Uncertainties in Deep Learning

in a nutshell

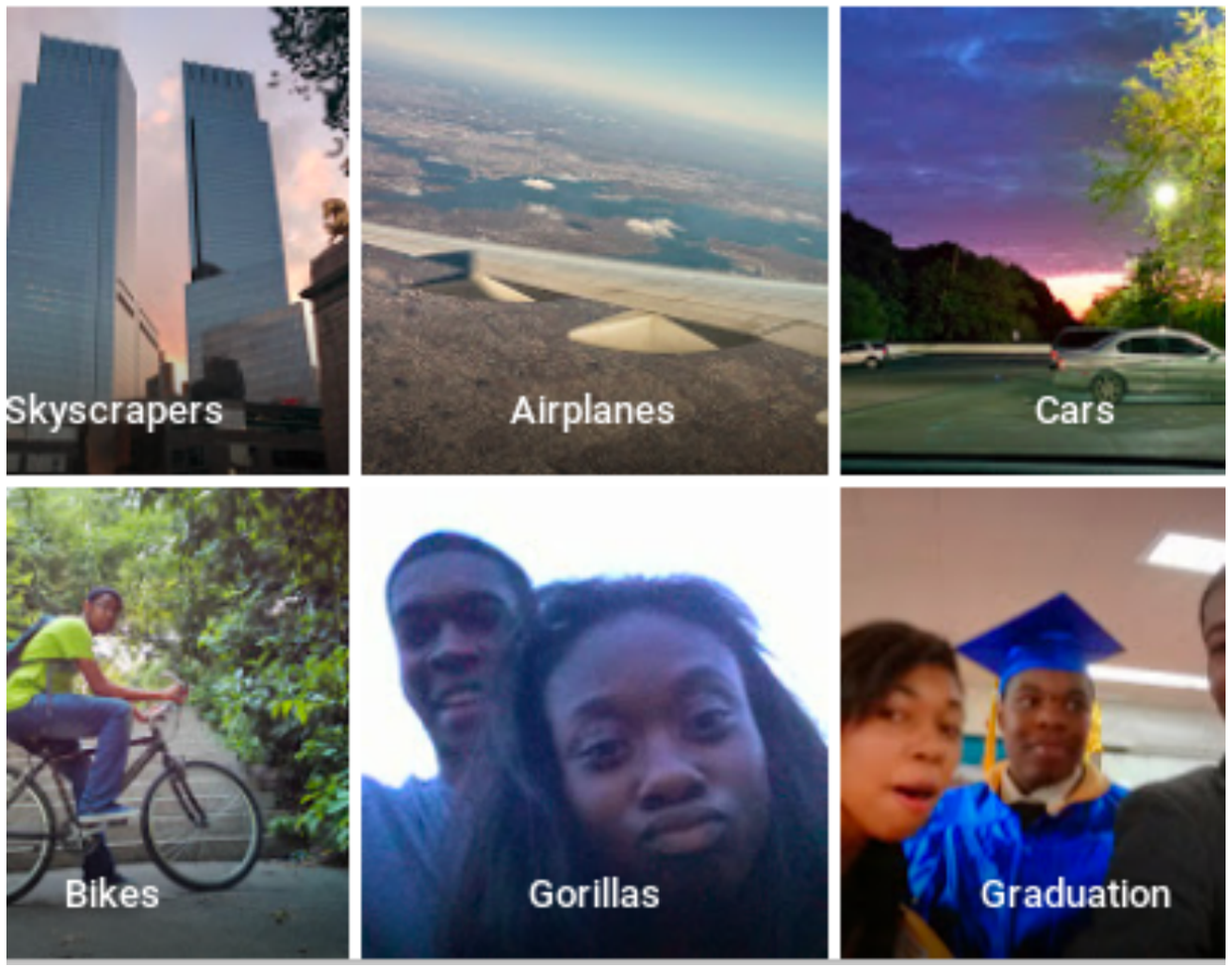
Sungjoon Choi
CPSLAB, SNU

Introduction

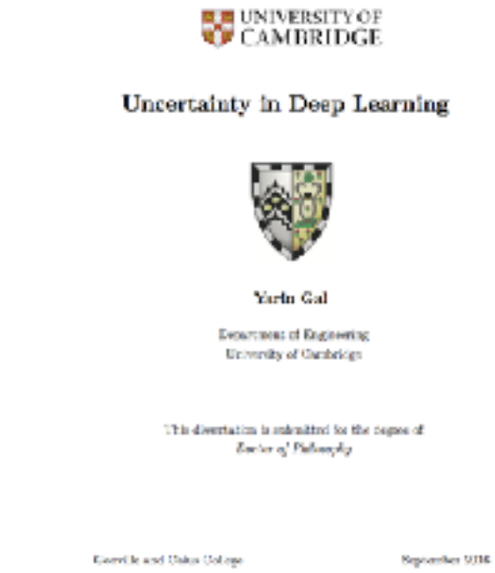
The first fatality from an assisted driving system

Introduction

Google Photos identified two black people as 'gorillas'



Contents


UNIVERSITY OF CAMBRIDGE

Uncertainty in Deep Learning

Yarin Gal

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

Kemmel & wolfson College September 2016


Under review as a conference paper at ICLR 2016

BAYESIAN UNCERTAINTY ESTIMATION FOR BATCH NORMALIZED DEEP NETWORKS

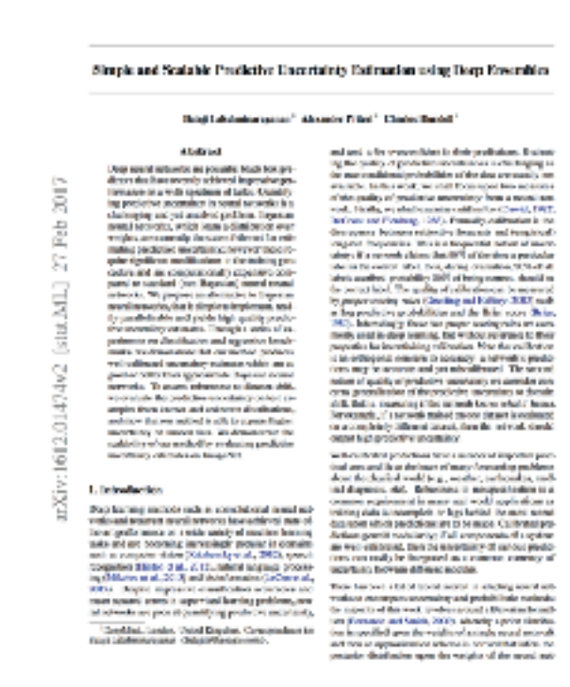
ABSTRACT

1 INTRODUCTION


Under review as a conference paper at ICLR 2017

REPRESENTING INFERENTIAL UNCERTAINTY IN DEEP NEURAL NETWORKS THROUGH SAMPLING

ABSTRACT


Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles

Abstract

1. Introduction


arXiv:1612.01474v2 [stat.ML] 27 Feb 2017


What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?

Alex Kendall
University of Cambridge
agk34@cam.ac.uk

Yarin Gal
University of Cambridge
yg279@cam.ac.uk

Abstract

1 Introduction

arXiv:1703.04977v2 [cs.CV] 5 Oct 2017


Uncertainty-Aware Reinforcement Learning for Collision Avoidance

Gregory Kahn*, Adam Villaflor*, Vitchyr Pong, Pieter Abbeel, Sergey Levine
University of California, Berkeley
{gkahn,avillaflor,vitchyr,pabbeel,svlevine}@berkeley.edu

Abstract

1 Introduction


Safe Visual Navigation via Deep Learning and Novelty Detection

Charles Richter and Nicholas Roy
Massachusetts Institute of Technology
Cambridge, MA, USA

