Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen
International Center for
Information Technology

R&D Project

Benchmarking Uncertainty Estimation Methods in Deep Learning for Regression

Aswinkumar Vijayananth

Submitted to Hochschule Bonn-Rhein-Sieg,
Department of Computer Science
in partial fullfilment of the requirements for the degree
of Master of Science in Autonomous Systems

Supervised by

Prof. Dr. Nico Hochgeschwender
Deebul Nair

November 2020

I, the undersigned below, declare that this work has not previously been submitted to this or any other university and that it is, unless otherwise stated, entirely my own work.

Date

Aswinkumar Vijayananth

Abstract

Your abstract

Acknowledgements

Thanks to ....

Contents

List of Tables

1

Introduction

1.1 Motivation

1.1.1 ...

1.1.2 ...

1.2 Challenges and Difficulties

1.2.1 ...

1.2.2 ...

1.2.3 ...

1.3 Problem Statement

1.3.1 ...

1.3.2 ...

1.3.3 ...

2

State of the Art

This chapter aims to explain the state-of-the-art uncertainty estimation methods considered for this research work.

2.1 Dropout as Bayesian Approximation

The method proposed by Gal *et al.* [[8]] estimates model uncertainty in Neural Net models using Dropout [[29]] which is a commonly used regularization technique. The work proves the equivalence between a Dropout applied Neural Network model and an approximated Deep Gaussian process model and establishes Dropout as a means to approximate the posterior predictive distribution. The absence of any need to make changes to the optimization process for integrating this method to a Neural Network model can be considered as one of its unique features. The technique applies to Neural Nets meant for both classification and regression tasks.

In order to obtain model uncertainty using this method in practice, dropout is enabled during the test-time and a given input is passed through the Neural Network a number of times equal to the pre-defined Monte-Carlo sample count hyper-parameter. The sample mean and variance of multiple stochastic forward passes correspond to the final model output and model variance respectively.

Though this method forms the basis of a few uncertainty estimation methods like [[23]], the need for multiple stochastic forward passes makes it computationally expensive and unsuited for real-time applications.

2.2 Deep Ensembles

Deep Ensembles [[20]] is a non-Bayesian approach to estimate predictive uncertainty by using outputs from an ensemble of Neural Networks. The work by LakshmiNarayanan *et al.* also proposes the idea of leveraging adversarial perturbations generated using the Fast Gradient SignMethod(FGSM)[[12]], to smoothen predictive distributions.

Authors show advantages of using Deep Ensembles over the base-line approach proposed by Gal *et al.* [[8]].The ability to report of high uncertainty estimates for Out-Of-Distribution and adversarial samples is claimed and proved by evaluating the method on a number of datasets for both regression and classification tasks. The work can be considered as one of the first frequentist approaches to estimate predictive uncertainty in Neural Networks.

With an increase in the number of member networks in an ensemble, the neural net model size increases as well. Therefore, the method is not suitable for very deep network architectures. Also, the need for diversity amongst ensemble members has a significant impact on the quality of estimated uncertainty.

2.3 Light-weight probabilistic deep networks

Lightweight Probabilistic Deep Networks [[9]] by Gast *et al.* enables conversion of a deterministic Neural Network to its probabilistic equivalent which is capable of propagating and outputting parameters of the predictive distribution, in turn uncertainty linked to it. The conversion is achieved in two steps:

- Introducing a probabilistic output layer that produces moments of the predictive distribution as its outputs.

- Replacing intermediate activations with their probabilistic equivalents. Assumed Density Filtering(ADF)(explained in 2.7.2) a form of Expectation Propagation is used to achieve it.

The need for minor architectural changes and no change in the optimization process is a key feature of this method. The method does not represent weights probabilistically, which results in over-confident predictions.

2.4 Aleatoric uncertainty as learned loss attenuation

Kendall *et al.* proposed the method[[14]] which aims to include aleatoric variance parameter as a part of the loss function to learn it from training data. The work presents a framework that combines Monte-Carlo Dropout[[8]] and the heteroscedastic loss function to estimate model and data uncertainties respectively. The idea of learning a mapping from input data to aleatoric uncertainty is an important contribution of this work. The method disregards any relationship between components of uncertainty as they are treated as independent entities, which does not hold true.

2.5 Prior Networks

Malinin *et al.*[[24]] proposed a technique to estimate predictive uncertainties in Neural Nets for classification, by parameterizing a prior distribution over the predictive distribution. The work uses Dirichlet distribution as the higher-order(prior) distribution over predictive categorical distributions, due to existence of conjugate prior relationship between the pair which makes the posterior analytically tractable. In this way, the variances(uncertainties) around parameters of the categorical distribution are modeled. Parameters of the higher-order distribution are included as a part of the loss function which gets optimized.

Prior Networks can be considered as one of the earliest works when it comes to learning how to model uncertainty from the given training data.Another important contribution of this work is that the method separates out the uncertainty that arises due to mismatch between training and test data distributions as "distributional uncertainty". Authors claim the work to outperform other uncertainty estimation methods when it comes to reporting misclassification and Out-Of-Distribution(OOD) input samples.

The proposed method is defined for classification setup and therefore cannot be used in regression nets. Also, the work lacks strong experimental evaluation as its assessed only on toy datasets with two other uncertainty estimation methods.

2.6 Choice of methods for this research work

The methods listed so far suffer from at least one of the following downsides: increased model inference time, increased model size, under/over estimation of uncertainties , disregard any relationship between components of uncertainty. This research work considers “A General Framework for Uncertainty Estimation in Deep Learning” [[23]] and “Deep Evidential Regression”[[1]] for benchmarking, based on claims made by their authors that they overcome deficits of other methods, as listed above. The rest of this chapter explains both the methods descriptively.

2.7 A General Framework for Uncertainty Estimation in Deep Learning

2.7.1 Overview

This work proposes a technique to distinctively estimate data and model uncertainties associated with an output of any Neural Network model.The technique is here after referred to as ”MCDO_ADF”, representing the fact that is a combination of two ideas, Monte-Carlo Drop Out(MCDO) and Assumed-Density-Filtering (ADF). MCDO_ADF treat the two uncertainty components to be related, which sets it apart from most of other uncertainty estimation methods that treat them to be independent. The method employs Bayesian Belief Networks combined with Monte-Carlo sampling for estimating the model uncertainty and relies on the idea of Assumed Density Filtering for estimating data uncertainty associated with an output.

Authors of the MCDO_ADF technique claim it to be a general framework to estimate uncertainties in Neural Networks. They give following reasons to validate their claim:

- Using this uncertainty estimation method does not require any architectural changes in the target Neural Network.
- Applicability of the method to Neural Network models of different tasks.
- Absence of any need of make changes in the optimization process.
- Ability of the technique to be applied to already trained models.

The upcoming sections of this chapter explain the MCDO_ADF technique and also analyze its claimed ”generality” by using it in a Resnet8 based Neural Network regression model meant for the application of steering-angle prediction in autonomous cars.(Note: A detailed description of the data set, training and inference procedures of the Neural Network model is available in 3).

2.7.2 Integrating MCDO_ADF with a Neural Network and estimating uncertainties

MCDO_ADF as an algorithm

Estimating uncertainty using the MCDO can be formulated as an algorithm consisting of the following steps:

- Transform the Neural Network of interest to its ADF(Assumed Density Filtering) version.
- Collect a predefined number(T) of Monte-Carlo(MC) samples by forwarding inputs and noise variances (x, v)) stochastically through the network for T times.
- Computation of output predictions and uncertainties

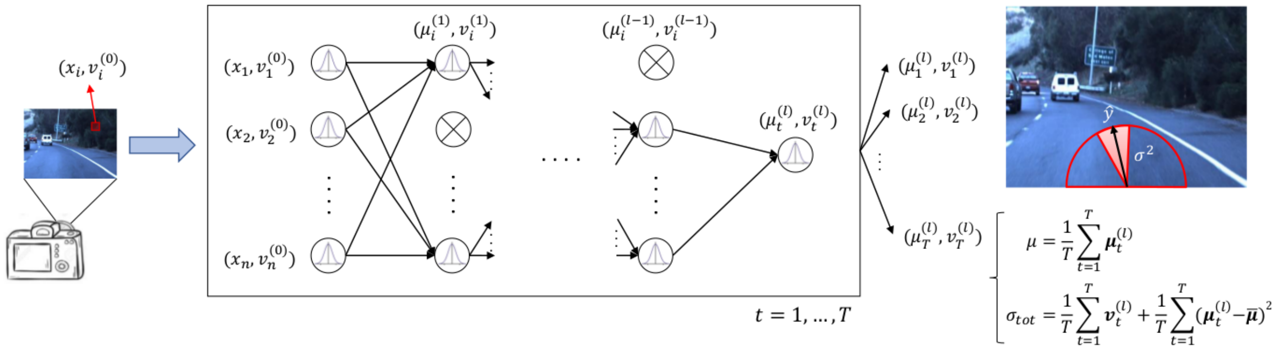

$(x_i, v_i^{(0)})$
$(x_1, v_1^{(0)})$
$(x_2, v_2^{(0)})$
$(x_n, v_n^{(0)})$
$(\mu_i^{(1)}, v_i^{(1)})$
$(\mu_i^{(l-1)}, v_i^{(l-1)})$
$(\mu_t^{(l)}, v_t^{(l)})$
$t = 1, \ldots, T$
$(\mu_1^{(l)}, v_1^{(l)})$
$(\mu_2^{(l)}, v_2^{(l)})$
$(\mu_T^{(l)}, v_T^{(l)})$
$\hat{y}$
σ^2
$\mu = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{\mu}_t^{(l)}$
$\sigma_{tot} = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{v}_t^{(l)} + \frac{1}{T}\sum_{t=1}^{T}(\boldsymbol{\mu}_t^{(l)} - \bar{\boldsymbol{\mu}})^2$

Figure 2.1: Illustration of the MCDO_ADF technique.Here x_i denotes the input through the i^{th} unit of a given hidden layer,$x_{i^{(n)}}$ denotes the noise variance input to the ith unit of the nth layer. Circles with crosses inscribed denote the dropped out neurons whereas the ones with the Gaussian distribution symbol denote the active units. T values of μ and v are collected from T stochastic forward passes. Image source:

Assumed Density Filtering (ADF)

The MCDO_ADF technique considers sensor noise to be the primary source of data uncertainty in neural network predictions and therefore feeds it to the Neural Network model during the inference. In order to propagate the input data distribution (parameterized by the input as its mean and sensor noise as variance) the technique of ADF is used. Briefly in the context of MCDO_ADF, ADF replaces every input activation into a probability distribution and also approximates the same using a tractable Gaussian distribution and makes both the mean and noise variance available in the output layer.Following points describe Assume Density Filtering in a more detailed manner:

- Assumed Density Filtering(ADF) is a technique in Bayesian machine learning to approximate intractable and complex distribution with distributions that are easy to handle. In the case of Bayesian Inference, ADF aims to project the true posterior onto a distribution of choice. The exponential family of distributions are a popular choice.
- In the case of MCDO_ADF there is a need to propagate the input data distribution so that the values of its mean and variance (noise variance) are available in the output layer.
- The input data distribution is considered to be Gaussian in nature. Every intermediate layer outputs the transformed version of the input distribution. However, when it propagates through non-linearities in a Neural Network the resulting distribution need not be essentially another Gaussian. Such a distribution emerging out of non-linear blocks is also conditioned by distribution over activations of the preceeding layers. Therefore, the resulting distribution becomes intractable.
- Such intractable and complex distributions are estimated using ADF by:
  - Assuming conditional independence between distribution outputted from a given layer with its preceeding layers.
  - Approximating the complex distribution with a Gaussian distribution whose pair has the least possible value of Kullback-Leibler divergence(described in 4.1.1). ADF achieves this by matching the first two moments of the distributions.
  - In practice, this is achieved by optimizing the global variational objective.

