty estimates that are similar to those of $\mathcal{M}(x_t)$. We sketch the insight for the method as follows. For each input x_t, model $\mathcal{M}$ provides a distribution over the class probability vector. We denote this distribution as $\mathcal{P}_{\mathcal{M}}(p_t|x_t)$, from which we are able to estimate the uncertainties. The closed form of this distribution is unknown to us, and we can only sample from this distribution (i.e. MC Dropout). The core idea is that we train the model $\mathcal{S}(x)$ to output a closed form distribution $\mathcal{P}_{\mathcal{S}}(p_t|x_t)$ such that the two distributions are close to each other by minimizing the KL divergence between the two distributions. Since the two distributions must share the same support, in other words, they must both output categorical distribution samples, we use the Dirichlet distribution to represent $\mathcal{P}_{\mathcal{S}}(p_t|x_t)$. Since the number of parameters of the Dirichlet distribution is the same as the number of classes and all of them are positive, we can use the network architecture of $\mathcal{M}$ for $\mathcal{S}$ and adding the softplus activation at the logit layer (since we are trying to fit positive numbers) to output these parameters.

5.1.1 Loss Function

Let ω be the training parameters for the model $\mathcal{S}$. To train model $\mathcal{S}$, for each input data x, we would like to minimize the KL divergence between the two distributions $\mathcal{P}_{\mathcal{M}}(p|x)$ and $\mathcal{P}_{\mathcal{S}}(p|x,\omega)$.

$$\begin{aligned}
\mathbb{D}_{\mathrm{KL}}(\mathcal{P}_{\mathcal{M}}(p|x)||\mathcal{P}_{\mathcal{S}}(p|x,\omega)) &= \int \mathcal{P}_{\mathcal{M}}(p|x)\log\frac{\mathcal{P}_{\mathcal{M}}(p|x)}{\mathcal{P}_{\mathcal{S}}(p|x,\omega)}dp \\
&= \underbrace{\int \mathcal{P}_{\mathcal{M}}(p|x)\log\mathcal{P}_{\mathcal{M}}(p|x)dp}_{\text{constant}} - \int \mathcal{P}_{\mathcal{M}}(p|x)\log\mathcal{P}_{\mathcal{S}}(p|x,\omega)dp \\
&= \qquad\qquad\text{constant}\qquad\qquad - \mathbb{E}_{\mathcal{P}_{\mathcal{M}}(p|x)}[\log\mathcal{P}_{\mathcal{S}}(p|x,\omega)]
\end{aligned} \tag{5.1}$$

Although we do not know the closed-form of $\mathcal{P}_{\mathcal{M}}(p|x)$, we can collect samples from it (i.e., by using MC-Dropout), and thus we can approximate the expectation term in the

Equation 5.1 using Monte-Carlo sampling:

$$\mathbb{D}_{\mathrm{KL}}(\mathcal{P}_{\mathcal{M}}(p|x)||\mathcal{P}_{\mathcal{S}}(p|x,\omega)) \approx \text{constant} - \frac{1}{N}\sum_{n=1}^{N}\log \mathcal{P}_{\mathcal{S}}(p_n|x,\omega) \quad \text{where } p_n \sim \mathcal{P}_{\mathcal{M}}(p|x) \tag{5.2}$$

Optimizing the KL divergence is equivalent to optimizing the expectation term. We define our loss function for each data point $L(x,\omega)$ as follow:

$$L(x,\omega) = -\frac{1}{N}\sum_{n=1}^{N}\log \mathcal{P}_{\mathcal{S}}(p_n|x,\omega) \text{ where } p_n \sim \mathcal{P}_{\mathcal{M}}(p|x) \tag{5.3}$$

By minimizing this loss function, we obtain a student model $\mathcal{S}$ that is able to approximate the predictive probability of a teacher model $\mathcal{M}$.

5.1.2 Extracting the uncertainties

We now describe the method to extract different types of uncertainties from the student network $\mathcal{S}$. Given an input x, the student network outputs a Dirichlet distribution $\mathcal{P}_{\mathcal{S}}(p|x,\omega) = \mathrm{Dir}(\alpha_1,...,\alpha_K|x)$.