Abstract

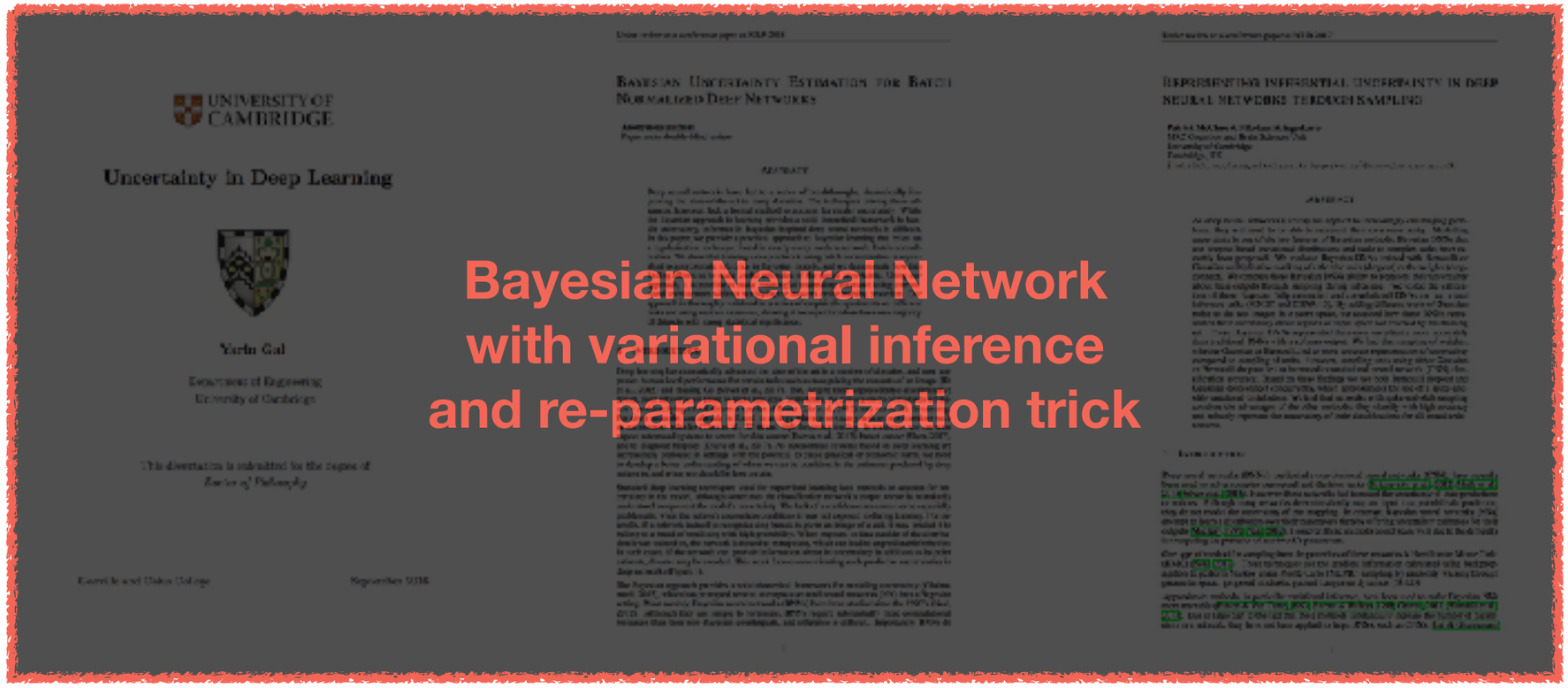

Uncertainty-Aware Learning from Demonstration Using Mixture Density Networks with Sampling-Free Variance Modeling

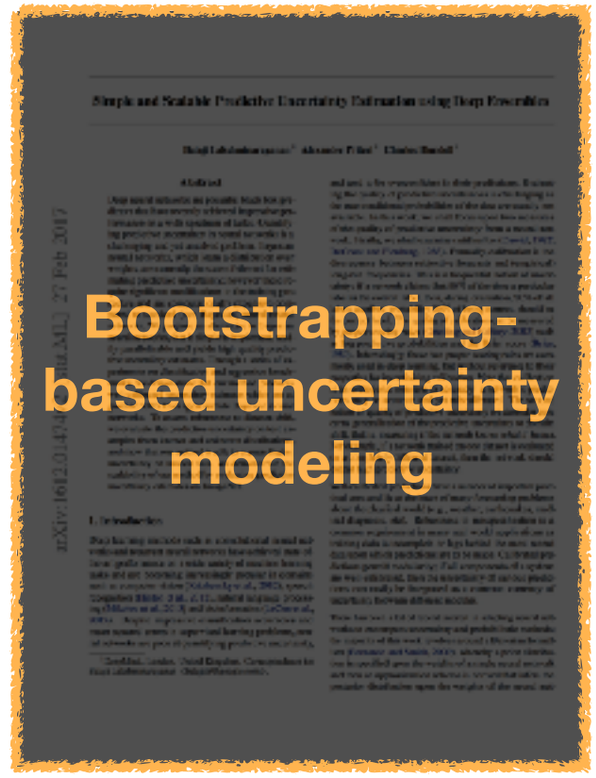
Sungjoon Choi, Kyungjae Lee, Sungbin Lim, and Songhwai Oh

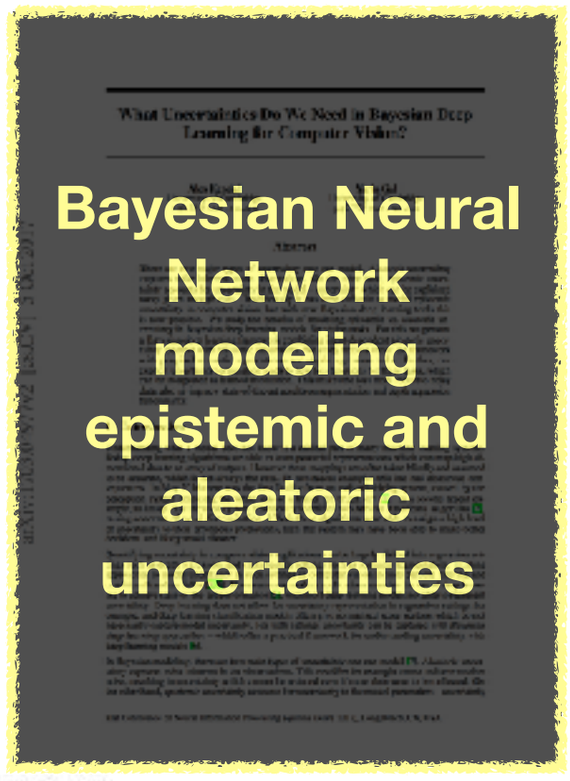
Abstract

1 Introduction

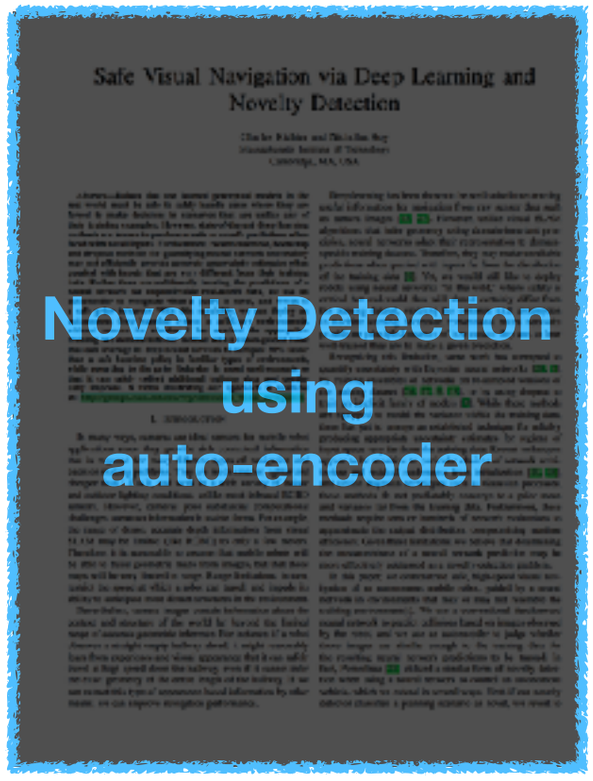
Contents

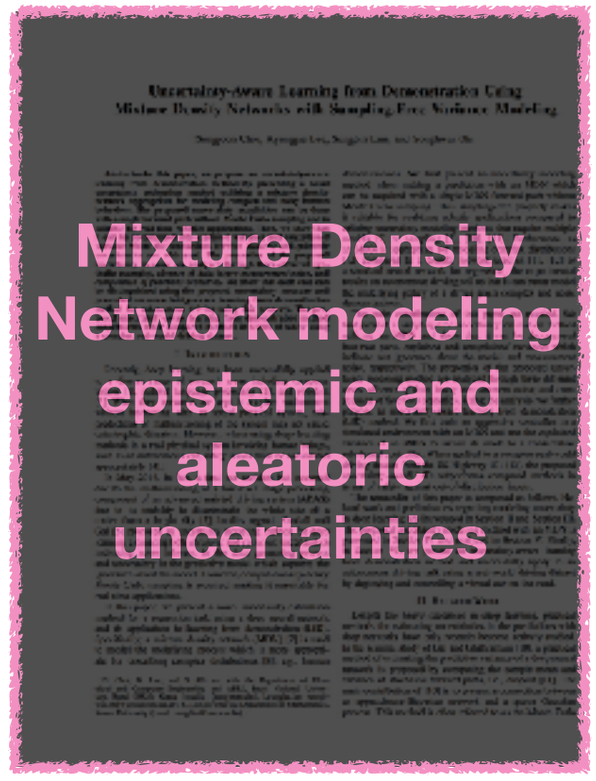

UNIVERSITY OF CAMBRIDGE
Uncertainty in Deep Learning
Yarin Gal
Department of Engineering
University of Cambridge
This dissertation is submitted for the degree of
Doctor of Philosophy
September 2016
Bayesian Uncertainty Estimation for Batch Normalized Deep Networks
Representing Inferential Uncertainty in Deep Neural Networks through Sampling
Bayesian Neural Network with variational inference and re-parametrization trick


Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles
Bootstrapping-based uncertainty modeling


What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?
Bayesian Neural Network modeling epistemic and aleatoric uncertainties


Uncertainty-Aware Reinforcement Learning for Collision Avoidance
Application to Safe RL


Safe Visual Navigation via Deep Learning and Novelty Detection
Novelty Detection using auto-encoder


Uncertainty-Aware Learning from Demonstration Using Mixture Density Networks with Sampling-Free Variance Modeling
Mixture Density Network modeling epistemic and aleatoric uncertainties

Y. Gal, Uncertainty in Deep Learning, 2016

UNIVERSITY OF CAMBRIDGE

Uncertainty in Deep Learning

Yarin Gal

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

Gonville and Caius College September 2016

Gal (2016)

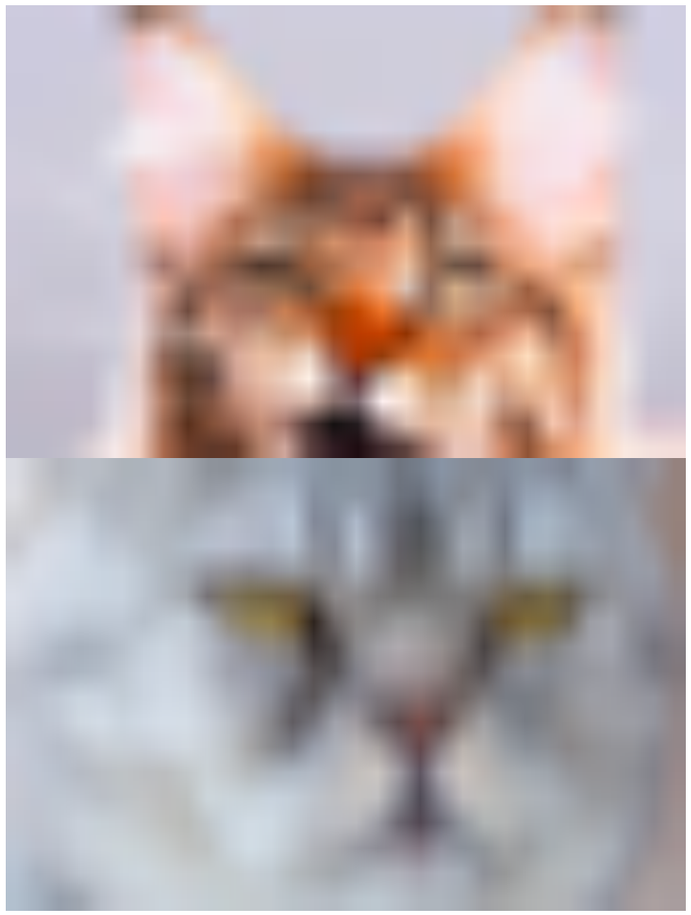
Model uncertainty

1. Given a model trained with several pictures of dog breeds, a user asks the model to decide on a dog breed using a photo of a cat.