In practice, every building block of a Neural Network has its corresponding ADF version and therefore during the inference time the entire model has to be transformed to its ADF equivalent. This gives the ability to the Neural Network model to propagate and output distributions which represent data uncertainty.

Data uncertainty estimation

The ADF transformed Neural Network produces two outputs from the final layer: mean ($\mu_{t^{(l)}}$) and variance($v_{t^{(l)}}$ of the propagated distribution, as shown in the figure 2.1. The pair of values is outputted for each of the T stochastic forward passes (described in the next paragraph) and the mean of T variance values is considered to be the value of data uncertainty. Likewise, the mean of T predictions is considered to be the model's prediction for the given input.

$$prediction = \mu = \frac{1}{T}\sum_{t=1}^{T} \mu_{t^{(l)}} \tag{2.1}$$

$$data\_uncertainty = \sigma_{data} = \frac{1}{T}\sum_{t=1}^{T} v_{t^{(l)}} \tag{2.2}$$

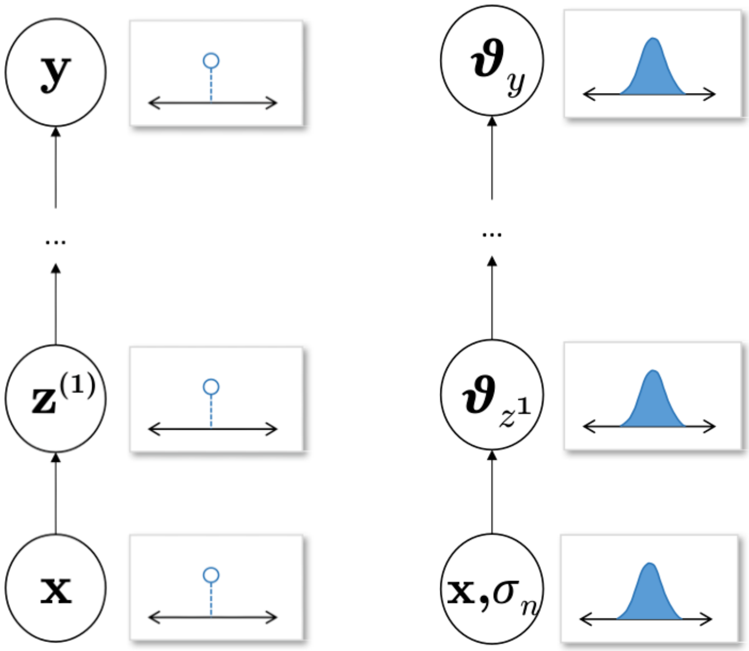


Figure 2.2: Illustration of forward passes in deterministic and ADF versions of a Neural Network. Here x denotes the input activations,$z^(n)$ denotes activation input to the nth layer,y denotes the output, θ_{z^n} represents input activation to the nth layer expressed as a probability distribution and σ corresponds to the noise variance. Image source:

Monte-Carlo Dropout (MCDO)

This uncertainty estimation method relies on the idea of Monte-Carlo(MC) sampling to estimate model uncertainty associated any prediction. In practice, MC sampling is achieved by enabling dropout during the test time and obtaining the desired number of samples (T), which are nothing but outputs of the Neural Network model during different forward passes of the input. Enabling dropout introduces stochasticity during those forward passes.

Following points briefly describe the dropout technique in a general context:

- Dropout([29]) is primarily a regularization technique used while training Neural Networks in order to avoid over-fitting.
- During dropout certain nodes of a given Neural Network layer are not considered for training. The nodes are ignored with a probability equal to the dropout rate (often denoted by **p**).
- Using dropout during training makes Neural Network layers to adapt in such a way that they cope with mistakes made my the prior layers.

In the context of Bayesian inference, the Dropout technique is used to approximate the posterior distribution over weights of a given Neural Network, when the training data and labels are given

$(P(W|X,Y))$. The approximation is obtained by applying dropout at the test-time. This makes it possible to obtain multiple predictions for any given input from different architectures resulting from application of dropout to the base Neural Network model. The different architectures obtained along with their weights can be considered as Monte-Carlo(MC) samples from the space of all possible architectures. The number of MC samples to be obtained is a hyper-parameter and denoted by T. In another perspective, T equals the number of forward passes through different architectures with different sets of weights $\{W_1^t, .., W_L^t\}_{t=1}^T$($L$ denotes the number of dropout applied layers in the Neural Network). The first and second moments (mean and variance) of predictions obtained from these stochastic forward passes of given input are utilized to compute model uncertainty(explained in **??**). One of the highlights of this technique is that its usage does not require any architectural changes and also can applied to already trained Neural Net models. The hyper-parameters T and p significantly impact the effectiveness of this technique. In the case of p a very high value (close to 1) increases sparsity in nodes and also results in longer convergence-time while a low value eliminates the MC-sampling utility. For our experiment, the value of $p = 0.02$ is used. The hyper-parameter T significantly impacts the inference time of a Neural Network model and therefore has to be chosen optimally based on the run-time requirement of the system where the model would be deployed.

Model uncertainty estimation

The MCDO_ADF technique estimates model uncertainty using predictions generated from the Neural Network model during multiple(T) forward passes, while the dropout is enabled. A given input is processed by the model T times, with a new combination of neurons considered for almost every forward pass. This produces an effect of gathering predictions from an ensemble consisting of T Neural Net models with different architectures. The variance of T gathered predictions is the estimated model uncertainty. In the following equations $\mu_{t^{(l)}}$ signifies the mean output from the ADF transformed version of the Model during t^{th} forward pass.

$$model\_uncertainty = \sigma_{model} = \frac{1}{T}\sum_{t=1}^{T}(\mu_{t^{(l)}} - \overline{\mu})^2 \tag{2.3}$$

$$where, \overline{\mu} = \frac{1}{T}\sum_{t=1}^{T}(\mu_{t^{(l)}} \tag{2.4}$$

Combining ADF and MCDO

The MCDO_ADF method considers a relationship to exist between the two components of uncertainty (data and model). The relationship is realized in this technique by combining both the ideas of ADF and MCDO. During inference,

- The given Neural Network model is transformed to its ADF equivalent so that the output layer produces both predictions(mean) and noise variance as the model's final outputs.

- For estimating model uncertainty, dropout is enabled in the ADF transformed version of the original Neural Network following which T MC samples are collected during T stochastic forward passes. It is this application of dropout on the ADF transformed version that produces the "effect of ensembling T ADF Neural Networks" and also considers any relationship between the two uncertainty components.
- Combining ADF and MCDO leads to another intuitive realization about the uncertainty components in this setup. Even when a particular input fed to the Neural Net model was observed frequently during training, if corrupted due to sensor noise then it will not only have high values of both data and model uncertainties.

Total uncertainty

The predictive uncertainty is estimated by summing up both its components (data and model) and is given by the following equation.

$$predictive\_uncertainty = \sigma_{total} = \frac{1}{T}\sum_{t=1}^{T}((\mu_{t^{(l)}} - \overline{\mu})^2 + v_{t^{(l)}}) \tag{2.5}$$

In summary, both ADF and MCDO techniques approximate probability distributions of data and model with Normal distributions respectively. ADF propagates the input data distribution and approximates them as Gaussians in every Neural Network layer while MCDO approximates the distribution around weights by sampling and forms a Gaussian distribution out of the samples. The variances of these Gaussian distributions are considered to be the uncertainty components and are summed up to yield the predictive uncertainty.

2.7.3 Inference procedure

The MCDO_ADF method can be applied to already trained deterministic version of the Neural Network models as mentioned in 2.7.1. However, it is also possible to train the Neural Network model of interest with dropout enabled and use the same for inference. In order to estimate the predictive uncertainty during inference:

- Every layer of the Neural Network has to replaced with its ADF equivalent so that they are equipped with the ability to propagate data distributions. The implementation of ADF equivalents for most of the Neural Network building blocks is available in the Github repository of [9].
- The value of noise variance (a constant value) obtained from the sensor's data sheet is fed along with the input data to the ADF transformed input layer of the network. During propagation through intermediate layers, it is ensured that at least a minimum value of variance is propagated. In the case of experiment described in the next chapter, a minimum value of 0.001 is used.
- Every input along with the noise variance undergoes T stochastic forward passes through the network to generate T predictions.

The experiment described in the next chapter discusses more on practical aspects of this technique.

2.7.4 Downsides

- The need for multiple (T) forward passes to obtain MC samples is computationally expensive and cannot be afforded in the case of real-time systems.While there is an option to reduce the value of T, it increases the difference between approximated and underlying distribution over weights of the model, thereby affecting the method's performance.
- The method considers sensor noise to be the only source of data uncertainty. Also, it treats the noise to be additive Gaussian in nature. However, sensor noise is just one of the factors contributing to data uncertainty. For instance, in the case of image data, usage of a lossy compression technique can also contribute to its noise. Also, it is possible for a given sensor to produce data whose noise levels differ. As it is impossible to consider and model every possible noise source, it is important for an uncertainty estimation method to learn to differentiate noise and useful information from given data.
- The authors of MCDO_ADF quote its ability to be applied to already trained models as one of the key reasons for its generality. However, retraining a Neural Network model is something feasible in most of the cases.
- As hyper parameters such as drop-out rate(p), number of MC samples(T) and noise variance have a major role to play in this technique, it adds to the responsibilities of the practitioner to optimally choose them based on the problem at hand.