Epistemic Uncertainty For the teacher network $\mathcal{M}$, we can use the mutual information metric to capture the epistemic uncertainty. Another way to capture the epistemic uncertainty is by calculating the variance of the probability samples. L. Smith and Y. Gal [77] proved that the latter one is an approximation of the mutual information metric. For each p_i, the variance is:

$$\mathrm{Var}[p_i] = \frac{z_k(1-z_k)}{Z} \tag{5.4}$$

where:

$$Z = 1 + \sum_k \alpha_k \text{ and } z_k = \frac{\alpha_k}{\sum_k \alpha_k}$$

Then, the epistemic uncertainty from the student network $\mathcal{S}$ for an input x can be estimated as follow:

$$\mathbb{MI}(x) \approx \frac{1}{K \times Z}\sum_{k=1}^{K} z_k(1-z_k) \tag{5.5}$$

Predictive Uncertainty The predictive entropy can be estimated by using the the mean probability vector of the distribution.

$$\mathbb{H}(x) = -\frac{1}{K}\sum_{k=1}^{K} z_k \log(z_k) \tag{5.6}$$

Aleatoric Uncertainty Finally, the aleatoric uncertainty can be estimated by:

$$\mathbb{AE}(x) = \mathbb{H}(x) - \mathbb{MI}(x) \tag{5.7}$$

5.2 Experiment

In this experiment, we show and compare the student network's uncertainty estimation to those of the teacher network. We use the CIFAR-10 dataset [49], which contains 60,000 32x32 RGB images in 10 classes, for training. We implement the VGG-16 architecture [76] for both student and teacher networks (with dropout layers). For testing, we use the CIFAR-10 and CIFAR-100 test dataset. The purpose of the CIFAR-100 dataset at testing time is to compare the estimated uncertainty for data came from out-of-distribution.

In Figure.5.1, we compare the difference between two network's uncertainty estimates on two different scenarios: in-distribution data (CIFAR-10 test data) and out-of-distribution data (CIFAR-100). Column-wise, we observe that the distribution between the student and teacher network for CIFAR-100 is flatter than the ones for CIFAR-10. Since the CIFAR-100 dataset can be considered as an outlier data, the estimated epistemic uncertainty should be, on average, higher than that for CIFAR-10 dataset. We also see that for both datasets, the student network tends to give lower epistemic estimation than the teacher network. For the aleatoric and predictive uncertainty, however, the teacher network's estimations are lower than the student's.