Gal (2016)

Model uncertainty

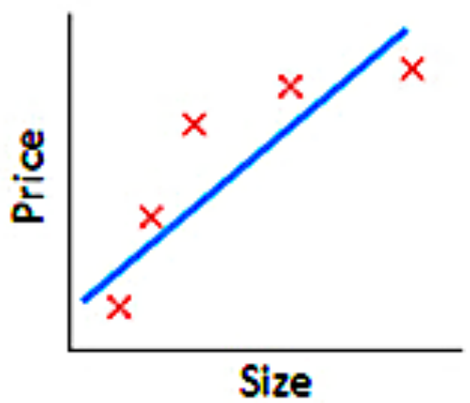
2. We have three different types of images to classify, cat, dog, and cow, where only cat images are noisy.



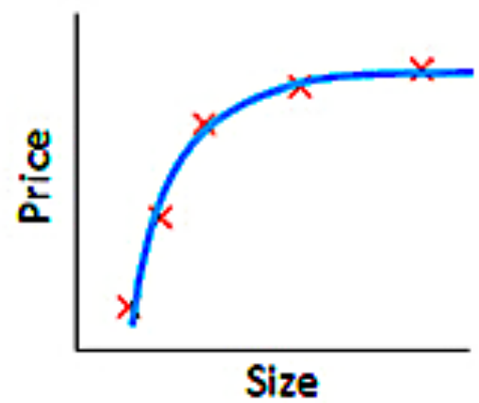
Gal (2016)

Model uncertainty

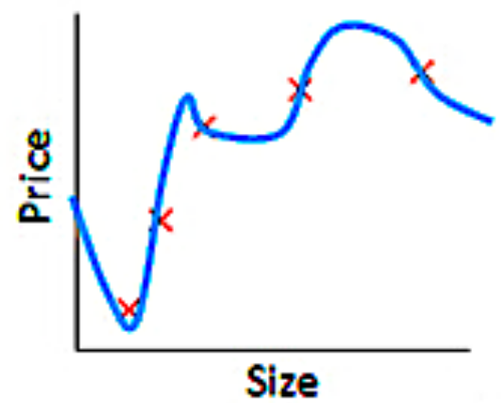
3. What is the best model parameters that best explain a given dataset? what model structure should we use?



$\theta_0 + \theta_1 x$



$\theta_0 + \theta_1 x + \theta_2 x^2$



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

Gal (2016)

Model uncertainty

1. Given a model trained with several pictures of dog breeds, a user asks the model to decide on a dog breed using a photo of a cat.

2. We have three different types of images to classify, cat, dog, and cow, where only cat images are noisy.

3. What is the best model parameters that best explain a given dataset? what model structure should we use?

Gal (2016)

Model uncertainty

1. Given a model trained with several pictures of dog breeds, a user asks the model to decide on a dog breed using a photo of a cat.

Out of distribution test data

2. We have three different types of images to classify, cat, dog, and cow, where only cat images are noisy.

Aleatoric uncertainty

3. What is the best model parameters that best explain a given dataset? what model structure should we use?

Epistemic uncertainty

Gal (2016)

Dropout as a Bayesian approximation

"We show that a neural network with arbitrary depth and non-linearities, with **dropout applied before every weight layer**, is mathematically equivalent to an approximation to a well known Bayesian model."

Gal (2016)

Dropout as a Bayesian approximation

$$\mathbb{E}[\mathbf{y}^*] \approx \frac{1}{T}\sum_{t=1}^{T}\widehat{\mathbf{y}}_t^*(\mathbf{x}^*)$$

$$\mathrm{Var}[\mathbf{y}^*] \approx \tau^{-1}\mathbf{I}_D + \frac{1}{T}\sum_{t=1}^{T}\widehat{\mathbf{y}}_t^*(\mathbf{x}^*)^T\widehat{\mathbf{y}}_t^*(\mathbf{x}^*) - \mathbb{E}[\mathbf{y}^*]^T\mathbb{E}[\mathbf{y}^*].$$

The resulting formulations are **surprisingly simple**.

Gal (2016)

Bayesian Neural Network

Posterior $p(\mathbf{w}|\mathbf{X},\mathbf{Y})$ Prior $p(\mathbf{w})$

In Bayesian inference, we aim to find a **posterior distribution** over the random variables of our interest given a prior distribution which is **intractable** in many case.

Gal (2016)

Bayesian Neural Network

Posterior $p(\mathbf{w}|\mathbf{X}, \mathbf{Y})$ Prior $p(\mathbf{w})$

Inference $p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) = \int p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{w})p(\mathbf{w}|\mathbf{X}, \mathbf{Y})d\mathbf{w}$

Note that even when given a posterior distribution, exact inference is very likely to be **intractable** as it contains integral with respect to the distribution over latent variables.

Gal (2016)

Bayesian Neural Network

Posterior $p(\mathbf{w}|\mathbf{X},\mathbf{Y})$ Prior $p(\mathbf{w})$

Inference $p(\mathbf{y}^*|\mathbf{x}^*,\mathbf{X},\mathbf{Y}) = \int p(\mathbf{y}^*|\mathbf{x}^*,\mathbf{w})p(\mathbf{w}|\mathbf{X},\mathbf{Y})d\mathbf{w}$

Variational Inference $\mathrm{KL}\left(q_\theta(\mathbf{w})||p(\mathbf{w}|\mathbf{X},\mathbf{Y})\right) = \int q_\theta(\mathbf{w}) \log \frac{q_\theta(\mathbf{w})}{p(\mathbf{w}|\mathbf{X},\mathbf{Y})} d\mathbf{w}$

Variational inference is used to approximate the (intractable) posterior distribution with (tractable) variational distribution with respect to the KL divergence.

Gal (2016)

Bayesian Neural Network

Posterior $p(\mathbf{w}|\mathbf{X}, \mathbf{Y})$ Prior $p(\mathbf{w})$

Inference $p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) = \int p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{w}) p(\mathbf{w}|\mathbf{X}, \mathbf{Y}) d\mathbf{w}$

Variational Inference $\mathrm{KL}\left(q_\theta(\mathbf{w})||p(\mathbf{w}|\mathbf{X}, \mathbf{Y})\right) = \int q_\theta(\mathbf{w}) \log \frac{q_\theta(\mathbf{w})}{p(\mathbf{w}|\mathbf{X}, \mathbf{Y})} d\mathbf{w}$

ELBO $\int q_\theta(\mathbf{w}) \log p(\mathbf{Y}|\mathbf{X}, \mathbf{w}) d\mathbf{w} - \mathrm{KL}(q_\theta(\mathbf{w})||p(\mathbf{w}))$

Minimizing the KL divergence is equivalent to maximizing the **evidence lower bound (ELBO)** which also contains the integral with respect to the distribution over latent variables.

Gal (2016)

Bayesian Neural Network

Posterior $p(\mathbf{w}|\mathbf{X},\mathbf{Y})$ Prior $p(\mathbf{w})$

Inference $p(\mathbf{y}^*|\mathbf{x}^*,\mathbf{X},\mathbf{Y}) = \int p(\mathbf{y}^*|\mathbf{x}^*,\mathbf{w})p(\mathbf{w}|\mathbf{X},\mathbf{Y})d\mathbf{w}$

Variational Inference $\mathrm{KL}\left(q_\theta(\mathbf{w})||p(\mathbf{w}|\mathbf{X},\mathbf{Y})\right) = \int q_\theta(\mathbf{w}) \log \frac{q_\theta(\mathbf{w})}{p(\mathbf{w}|\mathbf{X},\mathbf{Y})} d\mathbf{w}$

ELBO $\int q_\theta(\mathbf{w}) \log p(\mathbf{Y}|\mathbf{X},\mathbf{w}) d\mathbf{w} - \mathrm{KL}(q_\theta(\mathbf{w})||p(\mathbf{w}))$

Instead of the posterior distribution, we only need a **likelihood** to compute the ELBO.