2.8 Deep Evidential Regression

2.8.1 Overview

Deep Evidential Regression proposes a method (hereafter referred to as ”DER”) to estimate predictive uncertainty primarily in Neural Networks for regression, by simultaneously learning a hierarchy of distributions. The learned hierarchy consists of two levels of distributions: 1. A lower level Gaussian distribution over data, with parameters (mean μ and variance σ^2) 2. A higher level(also called Evidential) Normal-Inverse Gamma distribution over the parameters of the lower level distribution. In the perspective of the Bayesian Inference, the higher-order distribution can be taken as a prior over the the lower-order distribution which is obtained by evaluating likelihood of known data points for a particular choice of μ and σ^2. The evidential distribution evaluated at any particular instance(a combination of μ and σ^2), provides the subjective belief mass of the corresponding lower-order distribution there. This subjective belief mass is also called as ”evidence”. Lack of evidence means existence of uncertainty and therefore the value of evidence is used to quantify predictive uncertainty.

In order to put the above mentioned ideas into practice, DER provides a loss function whose objectives are to:

- Fit the training data to the evidential model.
- Learn the evidential prior which would provide uncertainty estimates during inference.
- In simple words, to learn the parameters of the higher-order evidential distribution.

The upcoming sections of this chapter explain the method in a detailed manner.

2.8.2 Conjugate priors

Let us consider a learning problem where Random Variables(RV) Θ and Y represent model parameters and data respectively. Assuming that RVs are jointly distributed and applying Bayes Rule to determine the probability distribution of Θ given Y,

$$P(\Theta|Y) = \frac{P(Y|\Theta)P(\Theta)}{P(Y)}$$

The equation can be expressed in words as follows:

$$\text{Posterior distribution of } \Theta \text{ given Y} = \frac{\text{Likelihood of Y given } \Theta \text{ . Prior over } \Theta}{\text{Marginal Likelihood of Y}}$$

During inference, for a particular choice of functions to represent the likelihood distribution, the nature of prior distribution function matches the nature of posterior distribution function. For example, if a normal distribution with unknown mean and variance is used to represent the likelihood distribution and if a Normal-Inverse Gamma distribution(NIG)(described in the next subsection) is used to represent the prior distribution then nature of posterior probability distribution is also observed to be Normal-Inverse Gamma in nature. This can be briefly written as ”Normal-Inverse Gamma distribution is the conjugate prior for Normal distribution in likelihood”. Conjugate priors help to reduce computations involved in determining the $P(\Theta|Y)$ value every time during the process of determining optimal set of parameters. Beta, Gamma and Normal distributions are favorite choices for priors as they act as conjugate priors for different likelihood distribution functions. In the context of DER, the conjugate prior relationship between distributions is used to introduce a hierarchy between the them to probablistically model the likelihood distribution.

Distribution hierarchy

Let us assume using a Normal distribution $\mathcal{N}(\mu, \sigma^2)$ to model a set of data points $x_1, x_2, .., x_i$.”When a probability distribution A is used to model the given set of data, the uncertainty in the fit is described by probability distribution/s B over parameters of A”. This means defining probability distributions over the set of parameters μ, σ^2 helps in describing uncertainty in the model fit.

The probability distribution of μ is modeled by a normal distribution due to its Gaussian nature and the fact that $\mu \in \mathbb{R}$. On the other hand, a Gamma distribution($\Gamma(\alpha, \beta)$) is used to model the probability distribution of σ^2 owing to its strictly positive nature. The following figure illustrates hierarchical relationship between distributions under consideration, where $(\mu_0, \sigma_0^2), (\alpha, \beta)$ represent the parameters of the higher-order Normal and Gamma distributions respectively.

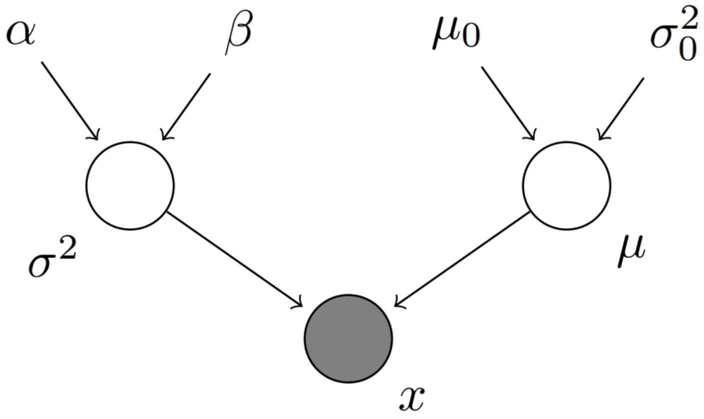


Figure 2.3: Hierarchy in distribution parameters.Image source:

Alternatively, a Normal-Inverse-Gamma distribution (often represented as NIG(α,β,γ,λ)) can be used to model the probability distribution of μ and σ^2 jointly. DER uses the distribution to realize its objectives. Significance of NIG's parameters is explained under the next section.

2.8.3 Evidential distribution

From the perspective of Bayesian inference

Let us consider a regression problem with an available dataset D with N pairs of data labels and targets represented by $(x_1, y_1), .., (x_N, y_N)$. The DER method assumes that the targets are drawn independent and identically from a Gaussian distribution with unknown mean and variance represented by μ and σ^2 respectively.

$$(y_1, .., y_N) \sim \mathcal{N}(\mu, \sigma^2) \tag{2.6}$$

The parameters μ and σ^2 are considered to be random variables that follow Gaussian and Inverse-Gamma distributions respectively.

$$\mu \sim \mathcal{N}(\gamma, \sigma^2\lambda^{-1}) \tag{2.7}$$

$$\sigma^2 \sim \Gamma^{-1}(\alpha, \beta) \tag{2.8}$$

where $\alpha, \beta, \gamma, \lambda$ denote parameters of the higher-order Normal Inverse Gamma (NIG) distribution. Let $\theta = (\mu, \sigma^2)$ denote the parameters of one instance of Gaussian distribution generating targets y_i and $m = (\alpha, \beta, \gamma, \lambda)$ denote the set of NIG distribution parameters.

We are interested to model the distribution around θ. Applying Bayes Rule, we get

$$P(\theta|m) = \frac{P(m|\theta)P(\theta)}{P(m)}, \tag{2.9}$$

$$\text{posterior. dist. over } \theta \text{ for the given value of m} = \frac{\text{likelihood of m evaluated at the given value of } \theta \text{ x prior over } \theta}{\text{likelihood of m evaluated at all possible values of } \theta}$$

Here, the prior over θ is a NIG distribution and the likelihood function is Gaussian in nature. Therefore, the posterior takes the form of an NIG distribution expressed as follows:

$$P(\mu, \sigma^2 \mid \gamma, \lambda, \alpha, \beta) = \frac{\sqrt{\lambda}}{\sigma\sqrt{2\pi}} \frac{\beta^\alpha}{\Gamma(\alpha)} \left(\frac{1}{\sigma^2}\right)^{\alpha+1} \exp\left(-\frac{2\beta + \lambda(\gamma - \mu)^2}{2\sigma^2}\right) \tag{2.10}$$

Significance of NIG parameters

γ and α are shape and location parameters of NIG distribution respectively. β refers to the inverse-scale (rate) parameter. This means that spread of the distribution is inversely related to β. There is a relationship that exists between parameters of the NIG distribution. γ can be interpreted as the sample mean of λ virtual observations, determining NIG's location. On the other hand, spread of the NIG distribution can be considered to have calculated from 2α virtual observations whose sample mean equals γ and their squared deviations summing to 2β. DER considers the count of virtual observations as evidence(ϕ) in support of the data sample at hand.

$$\phi = \lambda + 2\alpha \tag{2.11}$$

Figure 2.4 illustrates the impact of increase in evidence on the shape and spread of the NIG distribution (column **A**) and various realizations of the lower-order likelihood distribution (column **C**) from a given instance of NIG distribution(column **B**). Following are some of the key insights that can be obtained from the illustration:

- With increase in evidence (as expressed in 2.11) the belief mass increasingly concentrates around a specific value pair of μ and σ^2 in the column **A** meaning a reduction in uncertainty.

- Column **B** illustrates the evidential distribution centered around a particular value pair of μ and σ^2. This intuitively means that every point on the distribution corresponds to parameters of a possible likelihood distribution.

- Sampling the higher order distribution at various locations yield likelihood distributions of varying levels of evidence associated with them. Column **C** in the illustration shows such realizations. Darker the blue shade used to represent a likelihood-distribution, higher the evidence level associated with it.

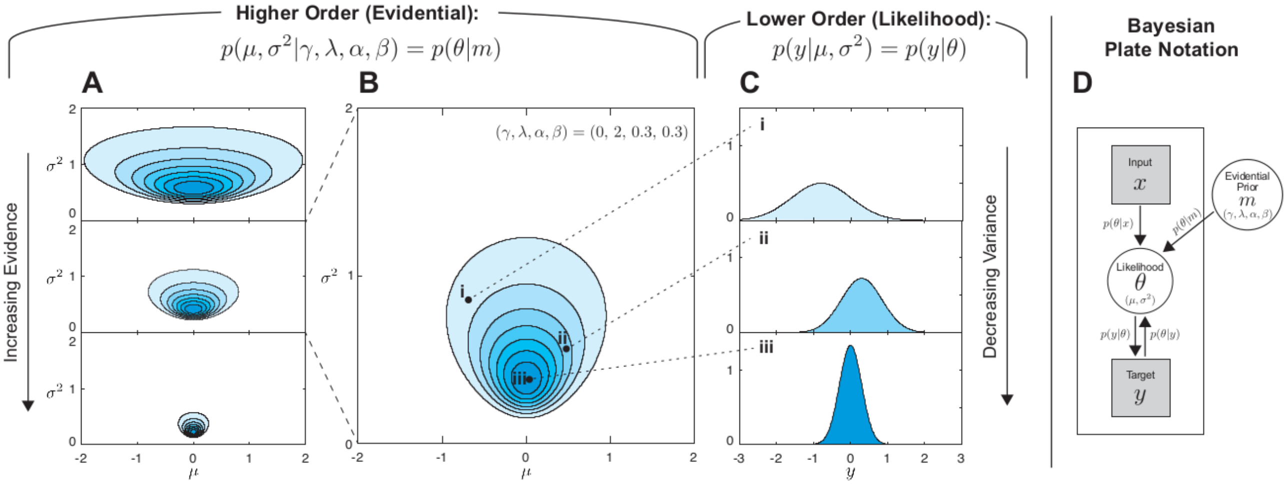


Figure 2.4: Realizations of the NIG distribution. Image source:

2.8.4 Evidential learning objectives

As described in the overview(2.8.1), the objectives of DER are two fold: 1. Maximize the model fit and 2. Minimize the evidence measure in an event of error. The method realizes these objectives in form of a loss function/s(two forms) which is integrated to the Neural Network of choice and optimized during training.

Maximizing the model fit

This objective of DER focuses on learning the underlying patterns in the data and also increasing the belief mass/evidence in favor of right predictions.