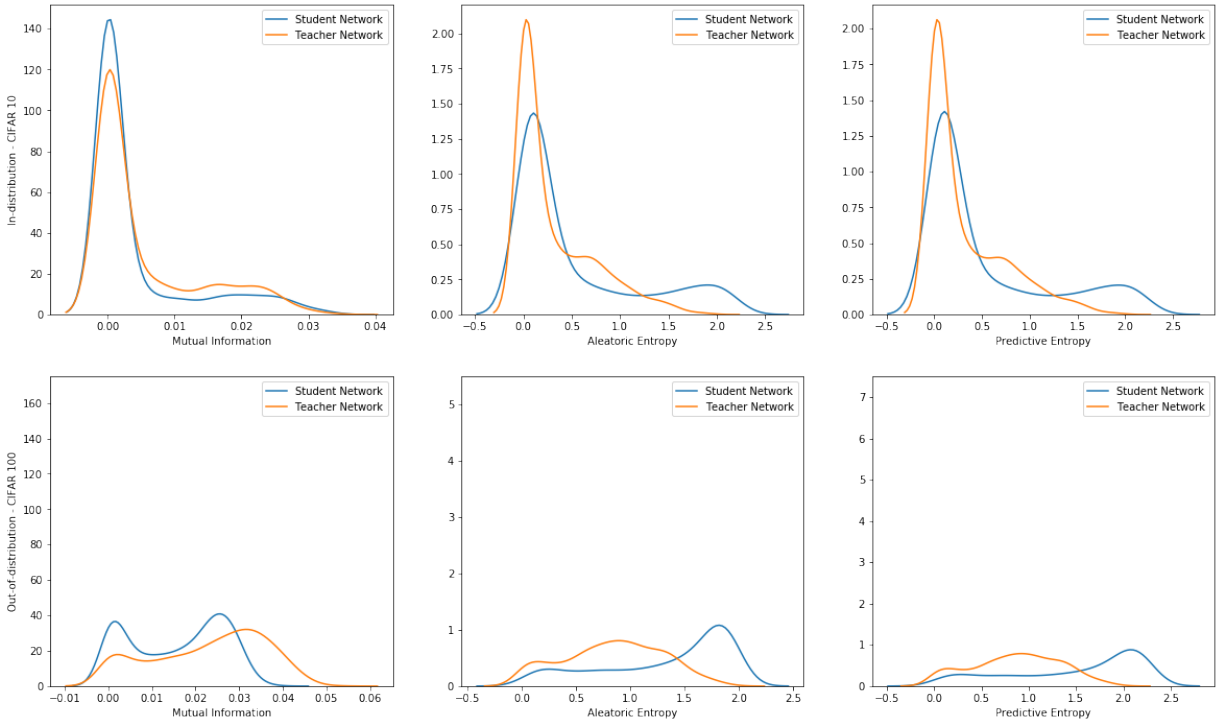

Figure 5.1: Comparison in uncertainty estimation between the student and teacher networks. The first row shows the estimations in CIFAR-10 dataset and the second row shows the estimations in CIFAR-100 dataset. For completeness, we also compute the mean-square error (MSE) for the estimations by the two networks (data point by data point basis). For CIFAR-10, the MSE for mutual information, aleatoric and predictive uncertainty estimates are 4.02e-05, 2.09e-01, 2.11e-0.1 respectively. Similarly, for CIFAR-100, the values are: 2.12e-04, 7.01e-01, 7.08e-01.

Chapter 6

Conclusion

In this thesis, we review the concept of Bayesian neural networks and its usefulness in uncertainty modeling. We study the behavior of the uncertainty estimates from BNN in the context of road-scene image segmentation and introduce several improvements in terms of calibration and model distillation. In particular:

- We use the ProcSy synthetic data to investigate the effects of several selected factors on BNN's uncertainty in the task of image semantic segmentation.
- We show that the estimated localization uncertainties by BNN are not calibrated and how calibration technique can mitigate this problem.
- We develop a distillation technique based on Dirichlet distribution that allows us to estimate the epistemic uncertainty in real-time.

In the future work, we would like to extend the distillation method and apply it to the image segmentation task. Also, our current experiments do not include any time-series problem in computer vision such as tracking. We believe that applying uncertainty estimation in this task is an interesting problem that should be studied in the future.

References

[1] Tesla driver killed after smashing into truck had just enabled autopilot. 2019.

[2] Aleksandr Y Aravkin, James V Burke, and Gianluigi Pillonetto. Sparse/robust estimation and kalman smoothing with nonsmooth log-concave densities: Modeling, computation, and theory. *The Journal of Machine Learning Research*, 14(1):2689–2728, 2013.

[3] Anoop Korattikara Balan, Vivek Rathod, Kevin P Murphy, and Max Welling. Bayesian dark knowledge. In *Advances in Neural Information Processing Systems*, pages 3438–3446, 2015.

[4] David Barber and Christopher M Bishop. Ensemble learning in bayesian neural networks. *Nato ASI Series F Computer and Systems Sciences*, 168:215–238, 1998.

[5] Christopher M Bishop. Novelty detection and neural network validation. *IEE Proceedings-Vision, Image and Signal processing*, 141(4):217–222, 1994.

[6] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622, 2015.

[7] Nilotpal Chakravarti. Isotonic median regression: a linear programming approach. *Mathematics of operations research*, 14(2):303–308, 1989.

[8] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018.

[9] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image seg-

mentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 801–818, 2018.

[10] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[11] Yufei Cui, Wuguannan Yao, Qiao Li, Antoni B Chan, and Chun Jason Xue. Accelerating monte carlo bayesian inference via approximating predictive uncertainty over simplex. *arXiv preprint arXiv:1905.12194*, 2019.

[12] Krzysztof Czarnecki and Rick Salay. Towards a framework to manage perceptual uncertainty for safe automated driving. In *SAFECOMP 2018 Workshops, WAISE, Proceedings*, pages 439–445. Springer, 2018.