Gal (2016)

Bayesian Neural Network

Posterior $p(\mathbf{w}|\mathbf{X},\mathbf{Y})$ Prior $p(\mathbf{w})$

Inference $p(\mathbf{y}^*|\mathbf{x}^*,\mathbf{X},\mathbf{Y}) = \int p(\mathbf{y}^*|\mathbf{x}^*,\mathbf{w})p(\mathbf{w}|\mathbf{X},\mathbf{Y})d\mathbf{w}$

Variational Inference $\mathrm{KL}\left(q_\theta(\mathbf{w})||p(\mathbf{w}|\mathbf{X},\mathbf{Y})\right) = \int q_\theta(\mathbf{w}) \log \frac{q_\theta(\mathbf{w})}{p(\mathbf{w}|\mathbf{X},\mathbf{Y})} d\mathbf{w}$

ELBO $\int q_\theta(\mathbf{w}) \log p(\mathbf{Y}|\mathbf{X},\mathbf{w}) d\mathbf{w} - \mathrm{KL}(q_\theta(\mathbf{w})||p(\mathbf{w}))$

ELBO (reparametrization) $\int p(\epsilon) \log p(\mathbf{Y}|\mathbf{X},\mathbf{w}) d\epsilon - \mathrm{KL}(q_\theta(\mathbf{w})||p(\mathbf{w}))$

$\mathbf{w} = g(\theta,\epsilon)$

Gal (2016)

Bayesian Neural Network

ELBO $$\int q_\theta(\mathbf{w}) \log p(\mathbf{Y}|\mathbf{X}, \mathbf{w}) d\mathbf{w} - \mathrm{KL}(q_\theta(\mathbf{w})||p(\mathbf{w}))$$

$$\mathbf{w} = g(\theta, \epsilon)$$ (**Re-parametrization** trick)

Re-parametrized ELBO $$\int p(\epsilon) \log p(\mathbf{Y}|\mathbf{X}, \mathbf{w}) d\epsilon - \mathrm{KL}(q_\theta(\mathbf{w})||p(\mathbf{w}))$$

Gal (2016)

Gaussian process approximation

$$\mathbf{K}(\mathbf{x},\mathbf{y}) = \int p(\mathbf{w})p(b)\sigma(\mathbf{w}^T\mathbf{x}+b)\sigma(\mathbf{w}^T\mathbf{y}+b)\mathbf{dw}db$$

↓ **MC approximation**

$$\widehat{\mathbf{K}}(\mathbf{x},\mathbf{y}) = \frac{1}{K}\sum_{k=1}^{K}\sigma(\mathbf{w}_k^T\mathbf{x}+b_k)\sigma(\mathbf{w}_k^T\mathbf{y}+b_k)$$

↓ **Apply to Gaussian processes**

$$\mathbf{w}_k \sim p(\mathbf{w}),\ b_k \sim p(b),$$

$$\mathbf{W}_1 = [\mathbf{w}_k]_{k=1}^K, \mathbf{b} = [b_k]_{k=1}^K$$

$$\widehat{\mathbf{K}}(\mathbf{x},\mathbf{y}) = \frac{1}{K}\sum_{k=1}^{K}\sigma(\mathbf{w}_k^T\mathbf{x}+b_k)\sigma(\mathbf{w}_k^T\mathbf{y}+b_k)$$

$$\mathbf{F} \mid \mathbf{X}, \mathbf{W}_1, \mathbf{b} \sim \mathcal{N}(\mathbf{0}, \widehat{\mathbf{K}}(\mathbf{X},\mathbf{X}))$$

$$\mathbf{Y} \mid \mathbf{F} \sim \mathcal{N}(\mathbf{F}, \tau^{-1}\mathbf{I}_N),$$

↓ **GP Marginal likelihood**

$$p(\mathbf{Y}|\mathbf{X}) = \int p(\mathbf{Y}|\mathbf{X},\mathbf{W}_1,\mathbf{W}_2,\mathbf{b})p(\mathbf{W}_1)p(\mathbf{W}_2)p(\mathbf{b})$$

Gal (2016)

Bayesian Neural Network with dropout

$$\widehat{\mathcal{L}}_{\text{dropout}}(\mathbf{M}_1, \mathbf{M}_2, \mathbf{b}) = -\frac{1}{M\tau}\sum_{i\in S}\log p(\mathbf{y}_i|\mathbf{f}^{g(\theta,\widehat{\epsilon_i})}(\mathbf{x})) + \lambda_1||\mathbf{M}_1||^2 + \lambda_2||\mathbf{M}_2||^2 + \lambda_3||\mathbf{b}||^2$$

Gal (2016)

Bayesian Neural Network with dropout

$$\widehat{\mathcal{L}}_{\text{dropout}}(\mathbf{M}_1, \mathbf{M}_2, \mathbf{b}) = -\frac{1}{M\tau}\sum_{i\in S} \log p(\mathbf{y}_i|\mathbf{f}^{g(\theta,\widehat{\epsilon_i})}(\mathbf{x})) + \lambda_1||\mathbf{M}_1||^2 + \lambda_2||\mathbf{M}_2||^2 + \lambda_3||\mathbf{b}||^2$$

Likelihood

$$p(\mathbf{y}|\mathbf{f}^{g(\theta,\hat{\epsilon})}(\mathbf{x})) = \mathcal{N}(\mathbf{y}; \hat{\mathbf{y}}_\theta(\mathbf{x}), \tau^{-1}\mathbf{I}_D)$$

Gal (2016)

Bayesian Neural Network with dropout

$$\widehat{\mathcal{L}}_{\text{dropout}}(\mathbf{M}_1, \mathbf{M}_2, \mathbf{b}) = -\frac{1}{M\tau}\sum_{i\in S}\log p(\mathbf{y}_i|\mathbf{f}^{g(\theta,\widehat{\epsilon_i})}(\mathbf{x})) + \lambda_1||\mathbf{M}_1||^2 + \lambda_2||\mathbf{M}_2||^2 + \lambda_3||\mathbf{b}||^2$$

Re-parametrized ELBO $\int p(\epsilon)\log p(\mathbf{Y}|\mathbf{X},\mathbf{w})d\epsilon - \mathrm{KL}(q_\theta(\mathbf{w})||p(\mathbf{w}))$

Re-parametrized likelihood **Prior**

Gal (2016)

Bayesian Neural Network with dropout

$$\widehat{\mathcal{L}}_{\text{dropout}}(\mathbf{M}_1, \mathbf{M}_2, \mathbf{b}) = -\frac{1}{M\tau}\sum_{i\in S}\log p(\mathbf{y}_i|\mathbf{f}^{g(\theta,\widehat{\epsilon}_i)}(\mathbf{x})) + \lambda_1||\mathbf{M}_1||^2 + \lambda_2||\mathbf{M}_2||^2 + \lambda_3||\mathbf{b}||^2$$

Algorithm 2 Optimisation of a neural network with dropout

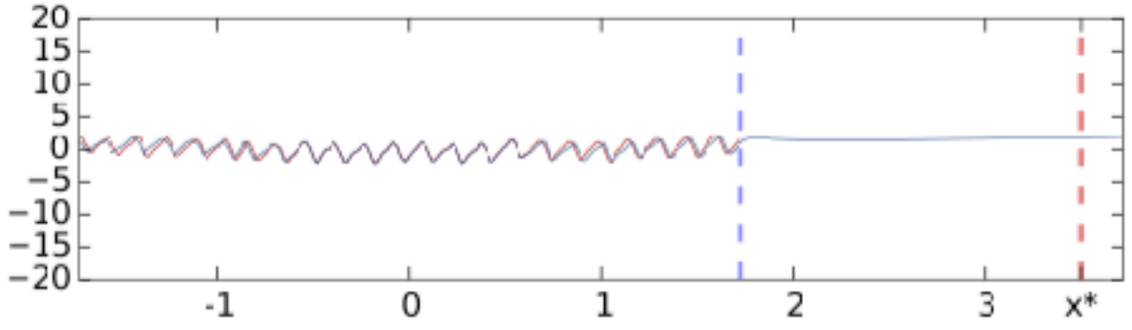
1: Given dataset $\mathbf{X}, \mathbf{Y}$,
2: Define learning rate schedule η,
3: Initialise parameters θ randomly.
4: **repeat**
5: Sample M random variables $\widehat{\boldsymbol{\epsilon}}_i \sim p(\boldsymbol{\epsilon})$, S a random subset of $\{1, .., N\}$ of size M.
6: Calculate derivative w.r.t. θ:

$$\widehat{\Delta\theta} \leftarrow -\frac{1}{M\tau}\sum_{i\in S}\frac{\partial}{\partial\theta}\log p(\mathbf{y}_i|\mathbf{f}^{g(\theta,\widehat{\epsilon}_i)}(\mathbf{x})) + \frac{\partial}{\partial\theta}\left(\lambda_1||\mathbf{M}_1||^2 + \lambda_2||\mathbf{M}_2||^2 + \lambda_3||\mathbf{b}||^2\right).$$

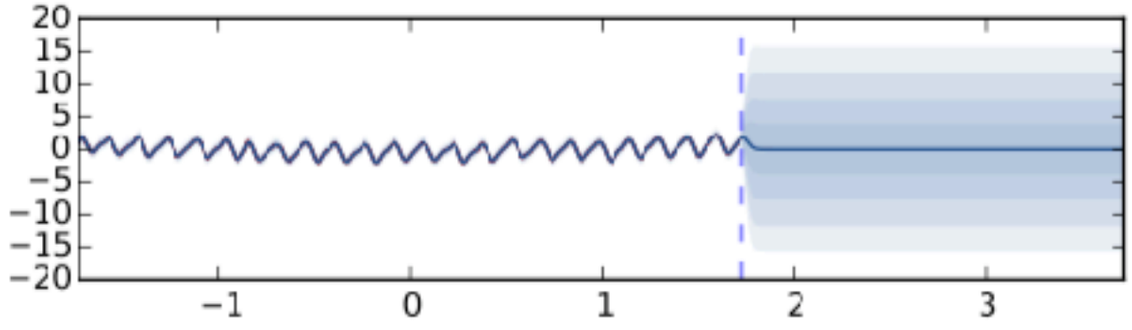
7: Update θ:
$\theta \leftarrow \theta + \eta\widehat{\Delta\theta}$.
8: **until** θ has converged.