Rewriting the equation 2.9

$$P(\theta|m) = \frac{P(m|\theta)P(\theta)}{P(m)}$$

Let us assume that we observe our target y_i which when added to the above equation yields,

$$P(\theta|y_i,m) = \frac{P(y_i|\theta,m)P(\theta|m)}{P(y_i|m)} \tag{2.12}$$

Before interpreting the above equation it is important to recollect the fact that in Bayesian Inference every Random Variable(RV) involved is considered to be jointly distributed. In the case of above equation, Random Variables Y,Θ and M corresponding to y_i, θ and m are jointly distributed.

In the above equation, we determine the probability distribution around θ when it is conditioned under specific values of y_i and m. The two terms in the numerator denote likelihood and prior as described in

2.8.3. The denominator term is termed as ”marginal likelihood” or ”evidence” in Bayesian inference, as it yields the total probability mass of $Y = y_i$ for all possible realizations of θ in the model parameterized by m. The evidence term is also important to normalize the likelihood so that it represents a probability measure. Column D of the figure 2.4 illustrates Bayesian inference in DER. The marginal likelihood term can be represented mathematically as follows:

$$P(y_i|m) = \int_{\theta} P(y_i|\theta, m)P(\theta|m)d\theta \tag{2.13}$$

$$P(y_i|m) = \int_{\sigma^2=0}^{\infty} \int_{\mu=-\infty}^{\infty} P(y_i|\mu, \sigma^2)P(\mu, \sigma^2|m)d\mu d\sigma^2 \tag{2.14}$$

The proposed loss function aims to determine the set of parameters m which maximizes the term $P(y_i|m)$(evidence) for the given target y_i. Similar to Maximum Likelihood Estimation, the objective of maximization of marginal likelihood is re-framed as minimization of negative log of marginal-likelihood(NLL) for computational convenience. The loss function is expressed as:

$$\mathcal{L}_i^{NLL}(w) = -logP(y_i|m) \tag{2.15}$$

$$\mathcal{L}_i^{NLL}(w) = -log(2^{0.5+\alpha}\beta^{\alpha}\sqrt{\frac{\lambda}{2\pi(1+\lambda)}}(2\beta + \frac{\lambda(\gamma - y_i)^2}{1+\lambda})^{-0.5-\alpha}) \tag{2.16}$$

Alternative to the usual way of minimizing Negative log likelihood the author proposes yet another form for the loss function which minimizes sum-of-squared errors between the prior and data sampled from the likelihood function. Following is the expression for the Sum Of Squared errors(SOS) version of the loss function:

$$\mathcal{L}_i^{SOS}(w) = (\frac{\Gamma(\alpha - 0.5)}{4\Gamma(\alpha)\lambda\sqrt{\beta}})(2\beta(1+\lambda) + (2\alpha - 1)\lambda(y_i - \gamma)^2) \tag{2.17}$$

The author claims the SOS version of loss function to be relatively stable while training and also to perform better than the other during evaluation. The portion of loss function $\mathcal{L}$ described in this section only achieves the ”model fitting” objective of DER.

Minimizing evidence on errors

The second objective of DER aims to minimize the evidence measure or to inflate uncertainty in the absence of training data. DER expresses this objective by adding a regularizer term to the loss function which penalizes the loss function in an event of its prediction deviating from the ground truth label. The penalty is scaled by evidence which expressed as sum of virtual observations as described in 2.11. Following is the expression for the regularizer term,

$$\mathcal{L}_i^{R}(w) = \|y_i - \gamma\| .(2\alpha + \lambda) \tag{2.18}$$

Here p refers to the order of norm used to represent the difference between the ground truth label y_i and predicted mean γ. Author uses the value of p=1 claiming it to be stable during the training process.

Putting both its objectives together the evidential loss function can be expressed as:

$$\mathcal{L}_i(w) = \mathcal{L}_i^{SOS}(w) + \mathcal{L}_i^{R}(w) \tag{2.19}$$

2.8.5 Estimating uncertainty

Epistemic uncertainty which quantifies the model's inherent lack of knowledge associated with an output can be expressed as the variance around its predictions.

$$Var(\mu) = \frac{\beta}{(\alpha - 1)\lambda} \tag{2.20}$$

The λ term in the denominator refers to the number of virtual observations. Aleatoric uncertainty can be computed with the following expression:

$$\mathbb{E}[\sigma^2] = \frac{\beta}{\alpha - 1} \tag{2.21}$$

From equations 2.21 and 2.20 both components of uncertainty can be related as follows:

$$epistemic\_uncertainty = \frac{aleatoric\_uncertainty}{\lambda} \tag{2.22}$$

This means that the epistemic uncertainty component is the mean of aleatoric uncertainty over λ virtual observations.

Predictive uncertainty can be evaluated as the sum of epistemic and aleatoric uncertainty components.

$$predictive\_uncertainty = aleatoric\_uncertainty + epistemic\_uncertainty \tag{2.23}$$

From eqns 2.21 and 2.20 predictive uncertainty can be computed as follows:

$$predictive\_uncertainty = \frac{\beta}{\alpha - 1} + \frac{\beta}{(\alpha - 1)\lambda} \tag{2.24}$$

After simplification,

$$predictive\_uncertainty = \frac{\beta(1 + \lambda)}{(a - 1)\lambda} \tag{2.25}$$

3

Methodology

This chapter explains different experiments conducted to compare the two state-of-the-art uncertainty methods considered for this research work. Also, the experimental results are presented and analyzed.

3.1 Datasets

3.1.1 Steering angle dataset

The Udacity steering angle dataset (available in [**?**]) consists of driving scene images captured by a set of three cameras(left, center, right) mounted behind the windshield of an ego vehicle. Along with the captured images, the dataset also contains steering angle, torque and vehicle speed values logged at that particular instance. The experiments conducted in this research work only utilizes the images captured the center camera and their corresponding steering angles expressed in radians. The data set contains driving scene images captured during different weather and traffic conditions(dataset samples can be found in 3.1). The data set consists of 33,808 images in total and for the experiments conducted in this research, a train-validation-test split ratio of 80:5:15 is used. The following table gives further details about the dataset.

Dataset folder identifier	**Conditions**	**Count**			
		Train	**Validation**	**Test**	**Total**
HMB_1	Divided highway and sunny conditions	3521	220	660	4401
HMB_2	Two lane road and sunny conditions	12637	790	2369	15796
HMB_4	Divided highway segment	1579	99	296	1974
HMB_5	Guard rail and two lane road	3388	212	635	4235
HMB_6	Divided multi-lane highway with a fair traffic and shadows prevalent all over	5922	370	1110	7402

Table 3.1: Train-validation-test split of the Udacity steering angle dataset

Steering angle prediction is both a safety and time critical application which serves as an essential component of any autonomous vehicle. As enhancing functional safety in such applications is one of the key objectives of using uncertainty estimation methods, the steering angle data set is chosen for benchmarking the techniques considered for this research work.

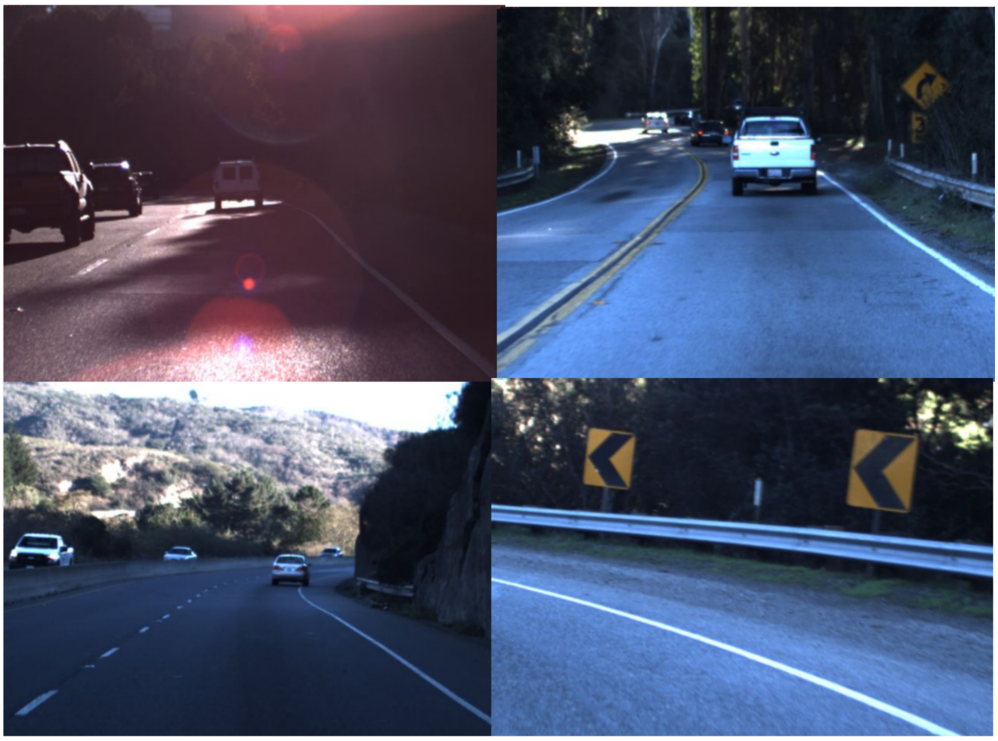


Figure 3.1: Sample images from the Udacity steering angle dataset. Image source:

3.1.2 1D datasets

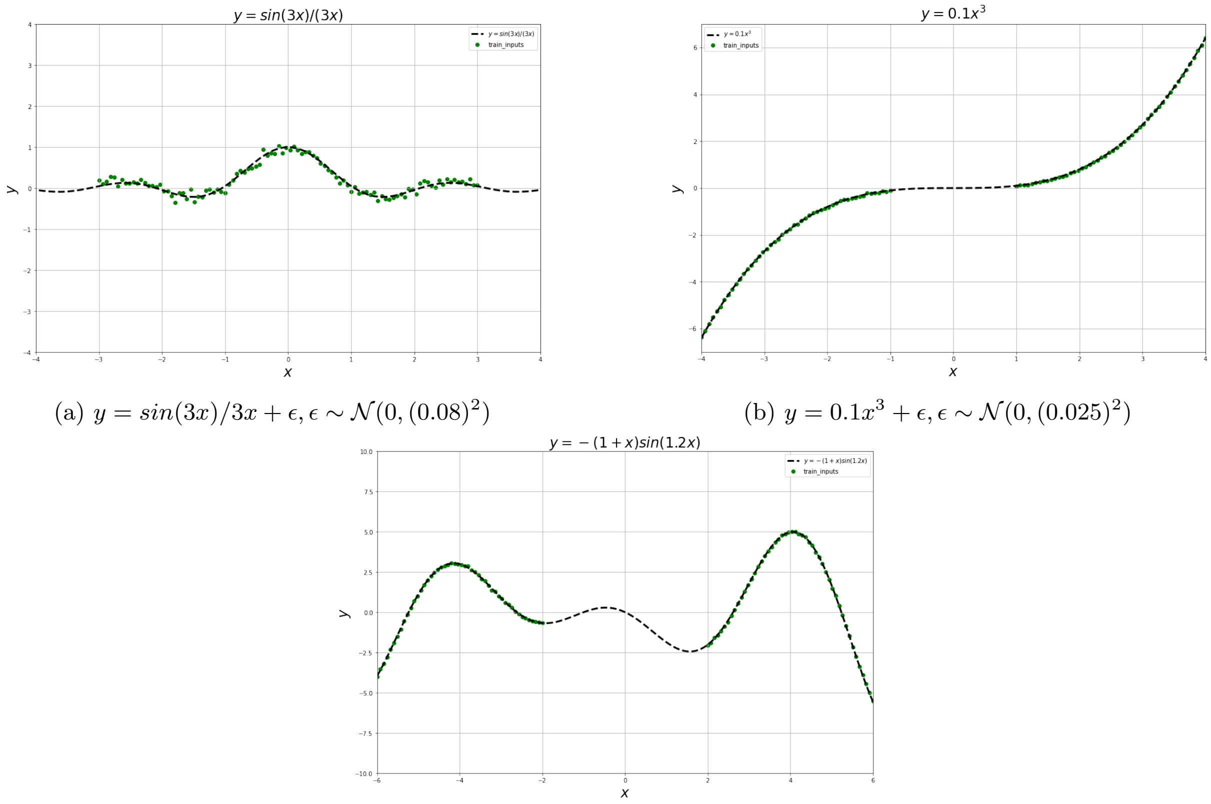
A set of three different one-dimensional functions are used to evaluate the considered state-of-the-art uncertainty methods. Use of low-dimensional datasets for evaluation has certain advantages: increased control over experimental parameters when compared to high-dimensional data, ease in visualizing results and reduced experimentation time. The following table lists the set of functions considered for this research work along with their specifications.

Identifier	**1D function** $y = f(x) + \epsilon$	**Noise parameters** $\epsilon \sim \mathcal{N}(\mu, (\sigma)^2)$	**Train and test data range** **(points are equally spaced)**
fn_1	$y = sin(3x)/3x + \epsilon$	$\epsilon \sim \mathcal{N}(0, (0.08)^2)$	train: 100 points in [-3,3], test: 100 points in [-5,5]
fn_2	$y = 0.1x^3 + \epsilon$	$\epsilon \sim \mathcal{N}(0, (0.025)^2)$	train: 50 points in [-4,-1] $\bigcup$ 50 points in [1,4] test: 200 points in [-4,4]
fn_3	$y = -(1 + x)sin(1.2x) + \epsilon$	$\epsilon \sim \mathcal{N}(0, (0.04)^2)$	train: 50 points in [-6,-2] $\bigcup$ 50 points in [2,6] test: 200 points in [-6,6]

Table 3.2: Specifications of 1D datasets

Train and test data ranges for every function considered are chosen with an intent to evaluate the performance of uncertainty estimation methods in both within and outside the bounds of training data. For fn_1, the test data range is chosen to lie on either sides of the of the training data range. For the other two functions, the test data ranges are chosen to lie in between their train data ranges.

A zero-mean Gaussian noise is added to training data of all three functions, with different values of standard-deviation. The set of functions are plotted in the following figure.



(a) $y = sin(3x)/3x + \epsilon, \epsilon \sim \mathcal{N}(0, (0.08)^2)$

(b) $y = 0.1x^3 + \epsilon, \epsilon \sim \mathcal{N}(0, (0.025)^2)$

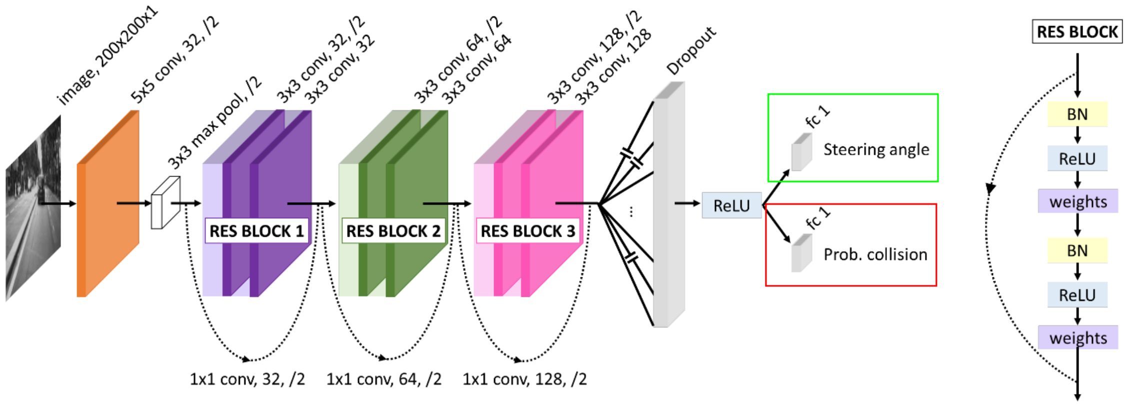
(c) $y = -(1 + x)sin(1.2x) + \epsilon, \epsilon \sim \mathcal{N}(0, (0.04)^2)$

Figure 3.2: Functions in 1D dataset

3.2 Network architectures

3.2.1 Dronet

Dronet, a residual convolutional network architecture is used for experiments conducted on the steering angle dataset. The Neural Network is primarily designed to safely navigate a drone by performing the tasks of steering angle prediction(regression) and collision detection(binary classification). However, for the experiments conducted in this research work the collision detection output is not required and therefore its corresponding output branch in the Neural Networks output layer is discarded. Figure 3.3 depicts Dronet's architecture.



(a) (b)

Figure 3.3: (a) Dronet architecture (b) Structure of every residual block . Collision classification output (bounded by the red box) is discarded and the steering angle prediction branch(bounded by the green box) is retained for this experiment. Image source:

The model used for experiments takes a gray-scale input of size 200x200 and propagates it through a pair of convolution and max-pooling layers, three residual blocks, a dropout layer, a ReLU activation and finally a fully-connected(fc) layer which outputs predicted steering angles. When it comes to integrating MCDO_ADF(described in 2.7) with Dronet, new dropout layers are introduced before every convolutional layer at the test time. On the other hand, DER (described in 2.8) is integrated to Dronet by introducing three more output branches in the last fc layer for outputting parameters of the evidential distribution (described in 2.8.3).

3.2.2 Neural Network for 1D datasets

In order to conduct experiments on the set of 1D functions, a simple Neural Network architecture consisting of two fully-connected layers and a ReLU(Rectified Linear Unit) non-linearity is used(as shown in). The network is modified based on requirements of the integrated uncertainty estimation method. In the case of variant used with DER, the last fully connected layer is modified to output parameters of the evidential distribution. The MCDO_ADF variant of the Neural Network contains an additional dropout layer in order to facilitate extraction of MC samples, as described in 2.7. Also, after training the model is converted to its ADF equivalent.

3.2.3 Gaussian Processes

3.3 Training details

3.3.1 Dronet with steering angle dataset

In order to benchmark the considered uncertainty estimation methods(MCDO_ADF and DER), a set of three Dronet models are used: 1. Vanilla version of Dronet 2. MCDO_ADF version of dronet with dropout layers after convolution layers 3. Evidential version of Dronet which uses the evidential loss function. Training details for those variants are provided in upcoming sections.

Choice of certain hyperparameter values remains unchanged for training all the three models and are listed in the table below.

Hyperparameter	**Value**
Input image size (hxwxc)	200 x 200 x 1
Batch size	32
Training epochs	100
Learning rate	0.001
Dropout rate	0.2
Weight decay	0.0001
Learning rate decay	0.00001
Choice of optimizer	Adam
Initializers	Kaiming-normal for Conv2D in residual blocks, Xavier-uniform for Conv2D, linear layers in non-residual blocks, Constant initialization for batch normalization layers (weights with 1 and biases with 0)

Vanilla Dronet

A simple dronet model predicts steering angle for the given image input. Training the model involves reduction of the Mean Squared Error (MSE) loss, which is popular choice for regression problems. Optimizing the MSE loss function intuitively means reduction of mean over euclidean distance (L2-Norm) between ground truth labels and predictions of observed data. The MSE loss function can be expressed as follows:

$$\mathbf{L}(y, \hat{y}) = \frac{1}{N}\sum_{i=0}^{N}(y - \hat{y}_i)^2 \tag{3.1}$$

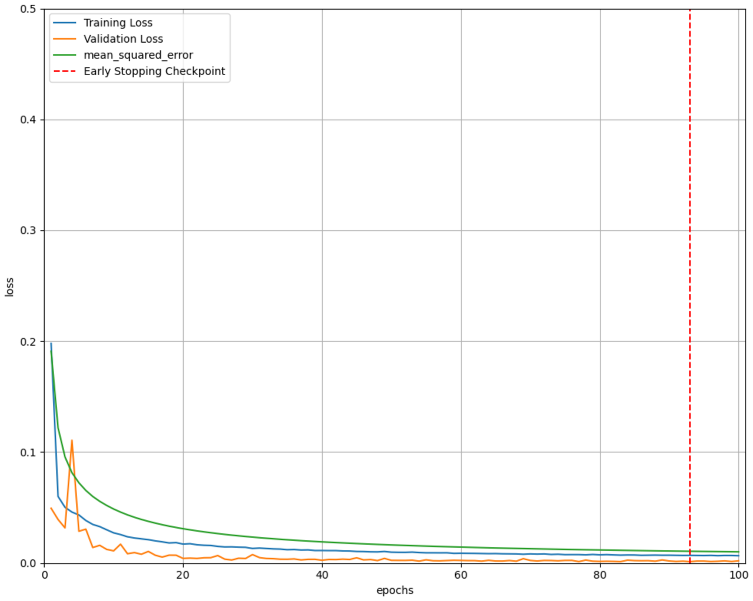


Figure 3.4: Plot depicting the trend of Mean-Squared-Error loss while training vanilla Dronet

The technique of early stopping is used to avoid over-fitting, by saving model weights at the epoch corresponding to the least validation loss.

MCDO_ADF Dronet

This variant of Dronet is trained to facilitate estimation of uncertainty associated with its predictions using the MCDO_ADF technique. The training procedure for this variant remains unchanged from the Vanilla variant except for the introduction of dropout after every convolution layer. Though it is sufficient to have the vanilla variant for applying this method, dropout was used during training with an intent to regularize the process.