[13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[14] John Denker and Daniel Schwartz. Large automatic learning, rule extraction, and generalization. *Complex systems*, 1(5):877–922, 1987.

[15] S Depeweg, JM Hernandez-Lobato, F Doshi-Velez, and S Udluft. Decomposition of uncertainty in bayesian deep learning for efficient and risk-sensitive learning. In *35th International Conference on Machine Learning, ICML 2018*, volume 3, pages 1920–1934, 2018.

[16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[17] Terrance DeVries and Graham W Taylor. Learning confidence for out-of-distribution detection in neural networks. *arXiv preprint arXiv:1802.04865*, 2018.

[18] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[19] Yuming Fang, Zhou Wang, Weisi Lin, and Zhijun Fang. Video saliency incorporating spatiotemporal cues and uncertainty weighting. *IEEE transactions on image processing*, 23(9):3910–3921, 2014.

[20] Alhussein Fawzi, Hamza Fawzi, and Omar Fawzi. Adversarial vulnerability for any classifier. In *Advances in Neural Information Processing Systems*, pages 1178–1187, 2018.

[21] Di Feng, Lars Rosenbaum, and Klaus Dietmayer. Towards safe autonomous driving: Capture uncertainty in the deep neural network for lidar 3d vehicle detection. *arXiv preprint arXiv:1804.05132*, 2018.

[22] Yarin Gal. Uncertainty in deep learning. 2016.

[23] 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.

[24] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1183–1192. JMLR. org, 2017.

[25] Pranav Ganti and Steven L Waslander. Visual slam with network uncertainty informed feature selection. *arXiv preprint arXiv:1811.11946*, 2018.

[26] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[27] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*, 2018.

[28] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[29] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1994.

[30] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330, 2017.

[31] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *International Conference on Learning Representations (ICLR)*, 2016.

[32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[33] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[34] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019.

[35] Peter J Huber. Robust statistics. In *International Encyclopedia of Statistical Science*, pages 1248–1251. Springer, 2011.

[36] Edwin T Jaynes and Oscar Kempthorne. Confidence intervals vs bayesian intervals. In *Foundations of probability theory, statistical inference, and statistical theories of science*, pages 175–257. Springer, 1976.

[37] Rhett Jones. Report: Uber's self-driving car sensors ignored cyclist in fatal accident. *Gizmodo*, 2018.

[38] Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *arXiv preprint arXiv:1511.02680*, 2015.

[39] 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.

[40] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5574–5584. Curran Associates, Inc., 2017.

[41] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[42] Mohammad Khan, Didrik Nielsen, Voot Tangkaratt, Wu Lin, Yarin Gal, and Akash Srivastava. Fast and scalable Bayesian deep learning by weight-perturbation in Adam. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2611–2620, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[43] Mohammad Emtiyaz Khan, Didrik Nielsen, Voot Tangkaratt, Wu Lin, Yarin Gal, and Akash Srivastava. Fast and scalable bayesian deep learning by weight-perturbation in adam. In *International Conference on Machine Learning*, pages 2616–2625, 2018.

[44] Samin Khan, Buu Phan, Rick Salay, and Krzysztof Czarnecki. Procsy: Procedural synthetic dataset generation towards influence factor studies of semantic segmentation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019. To appear.

[45] Donald Knuth. *The TEXbook*. Addison-Wesley, Reading, Massachusetts, 1986.

[46] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[47] Adam Davis Kraft. *Vision by alignment*. PhD thesis, Massachusetts Institute of Technology, 2018.

[48] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

[49] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[50] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[51] Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression. In *Proceedings of the 35th International Conference on Machine Learning*, pages 2796–2804, 2018.

[52] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413, 2017.

[53] Leslie Lamport. *LaTeX — A Document Preparation System*. Addison-Wesley, Reading, Massachusetts, second edition, 1994.

[54] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[55] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[56] Yun-Fu Liu, Da-Wei Jaw, Shih-Chia Huang, and Jenq-Neng Hwang. Desnownet: Context-aware deep network for snow removal. *IEEE Transactions on Image Processing*, 27(6):3064–3073, 2018.