Gal (2016)

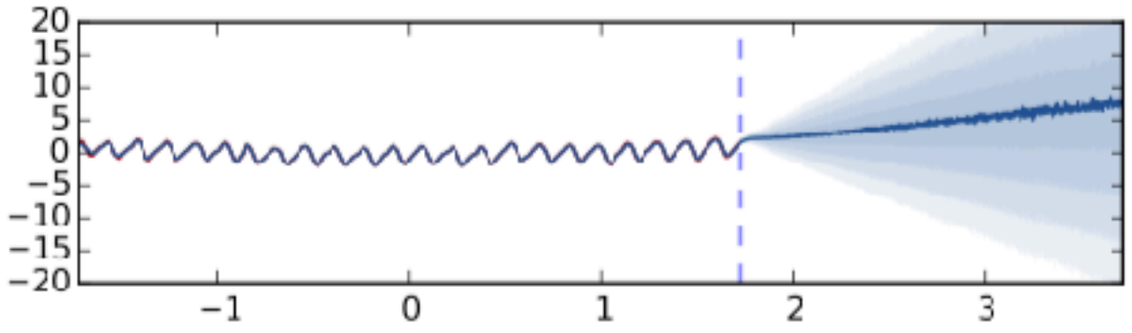
Predictive mean and uncertainties



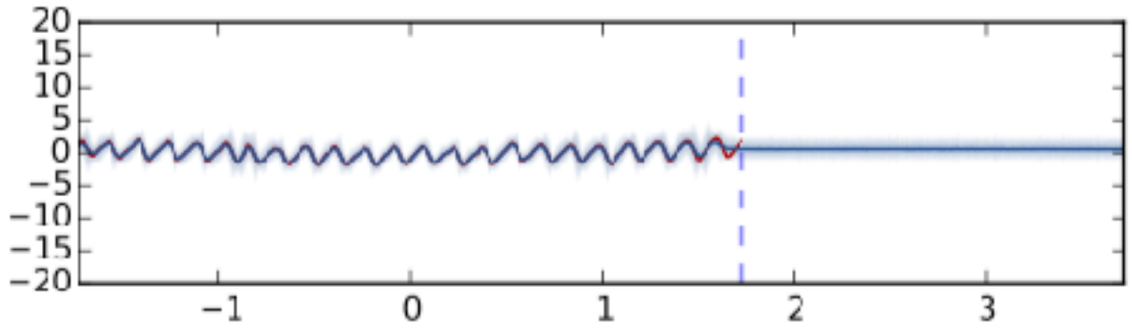
(a) Standard dropout with weight averaging



(b) Gaussian process with SE cov. function



(c) MC dropout with ReLU non-linearities



(d) MC dropout with TanH non-linearities

Gal (2016)

Exploration in deep reinforcement learning

With our uncertainty estimates given by a **dropout Q-network** we can use techniques such as **Thompson sampling** to converge faster than with **epsilon greedy** while avoiding over-fitting

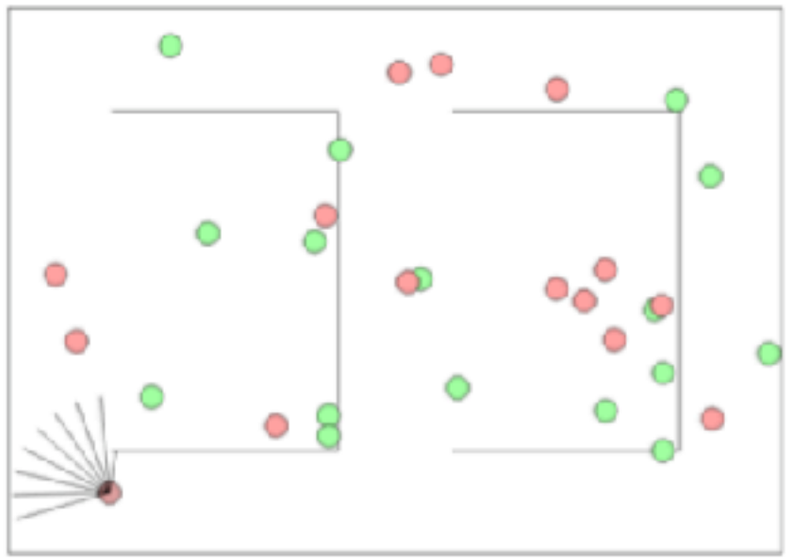


Fig. 5.3 Depiction of the reinforcement learning problem used in the experiments. The agent is in the lower left part of the maze, facing north-west.

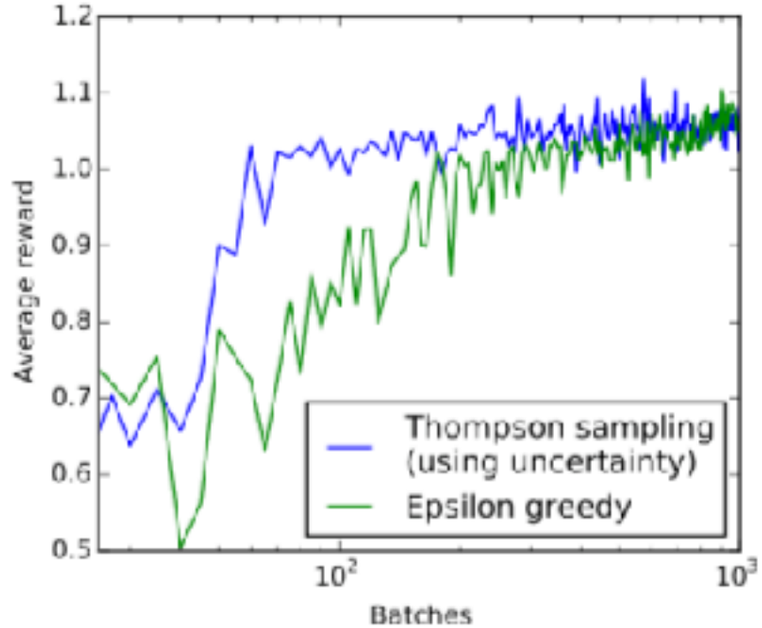


Fig. 5.4 **Log plot of average reward** obtained by both epsilon greedy (in green) and our approach (in blue), as a function of the number of batches.

Gal (2016)

Deep insights

Some “tricks of the trade” to get good predictive uncertainty estimates

First, it seems that “**over-parametrized**” models result in better uncertainty estimates than smaller models. Models with a large number of parameters can capture a larger class of functions, leading to more ways of explaining the data, and as a result larger uncertainty estimates further from the data.

The **dropout probability** is important as well, with larger models requiring a larger dropout probability: varying the dropout probability p we have that large models push p towards 0.5. For a fixed model size K, smaller probabilities p result in decreasing predictive uncertainty.

Gal (2016)

Deep insights

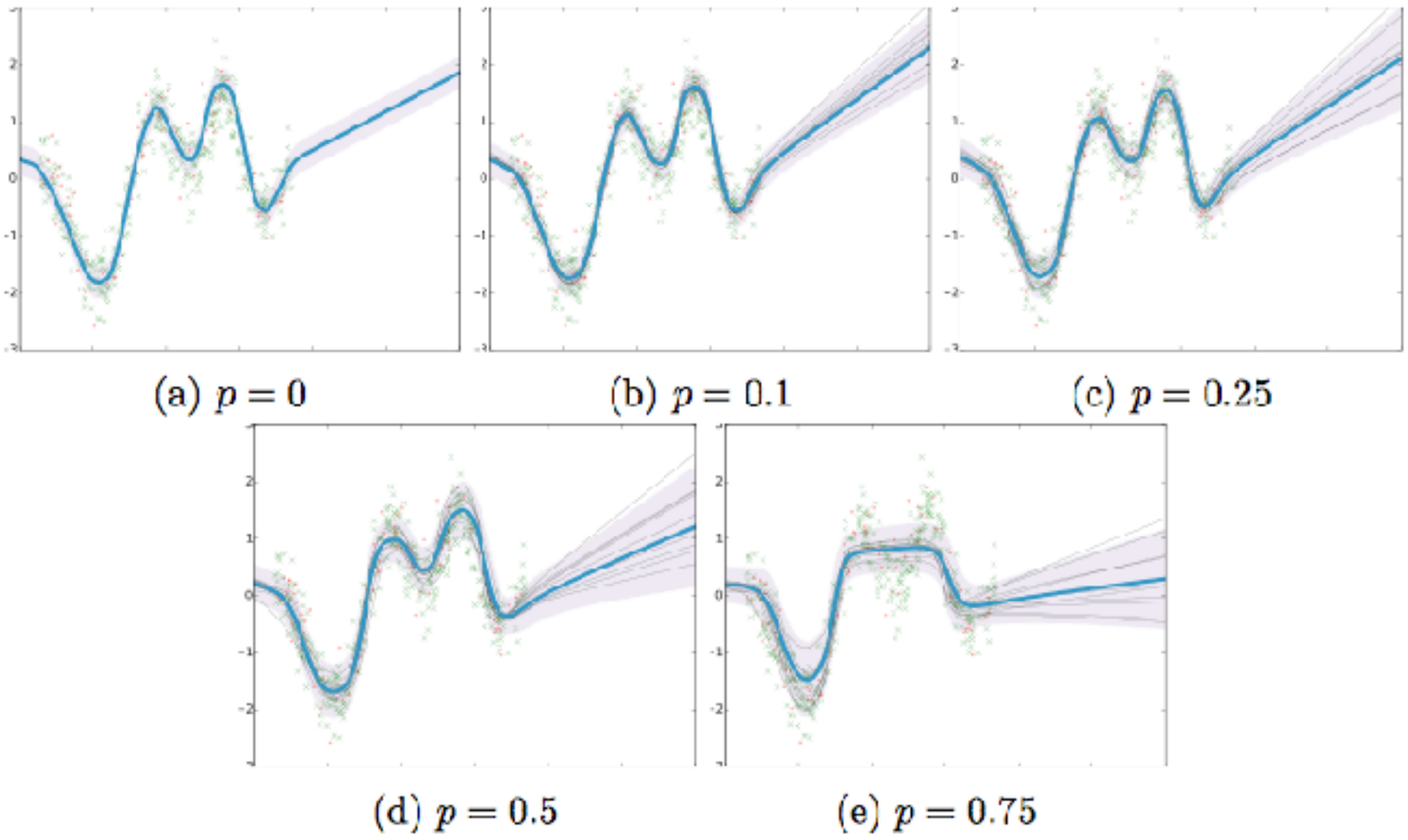
<u>What determines the uncertainty?</u>

Predictive uncertainty is determined through a combination of **model structure**, **model prior**, and **approximating distributions**.