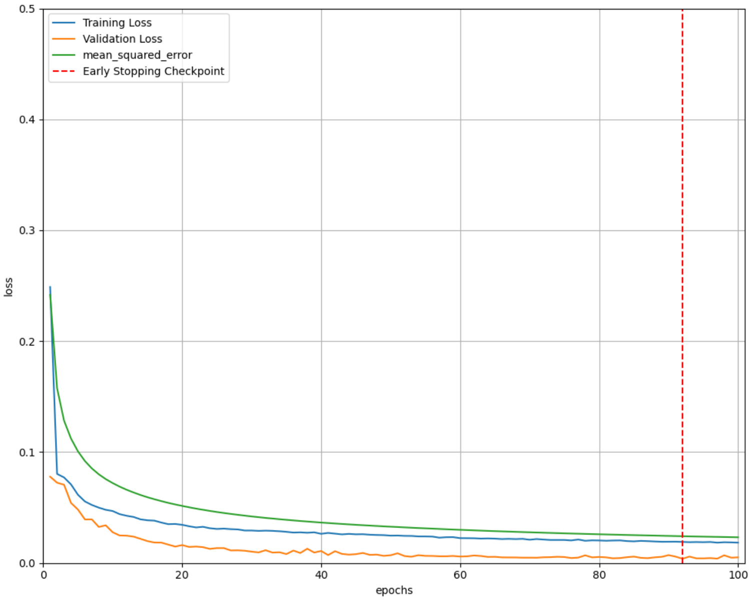


Figure 3.5: Plot depicting the trend of Mean-Squared-Error loss while training MCDO_ADF Dronet

The inference procedure for MCDO_ADF applied models is clearly explained in 2.7.3. There are three hyper-parameters additionally required for the procedure: 1. Monte-Carlo(MC) sample count 2. Noise-variance 3. Minimum-variance. The value of MC sample count has a direct impact on the inference time as it determines number of forward passes for a given input to determine model uncertainty. For this experiment, we set its value to be 20 to replicate results provided in [23]. Both the values of noise-variance and minimum variance are chosen to be 0.001. Noise variance indicates the level of sensor noise and minimum-variance signifies the minimum value of noise-variance to be propagated through every layer.

Evidential Dronet

The evidential Dronet model outputs parameters of the evidential distribution(described in the section 2.8.3) for a given input image. The distribution parameters can be in turn used to calculate prediction and uncertainty associated with it. Except for the choice of evidential loss function(described in 2.8.4) for this model , training criteria remains unchanged from the vanilla variant.

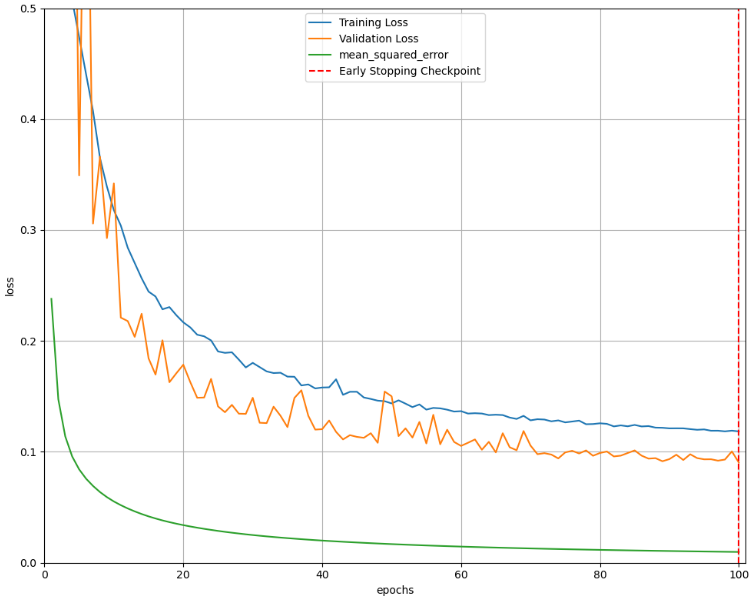


Figure 3.6: Plot depicting the trend of evidential loss while training evidential Dronet

3.3.2 Neural Net with 1D dataset

Similar to the case of Dronet, two variants of the simple network described in 3.2.2 are used for evaluation. The MCDO_ADF variant is trained using MSE loss, while evidential loss is used for its evidential counterpart. Also, the MCDO_ADF variant uses dropout during test time. The following table contains values of hyperparamters that remain common to both the variants.

Hyperparameter	**Value**
Batch size	Training data set size
Optimizer	Adam
Learning rate	0.001
Epochs	20000
Initializers	Normal initializer for weights and zeros for bias

Table 3.3: Choice of hyperparameters for Neural nets used with 1D datasets

4

Experimental Evaluation

4.1 Metrics

4.1.1 Kullback-Leiber Divergence

4.1.2 Root Mean Squared Error(RMSE)

Root Mean Squared Error (RMSE), measures the spread of distances between model predictions and their corresponding ground truth values. Alternatively, it can be explained as the standard deviation of prediction errors. RMSE is a well-known accuracy metric in regression problems. The metric is non-negative in nature with lower values indicating better model fit.

$$\mathbf{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{N}} \tag{4.1}$$

Here $\hat{y}$,y and N represent ground truth labels, predictions and number of data points respectively.

4.1.3 Explained Variance(EVA)

The Explained Variance (EVA) is a measure of a regressor's ability to capture variance(variation) of given data. The metric can be computed with the following expression:

$$\mathbf{EVA} = \frac{Variance(\hat{y} - y)}{Variance(\hat{y})} \tag{4.2}$$

Here $\hat{y}$,y represent ground-truth labels and predictions respectively. The numerator term denotes variance of residuals whereas the denominator denotes the underlying variance in ground-truth labels. For an ideal regressor, the value of EVA equals 0.

4.1.4 Negative-Log-Likelihood(NLL)

In this research work,the Negative-Log-Likelihood(NLL) metric is used compare performance of uncertainty estimation methods. NLL for a given pair of prediction and uncertainty can be computed as follows

- A distribution (often Gaussian) is created with the model prediction as its mean and uncertainty as its variance.
- The conditional probability of observing the ground-truth label(corresponding to the input) in the created distribution is determined. This is nothing but the likelihood value of ground-truth in the created distribution.
- In order to handle and effectively represent very low values of likelihood, negative of natural logarithm is applied to the value. After application of negative logarithm, the total likelihood can be computed by summing all individual values.

$$\mathbf{NLL} = -\sum_{i=1}^{N} \ln P(\hat{y}_i|\mathcal{N}(y_i, \sigma^2)) \tag{4.3}$$

Here $\hat{y}$,y,σ^2 and N represent ground truth labels, predictions, predictive uncertainty and number of data points respectively. Lower the value of NLL better the performance of an uncertainty estimation method associated with it. However, the value of NLL highly depends on the number and choice of test points used for evaluation. Therefore, the metric can be used to compare performance of uncertainty estimation methods only when the same data set is used for their evaluation.

Limitations of NLL

NLL measures the goodness of fit of the approximated posterior distribution to the true function's mean(ground truth). The metric fails to evaluate fidelity of the approximated posterior distribution. Therefore, NLL "may be a good criteria for model selection, it is not a reliable criteria for determining how well an approximate posterior aligns with the true posterior"[31]. This can be understood better with help of the illustration below.

4.2 Predictive accuracy and uncertainty quality

4.2.1 Quantitative comparison

- It can be inferred that Evidential Dronet outperforms the other two models in terms of both predictive accuracy and quality of uncertainty estimation.
- In the case of predictive accuracy expressed in terms of RMSE, Evidential Dronet outperforms the Vanilla variant only by a slight margin. However, difference in RMSE between Evidential

Model	**RMSE**	**EVA**	**NLL**		
			Epistemic	**Aleatoric**	**Predictive**
Vanilla Dronet	0.034	0.98	NA	NA	NA
MCDO_ADF Dronet	0.15	0.68	-0.71	7.51	-0.74
Evidential Dronet	**0.022**	**0.99**	**-2.03**	**-1**	**-0.94**

Table 4.1: A quantitative comparison of uncertainty estimation methods when applied to Dronet

and MCDO_ADF variants is considerable. This could be partly attributed to the fact that both predictions and uncertainty estimates outputted by the MCDO_ADF variant depend on the count of Monte-Carlo(MC) samples considered. Higher the value of MC samples better the model's performance(refer figure 4.1). However, this holds true only un till a particular value of MC samples. For this experiment, 20 MC samples are considered.

- The relationship between MC sample counts and predictive accuracy holds good for the Explained Variance (EVA) measure as well.
- The quality of predictive uncertainty estimated by the Evidential Dronet expressed in terms of Negative-Log-Likelihood (NLL) is better than the MCDO_ADF variant. This intuitively means that the former technique is able to determine parameters of the distribution in which likelihood of finding ground truth labels is higher than in the distribution outputted by the latter.

4.2.2 Qualitative comparison

In order to qualitatively compare predictive uncertainties outputted by both the models, test images corresponding to the highest and lowest values of uncertainty(aleatoric, epistemic and predictive) are taken into consideration and analyzed. The analysis is presented in the table below.

4.3 Out-Of-Distribution(OOD) testing

One of the important uses for having an estimate of uncertainty associated with a Neural Net model's output is "selective prediction". This means that the confidence estimate can be used to determine the correctness of an output and in turn decide whether to consider it for further processing/decision-making or not. It is crucial for the uncertainty estimation method to produce a higher value of uncertainty when the model faces test samples from an unknown data distribution. This section evaluates the considered uncertainty estimation methods on their response to out-of-distribution inputs.

4.3.1 Response to Out-Of-Distribution data

4.3.2 Robustness to adversarial attacks

Adversarial examples are malicious inputs designed to fool machine learning models[[17]]. Such data samples can be considered as an extreme case of Out-Of-Distribution as they are synthesized by perturbing