[57] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[58] David JC MacKay. *Bayesian methods for adaptive models*. PhD thesis, California Institute of Technology, 1992.

[59] Radek Mackowiak, Philip Lenz, Omair Ghori, Ferran Diego, Oliver Lange, and Carsten Rother. Cereals-cost-effective region-based active learning for semantic segmentation. *arXiv preprint arXiv:1810.09726*, 2018.

[60] Andrey Malinin, Bruno Mlodozeniec, and Mark Gales. Ensemble distribution distillation. *arXiv preprint arXiv:1905.00076*, 2019.

[61] Patrick McClure and Nikolaus Kriegeskorte. Representing inferential uncertainty in deep neural networks through sampling. *arXiv preprint arXiv:1611.01639*, 2016.

[62] Dimity Miller, Lachlan Nicholson, Feras Dayoub, and Niko Sünderhauf. Dropout sampling for robust object detection in open-set conditions. *arXiv preprint arXiv:1710.06677*, 2017.

[63] Dimity Miller, Lachlan Nicholson, Feras Dayoub, and Niko Sünderhauf. Dropout sampling for robust object detection in open-set conditions. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–7. IEEE, 2018.

[64] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[65] Jishnu Mukhoti and Yarin Gal. Evaluating bayesian deep learning methods for semantic segmentation. *arXiv preprint arXiv:1811.12709*, 2018.

[66] Kevin P Murphy. *Machine learning: a probabilistic perspective*. 2012.

[67] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[68] David A Nix and Andreas S Weigend. Estimating the mean and variance of the target probability distribution. In *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference On*, volume 1, pages 55–60. IEEE, 1994.

[69] Georgios Papadopoulos, Peter J Edwards, and Alan F Murray. Confidence estimation methods for neural networks: A practical comparison. *IEEE transactions on neural networks*, 12(6):1278–1287, 2001.

[70] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[71] Buu Phan, Samin Khan, Krzysztof Czarnecki, and Rick Salay. Bayesian uncertainty quantification with synthetic data. In *International Conference on Computer Safety, Reliability, and Security*. Springer (to be appeared), 2019.

[72] Buu Phan, Rick Salay, Krzysztof Czarnecki, Vahdat Abdelzad, Taylor Denouden, and Sachin Vernekar. Calibrating uncertainties in object localization task. *The Third Bayesian Deep Learning Workshop, NeuRIPS 2019*, 2018.

[73] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[74] David Heinz Schumann et al. Robust variable selection. 2009.

[75] Murat Sensoy, Lance Kaplan, and Melih Kandemir. Evidential deep learning to quantify classification uncertainty. In *Advances in Neural Information Processing Systems*, pages 3179–3189, 2018.

[76] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[77] Lewis Smith and Yarin Gal. Understanding measures of uncertainty for adversarial example detection. *UAI*, 2018.

[78] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[79] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[80] The Tesla Team. A tragic loss. 2016.

[81] Mattias Teye, Hossein Azizpour, and Kevin Smith. Bayesian uncertainty estimation for batch normalized deep networks. In *International Conference on Machine Learning*, pages 4914–4923, 2018.

[82] Naftali Tishby, Esther Levin, and Sara A Solla. Consistent inference of probabilities in layered networks: Predictions and generalization.

[83] Frederick Tung, Jianhui Chen, Lili Meng, and James J Little. The raincouver scene parsing benchmark for self-driving in adverse weather and at night. *IEEE Robotics and Automation Letters*, 2(4):2188–2193, 2017.

[84] Jake VanderPlas. Frequentism and bayesianism: a python-driven primer. *arXiv preprint arXiv:1411.5018*, 2014.

[85] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688, 2011.

[86] Wenda Xu, Jia Pan, Junqing Wei, and John M Dolan. Motion planning under uncertainty for on-road autonomous driving. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 2507–2512. IEEE, 2014.

[87] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. BDD100K: A diverse driving video database with scalable annotation tooling. *CoRR*, abs/1805.04687, 2018.

[88] Wei Zhou, Julie Stephany Berrio, Stewart Worrall, and Eduardo Nebot. Automated evaluation of semantic segmentation robustness for autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 2019.