Gal (2016)

Deep insights

Model precision?



(a) $p = 0$ (b) $p = 0.1$ (c) $p = 0.25$

(d) $p = 0.5$ (e) $p = 0.75$

P. McClure, Representing Inferential Uncertainty in Deep Neural Networks Through Sampling, 2017

Under review as a conference paper at ICLR 2017

REPRESENTING INFERENTIAL UNCERTAINTY IN DEEP NEURAL NETWORKS THROUGH SAMPLING

Patrick McClure & Nikolaus Kriegeskorte
MRC Cognition and Brain Sciences Unit
University of Cambridge
Cambridge, UK
{patrick.mcclure,nikolaus.kriegeskorte}@mrc-cbu.cam.ac.uk

ABSTRACT

As deep neural networks (DNNs) are applied to increasingly challenging problems, they will need to be able to represent their own uncertainty. Modelling uncertainty is one of the key features of Bayesian methods. Bayesian DNNs that use dropout-like variational distributions and make predictions by averaging Monte Carlo samples have recently been proposed. We evaluate Bayesian DNNs trained with Bernoulli or Gaussian multiplicative masking of either the units (dropout) or the weights (dropconnect). We compare these Bayesian DNNs ability to represent their uncertainty about their outputs through sampling during inference. We tested the calibration of these Bayesian fully connected and convolutional DNNs on two visual inference tasks (MNIST and CIFAR-10). By adding different levels of Gaussian noise to the test images in z-score space, we assessed how these DNNs represented their uncertainty about regions of input space not covered by the training set. These Bayesian DNNs represented their own uncertainty more accurately than traditional DNNs with a softmax output. We find that sampling of weights, whether Gaussian or Bernoulli, led to more accurate representation of uncertainty compared to sampling of units. However, sampling units using either Gaussian or Bernoulli dropout led to increased convolutional neural network (CNN) classification accuracy. Based on these findings we use both Bernoulli dropout and Gaussian dropconnect concurrently, which approximates the use of a spike-and-slab variational distribution. We find that networks with spike-and-slab sampling combine the advantages of the other methods: they classify with high accuracy and robustly represent the uncertainty of their classifications for all tested architectures.

1 INTRODUCTION

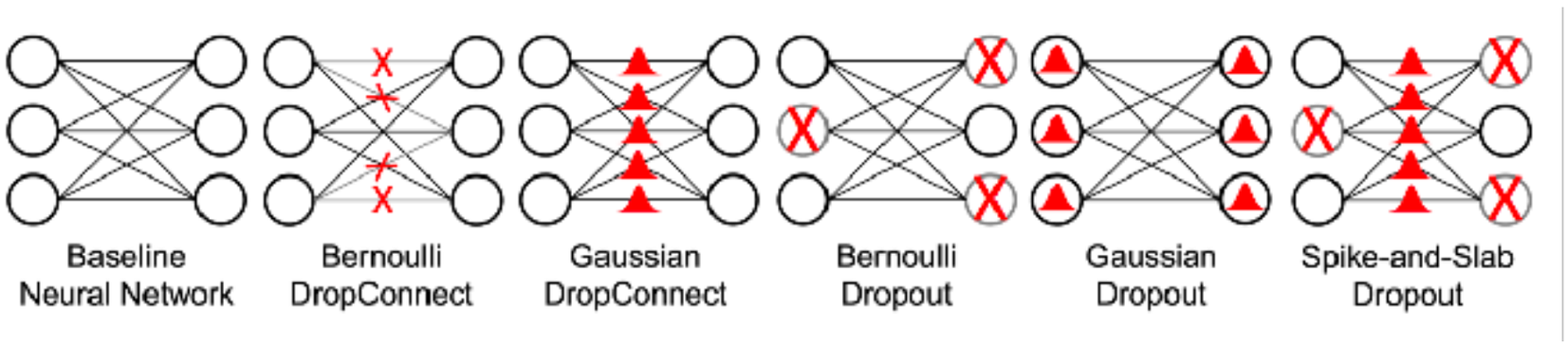
Deep neural networks (DNNs), particularly convolutional neural networks (CNN), have recently been used to solve complex perceptual and decision tasks (Krizhevsky et al., 2012; Mnih et al., 2015; Silver et al., 2016). However, these networks fail to model the uncertainty of their predictions or actions. Although many networks deterministically map an input to a probabilistic prediction, they do not model the uncertainty of that mapping. In contrast, Bayesian neural networks (BNNs) attempt to learn a distribution over their parameters thereby offering uncertainty estimates for their outputs (MacKay, 1992; Neal, 2012). However, these methods do not scale well due to the difficulty in computing the posterior of a network's parameters.

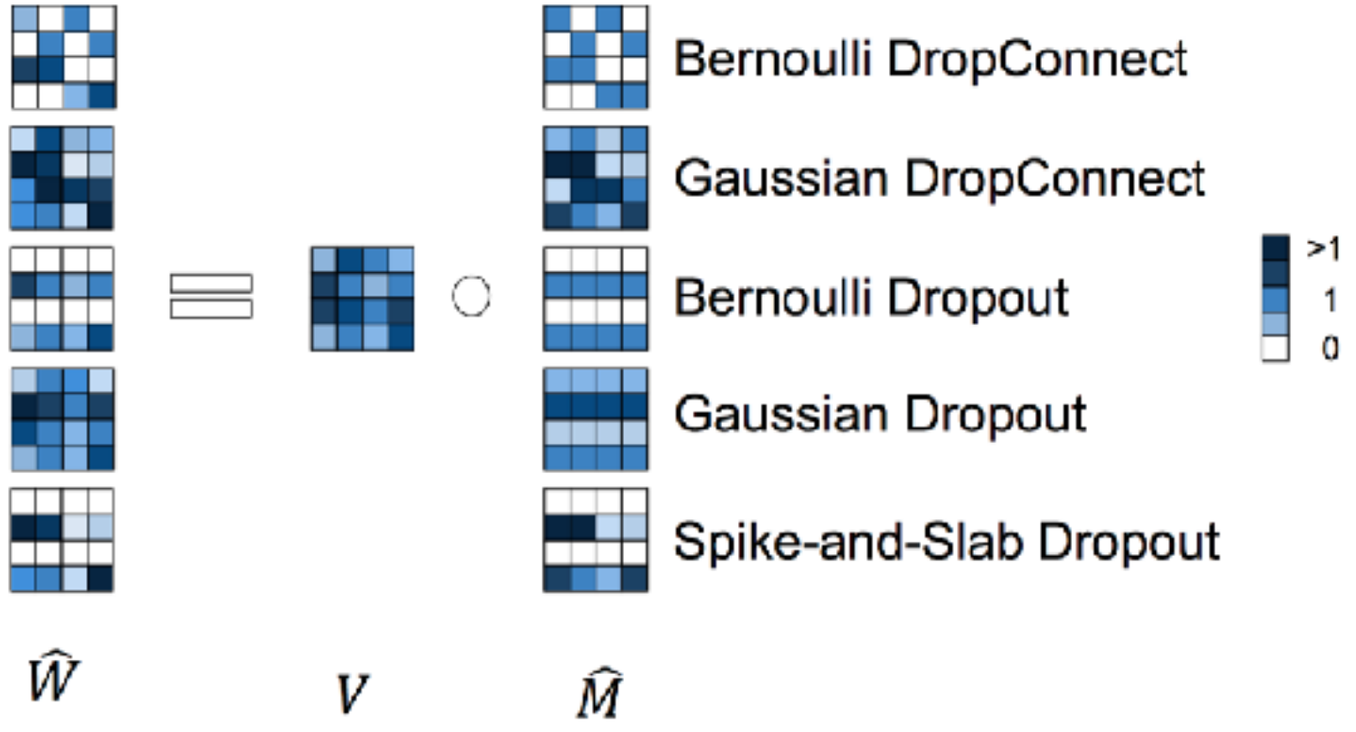
One type of method for sampling from the posteriors of these networks is Hamiltonian Monte Carlo (HMC) (Neal, 2012). These techniques use the gradient information calculated using backpropagation to perform Markov chain Monte Carlo (MCMC) sampling by randomly walking through parameter space. Proposed stochastic gradient Langevin dynamics (SGLD)

Approximate methods, in particular variational inference, have been used to make Bayesian NNs more tractable (Hinton & Van Camp, 1993; Barber & Bishop, 1998; Graves, 2011; Blundell et al., 2015). Due in large part to the fact that these methods substantially increase the number of parameters in a network, they have not been applied to large DNNs, such as CNNs. (Gal & Ghahramani

McClure & Kriegeskorte (2017)