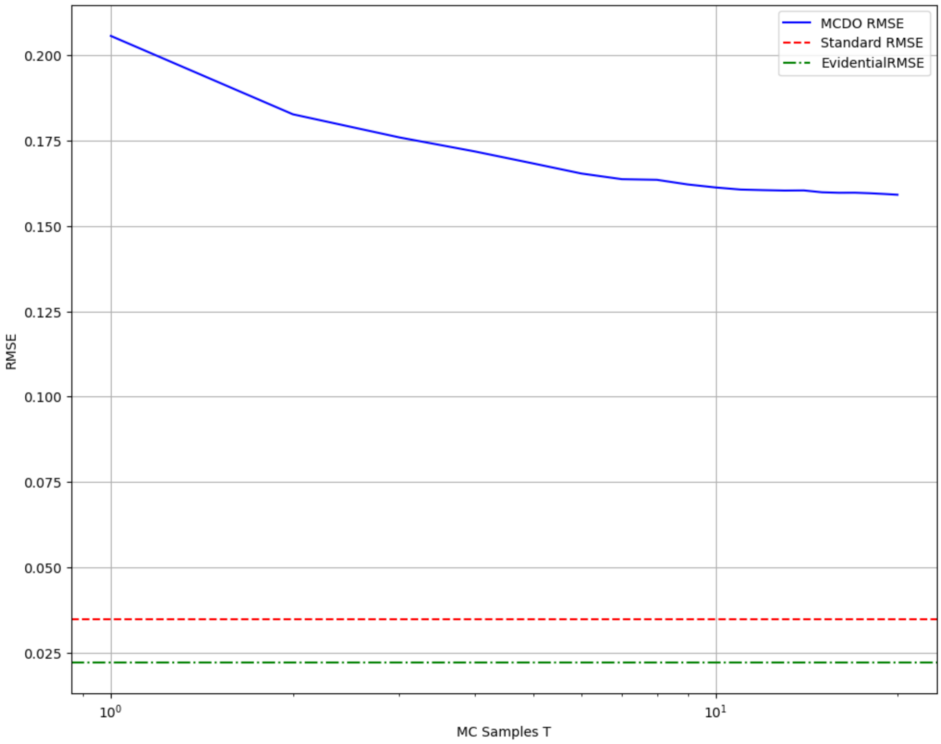
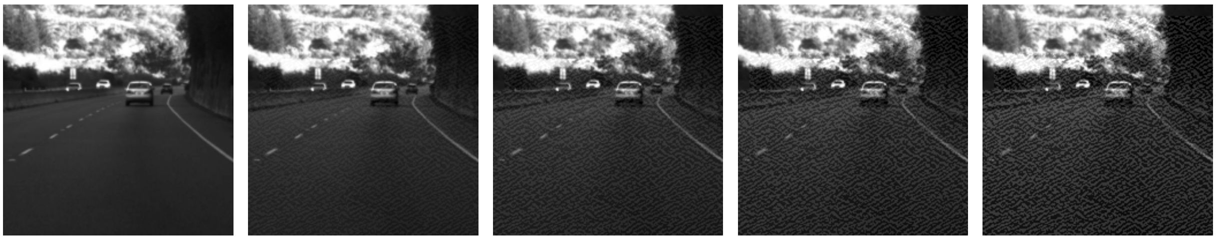


Figure 4.1: A plot depicting relationship between the MC sample count and RMSE during the MCDO_ADF model inference

inputs in an adversarial fashion to cause maximum error on the model prediction. An uncertainty estimation method needs to be capable of identifying and reporting an adversarially perturbed input data sample by producing a high value of uncertainty.

In order to evaluate the robustness of the considered uncertainty estimation methods, samples from training data distribution are perturbed using the FGSM(Fast Gradient Sign Method)[12] and fed to the trained ResNet-8 model. Changes in the estimated values of predictive uncertainty and RMSE with respect to increasing levels of perturbation(denoted by ϵ) between 0 and 0.1 for a given image are plotted in the graphs shown in the figure4.3. A set of images produced from a given image subject to different levels of adversarial perturbations is shown in the figure4.4.

It can be inferred from the plot that the increase in values of RMSE of the model using DER is consistent with the increase in perturbation levels, while a lot of variations can be observed in the case of MCDO_ADF variant. In the case of relationship between ϵ and predictive uncertainty, the predictive uncertainties estimated by DER remains almost constant in the interval between $\epsilon =$ but increase steadily then after. Similar to the case of ϵ Vs RMSE, a lot of fluctuations can be observed in the uncertainty estimates produced for different levels of ϵ.



(a) $\epsilon = 0$ (b) $\epsilon = 0.02$ (c) $\epsilon = 0.06$ (d) $\epsilon = 0.08$ (e) $\epsilon = 0.10$

Figure 4.2: Images depicting increasing levels of Adversarial noise

Function **y=f(x)**	**Metric**	**Homoscedastic** **GP**	**Heteroscedastic** **GP**	**Evidential**	**MCDO_ADF**
$y = \frac{sin(3x)}{3x}$	NLL	**-0.82**	-0.51	0.57	2.27
	RMSE	**0.07**	0.08	0.40	0.39
	EVA	**0.94**	0.91	-0.29	-0.43
$y = 0.1x^3$	NLL	**-2.44**	-1.77	-0.15	1.11
	RMSE	~**0.00**	0.03	0.23	0.11
	EVA	0.99	0.99	**0.99**	0.99
$y = -(1 + x + sin(1.2x)$	NLL	**-1.02**	1.91	6.14	>>1(182049)
	RMSE	0.61	**0.15**	2.68	0.8
	EVA	0.94	**0.99**	-0.05	0.87

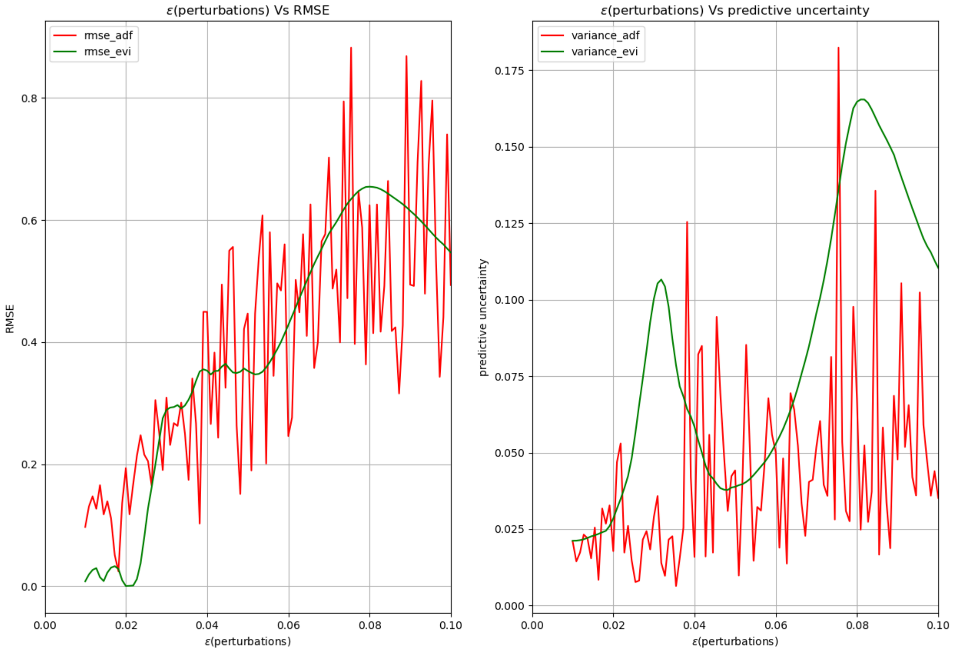
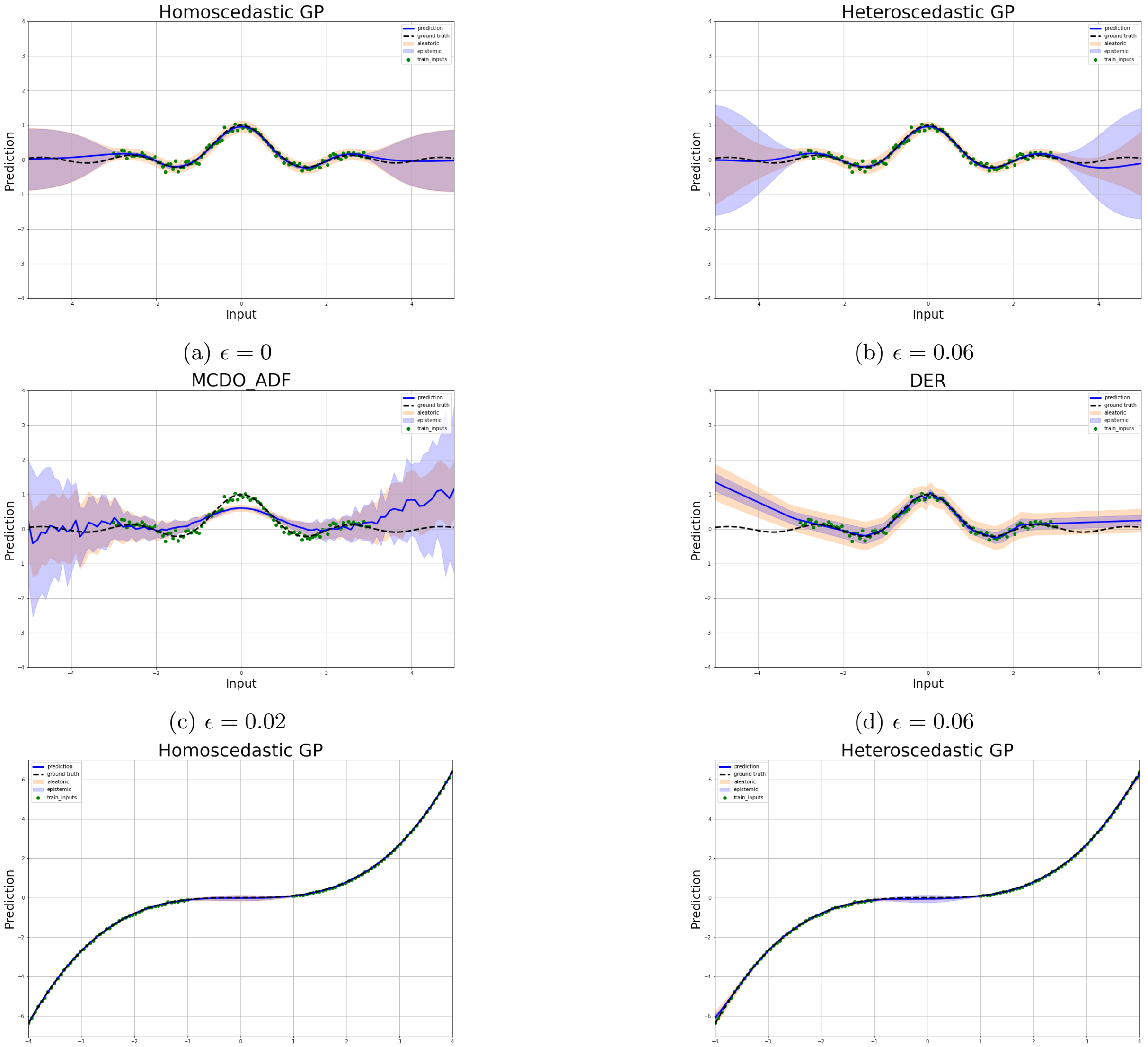


Figure 4.3: (a) Plot depicting the change in RMSE with respect to increase in adversarial perturbations for a single image (b) Plot depicting the change in predictive uncertainty with respect to increase in adversarial perturbations for a single image

Category	**Method**	**Analysis**
Aleatoric	MCDO_ADF	• The method performs well in estimating data uncertainty.The presence of glare, and poor illumination in images reported with high values show that the estim uncertainty is indeed aleatoric. • Most of the images reported with low values of aleatoric uncertainty are characte by high contrast and low noise levels. • There also exist certain over-illuminated images for which the method reports a value of data-uncertainty and is undesirable.
Aleatoric	DER/Evidential	• The method reports a high-value of aleatoric uncertainty for blurry and unclear im and is desirable. • It is interesting to note that this method reports high-values of data uncertaint images with objects such as grass that produce patterns similar to noise. • Similar to MCDO_ADF this method performs well in reporting low aleatoric u tainty for clear images except for the ones with shadows.
Epistemic	MCDO_ADF	• The method reports high values of epistemic uncertainty for both images with un lane markings(intuitively a key feature to predict steering angles) and noise ind by factors such as blur and glare. • Reporting high values of epistemic uncertainty for noisy images proves the meth ability to jointly model both the components of uncertainty. • Similar to the aleatoric case, the method reports low values of epistemic uncerta for clear images except for the ones with poor and over illumination.
Epistemic	DER/Evidential	• The method produces high values of model uncertainty for both images with un features and noise, similar to MCDO_ADF. • However, there is a difference between images reported with high values of epist uncertainty between MCDO_ADF and evidential showing the fact that they have different criteria for uncertainty estimation. • The set of images reported with low values of epistemic uncertainty have a l similarities with that of ones with low-aleatoric variances.
Total	MCDO_ADF and DER/Evidential	• The set of images reported with high/low values of predictive uncertainties by bot methods correspond to the ones with high/low values of their components(alea and epistemic). • This could be attributed to the fact that the predictive uncertainty is obtained a sum of its components in both the methods.