Different variational distributions





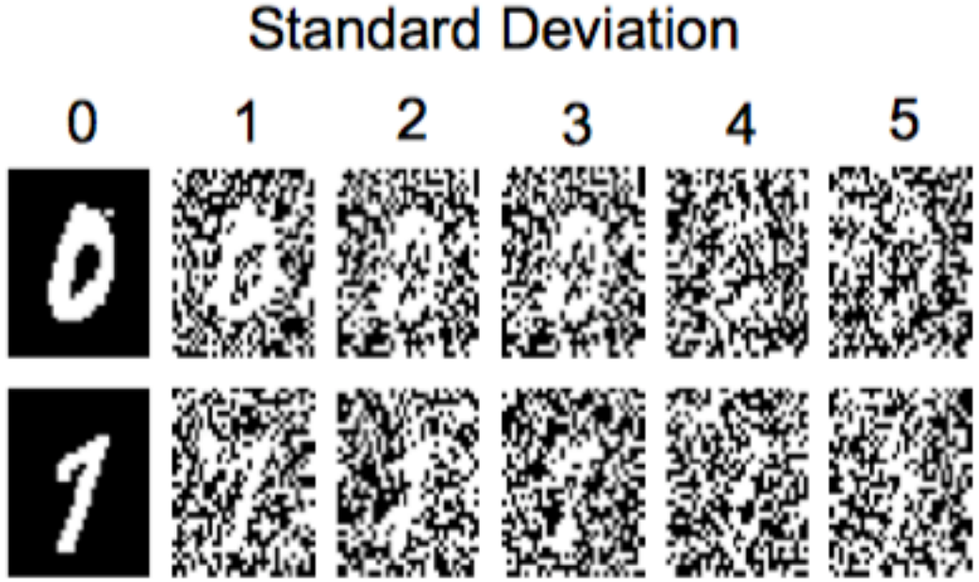
McClure & Kriegeskorte (2017)

Different variational distributions

- **Bernoulli distributions**: return 1 with probability p and 0 with probability (1-p)
- **Bernoulli drop-connect**: each element of the mask is sampled independently with Bernoulli distributions
- **Gaussian drop-connect**: the elements of the mask are sample from a normal distribution centered at variational parameter
- **Gaussian dropout**: each element in a row of the mask has the same value
- **Spike-and-slap dropout**: A normalized linear combination of a ‘spike’ of probability mass at zero and a ‘slab’ consisting of a Gaussian distribution where it returns a 0 with probability p or a random sample from a Gaussian distribution with probability (1-p).

McClure & Kriegeskorte (2017)

Results on MNIST



Test by varying the measurement noise of an image.

McClure & Kriegeskorte (2017)

Results on MNIST

Investigate how using MC sampling affects a DNN's ability to represent the uncertainty.

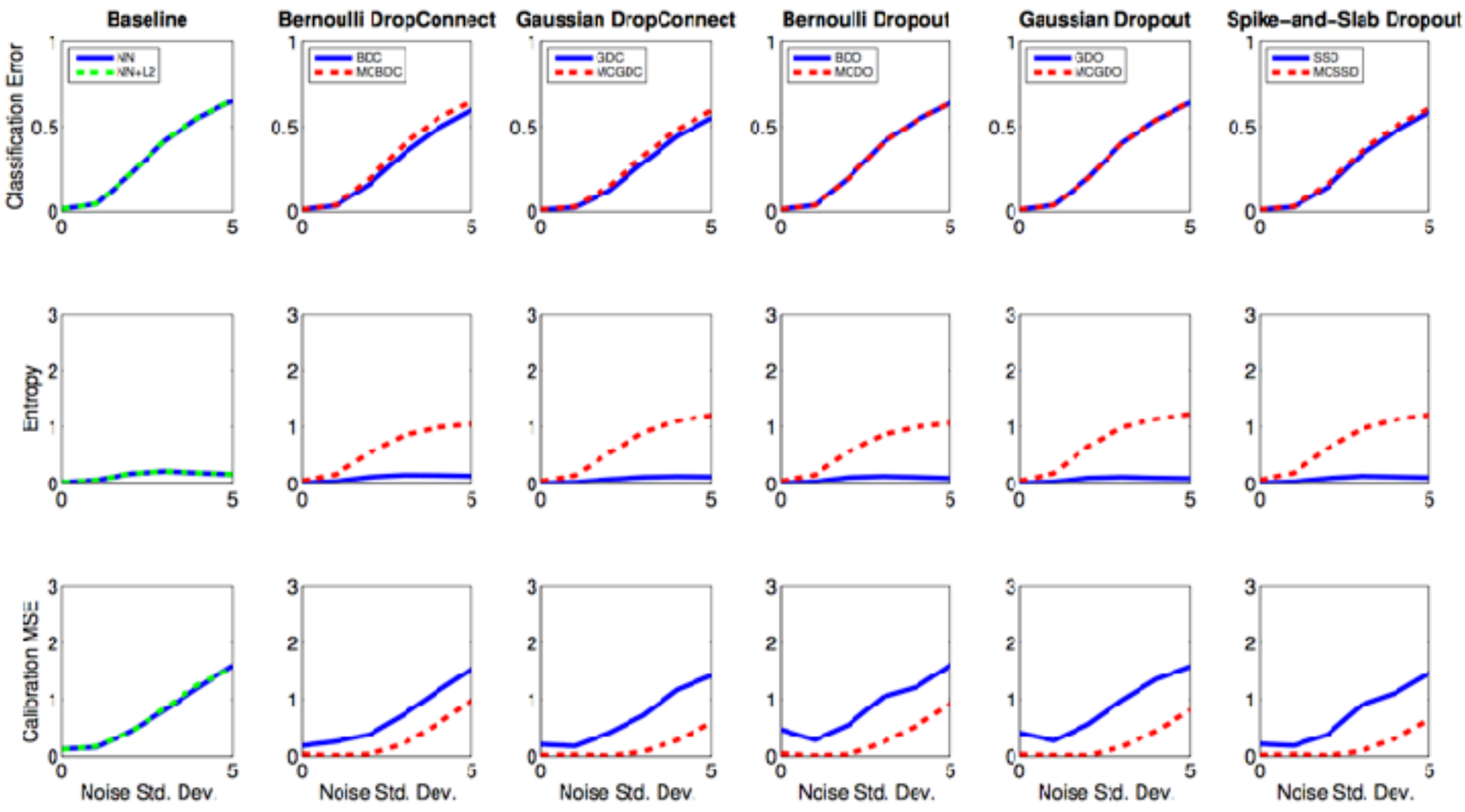
1. **no sampling** (baseline DNN and DNN with L2-regularization)
2. **sampling during training** (dropout and drop-connect)
3. **sampling during training and inference** (MC dropout and MC drop-connect)

Computed following statistics:

1. **classification error**:
2. **entropy**: how the probability mass is located.
3. **calibration (error)**: how well the probability distribution models own uncertainty.

McClure & Kriegeskorte (2017)

Results on MNIST

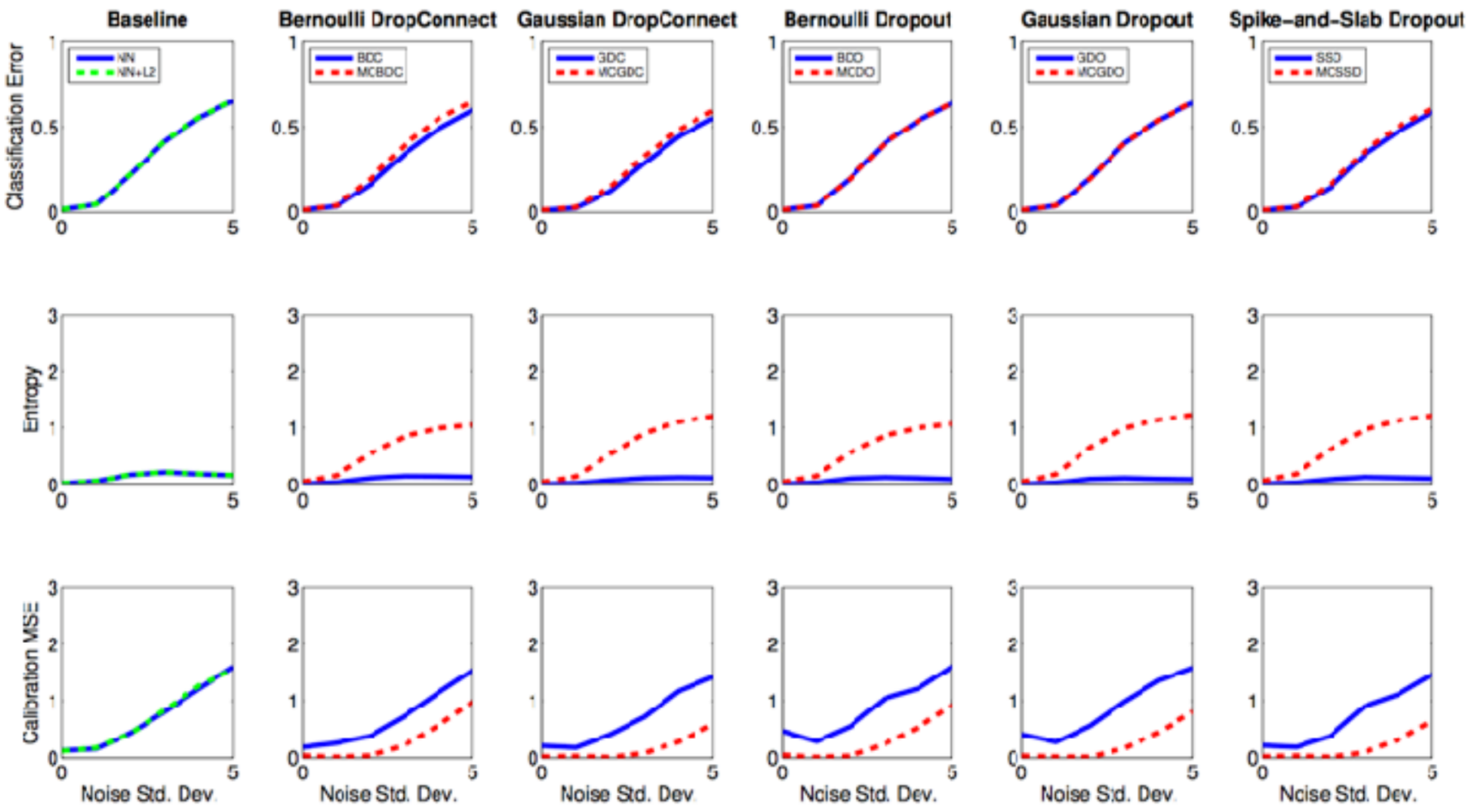


Computed following statistics:

1. **classification error**:
2. **entropy**: how the probability mass is located.
3. **calibration (error)**: how well the probability distribution models own uncertainty.