Table 4.2: A qualitative comparison of uncertainty estimation methods when applied to Dronet



Homoscedastic GP

(a) $\epsilon = 0$

Heteroscedastic GP

(b) $\epsilon = 0.06$

MCDO_ADF

(c) $\epsilon = 0.02$

DER

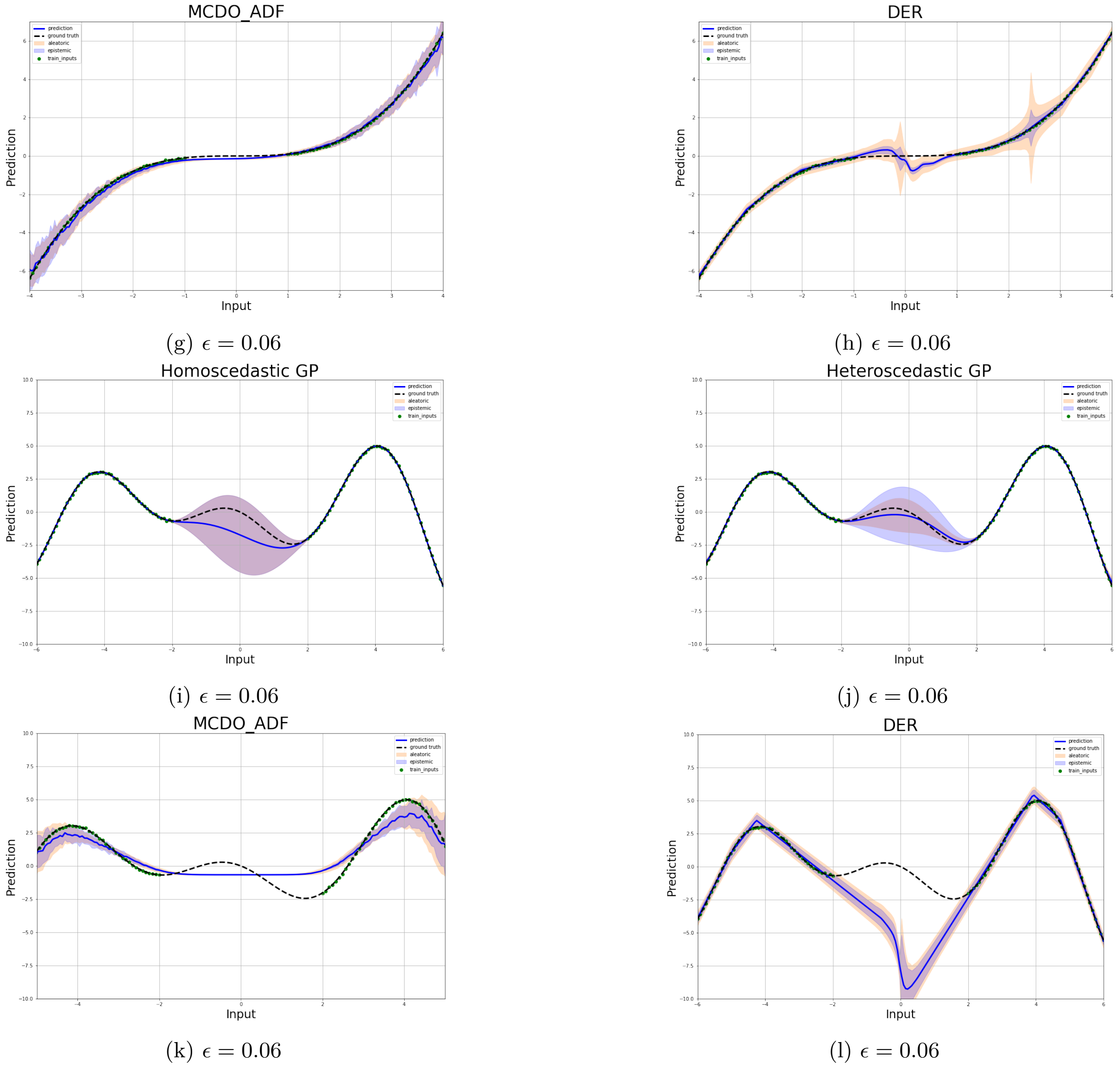
(d) $\epsilon = 0.06$

Homoscedastic GP

(e) $\epsilon = 0.06$

Heteroscedastic GP

(f) $\epsilon = 0.06$



(g) $\epsilon = 0.06$

(h) $\epsilon = 0.06$

(i) $\epsilon = 0.06$

(j) $\epsilon = 0.06$

(k) $\epsilon = 0.06$

(l) $\epsilon = 0.06$

Figure 4.4: Images depicting increasing levels of Adversarial noise

5

Conclusions

5.1 Contributions

5.2 Lessons learned

5.3 Future work

A

Design Details

Your first appendix

B

Parameters

Your second chapter appendix

References

[1] Alexander Amini, Wilko Schwarting, Ava Soleimany, and Daniela Rus. Deep evidential regression. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[2] Fabio Arnez, HuÃąscar Espinoza, Ansgar Radermacher, and FranÃğois Terrier. A comparison of uncertainty estimation approaches in deep learning components for autonomous vehicle applications. *arXiv preprint arXiv:2006.15172*, 2020.

[3] Jonathan T. Barron. A general and adaptive robust loss function. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4331–4339, 2019.

[4] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. 2006.

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

[6] Yukun Ding, Jinglan Liu, Jinjun Xiong, and Yiyu Shi. Revisiting the evaluation of uncertainty estimation and its application to explore model complexity-uncertainty trade-off. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 22–31, 2020.

[7] Lorry Dysert. Is estimate accuracy an oxymoron. *Cost engineering*, 49(1):32–36, 2007.

[8] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: representing model uncertainty in deep learning. In *ICML'16 Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, pages 1050–1059, 2016.

[9] Jochen Gast and Stefan Roth. Lightweight probabilistic deep networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3369–3378, 2018.

[10] Paul W. Goldberg, Christopher K. I. Williams, and Christopher M. Bishop. Regression with input-dependent noise: A gaussian process treatment. In *Advances in Neural Information Processing Systems 10*, volume 10, pages 493–499, 1997.

[11] Ziv Goldfeld and Yury Polyanskiy. The information bottleneck problem and its applications in machine learning. *IEEE Journal on Selected Areas in Information Theory*, 1(1):19–38, 2020.

[12] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR 2015 : International Conference on Learning Representations 2015*, 2015.

[13] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, pages 1321–1330, 2017.

[14] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision. In *NIPS'17 Proceedings of the 31st International Conference on Neural Information Processing Systems*, volume 30, pages 5580–5590, 2017.

[15] Anoop Korattikara, Vivek Rathod, Kevin Murphy, and Max Welling. Bayesian dark knowledge. In *NIPS'15 Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, volume 28, pages 3438–3446, 2015.

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

[17] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *ICLR (Poster)*, 2016.

[18] Alexey Kurakin, Ian J. Goodfellow, Samy Bengio, Yinpeng Dong, Fangzhou Liao, Ming Liang, Tianyu Pang, Jun Zhu, Xiaolin Hu, Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, Alan L. Yuille, Sangxia Huang, Yao Zhao, Yuzhe Zhao, Zhonglin Han, Junjiajia Long, Yerkebulan Berdibekov, Takuya Akiba, Seiya Tokui, and Motoki Abe. Adversarial attacks and defences competition. *arXiv preprint arXiv:1804.00097*, pages 195–231, 2018.

[19] Alex Labach, Hojjat Salehinejad, and Shahrokh Valaee. Survey of dropout methods for deep neural networks. *arXiv preprint arXiv:1904.13310*, 2019.

[20] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, volume 30, pages 6402–6413, 2017.

[21] Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S. Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as gaussian processes. In *International Conference on Learning Representations*, 2018.

[22] Antonio Loquercio, Ana I. Maqueda, Carlos R. del-Blanco, and Davide Scaramuzza. Dronet: Learning to fly by driving. *IEEE Robotics and Automation Letters*, 3(2):1088–1095, 2018.

[23] Antonio Loquercio, Mattia Segu, and Davide Scaramuzza. A general framework for uncertainty estimation in deep learning. *IEEE Robotics and Automation Letters*, 5(2):3153–3160, 2020.

[24] Andrey Malinin and Mark Gales. Predictive uncertainty estimation via prior networks. In *NIPS'18 Proceedings of the 32nd International Conference on Neural Information Processing Systems*, volume 31, pages 7047–7058, 2018.

[25] Senthil Mani, Anush Sankaran, Srikanth Tamilselvam, and Akshay Sethi. Coverage testing of deep learning models using dataset characterization. *arXiv preprint arXiv:1911.07309*, 2019.

[26] Simon J. D. Prince. *Computer Vision: Models, Learning, and Inference.* 2012.

[27] Andrew Michael Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan Daniel Tracey, and David Daniel Cox. On the information bottleneck theory of deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124020, 2019.

[28] Murat Sensoy, Lance M. Kaplan, and Melih Kandemir. Evidential deep learning to quantify classification uncertainty. In *Advances in Neural Information Processing Systems*, volume 31, pages 3179–3189, 2018.

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

[30] Aki Vehtari, Andrew Gelman, and Jonah Gabry. Practical bayesian model evaluation using leave-one-out cross-validation and waic. *Statistics and Computing*, 27(5):1413–1432, 2017.

[31] Jiayu Yao, Weiwei Pan, Soumya Ghosh, and Finale Doshi-Velez. Quality of uncertainty quantification for bayesian neural network inference. *arXiv preprint arXiv:1906.09686*, 2019.

[32] Lingxue Zhu and Nikolay Laptev. Deep and confident prediction for time series at uber. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 103–110, 2017.