McClure & Kriegeskorte (2017)

Results on MNIST

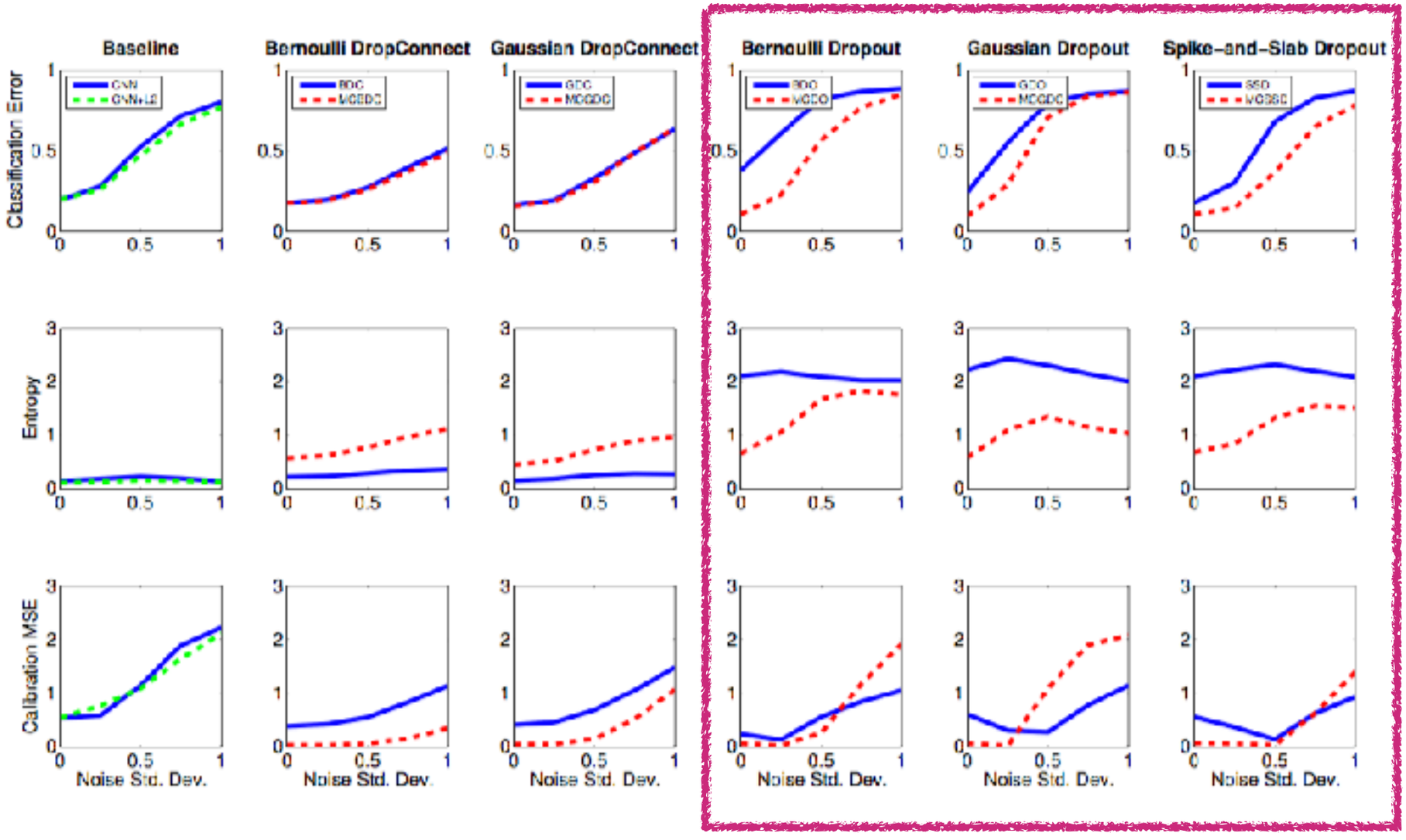


What we can see is:

1. **classification error**: performance does **not** change!
2. **entropy**: sampling during inference enables modeling uncertainty information.
3. **calibration (error)**: sampling during inference is better w.r.t. modeling uncertainty.

McClure & Kriegeskorte (2017)

Results on CIFAR



What we can see is:

1. **classification error**: performance does **not** change!
2. **entropy**: sampling during inference enables modeling uncertainty information.
3. **calibration (error)**: sampling during inference is better w.r.t. modeling uncertainty.

Anonymous, Bayesian Uncertainty Estimation for Batch Normalized Deep Networks, 2018

BAYESIAN UNCERTAINTY ESTIMATION FOR BATCH NORMALIZED DEEP NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep neural networks have led to a series of breakthroughs, dramatically improving the state-of-the-art in many domains. The techniques driving these advances, however, lack a formal method to account for model uncertainty. While the Bayesian approach to learning provides a solid theoretical framework to handle uncertainty, inference in Bayesian-inspired deep neural networks is difficult. In this paper, we provide a practical approach to Bayesian learning that relies on a regularization technique found in nearly every modern network, batch normalization. We show that training a deep network using batch normalization is equivalent to approximate inference in Bayesian models, and we demonstrate how this finding allows us to make useful estimates of the model uncertainty. Using our approach, it is possible to make meaningful uncertainty estimates using conventional architectures without modifying the network or the training procedure. Our approach is thoroughly validated in a series of empirical experiments on different tasks and using various measures, showing it to outperform baselines on a majority of datasets with strong statistical significance.

1 INTRODUCTION

Deep learning has dramatically advanced the state of the art in a number of domains, and now surpasses human-level performance for certain tasks such as recognizing the contents of an image (He et al., 2015) and playing Go (Silver et al., 2017). But, despite their unprecedented discriminative power, deep networks are prone to make mistakes. Sometimes, the consequences of mistakes are minor – misidentifying a food dish or a species of flower (Liu et al., 2016) may not be life threatening. But deep networks can already be found in settings where errors carry serious repercussions such as autonomous vehicles (Chen et al., 2016) and high frequency trading. In medicine, we can soon expect automated systems to screen for skin cancer (Esteva et al., 2017), breast cancer (Shen, 2017), and to diagnose biopsies (Djuric et al., 2017). As autonomous systems based on deep learning are increasingly deployed in settings with the potential to cause physical or economic harm, we need to develop a better understanding of when we can be confident in the estimates produced by deep networks, and when we should be less certain.

Standard deep learning techniques used for supervised learning lack methods to account for uncertainty in the model, although sometimes the classification network's output vector is mistakenly understood to represent the model's uncertainty. The lack of a confidence measure can be especially problematic when the network encounters conditions it was not exposed to during training. For example, if a network trained to recognize dog breeds is given an image of a cat, it may predict it to belong to a breed of small dog with high probability. When exposed to data outside of the distribution it was trained on, the network is forced to extrapolate, which can lead to unpredictable behavior. In such cases, if the network can provide information about its uncertainty in addition to its point estimate, disaster may be avoided. This work focuses on estimating such predictive uncertainties in deep networks (Figure 1).

The Bayesian approach provides a solid theoretical framework for modeling uncertainty (Ghahramani, 2015), which has prompted several attempts to extend neural networks (NN) into a Bayesian setting. Most notably, Bayesian neural networks (BNNs) have been studied since the 1990's (Neal, 2012). Although they are simple to formulate, BNNs require substantially more computational resources than their non-Bayesian counterparts, and inference is difficult. Importantly, BNNs do

Anonymous (2018)

Monte Carlo Batch Normalization (MCBN)

Batch normalized deep nets as Bayesian modeling

Batch normalization is a unit-wise operation to standardize the distribution of each unit's input. It essentially converts a unit's output in the following way:

$$\hat{h} = \frac{h - \mathbb{E}[h]}{\sqrt{\mathrm{Var}[h]}}$$

where the expectations are computed over the training set.

In a batch normalized network, we decouple the learnable parameters from the **stochastic parameters** (batch norm params).

Anonymous (2018)

Batch normalized deep nets as Bayesian modeling

$$\mathcal{L}_{\mathrm{RR}}(\boldsymbol{\theta}) := -\frac{1}{N}\sum_{i=1}^{N} \ln f_{\{\boldsymbol{\theta},\hat{\mathbf{z}}_i\}}(\mathbf{x}_i, \mathbf{y}_i) + \Omega(\boldsymbol{\theta})$$

Learnable parameter $\boldsymbol{\theta} = \{\mathbf{W}^{1:L}, \boldsymbol{\gamma}^{1:L}, \boldsymbol{\beta}^{1:L}\}$

Stochastic parameter $\mathbf{z} = \{\boldsymbol{\mu}_{\mathbf{B}}^{1:L}, \boldsymbol{\sigma}_{\mathbf{B}}^{1:L}\}$

Anonymous (2018)

Batch normalized deep nets as Bayesian modeling

$$\mathbb{E}_{p^*}[\mathbf{y}] = \int \mathbf{y} p^*(\mathbf{y}|\mathbf{x}, \mathbf{D}) d\mathbf{y}$$

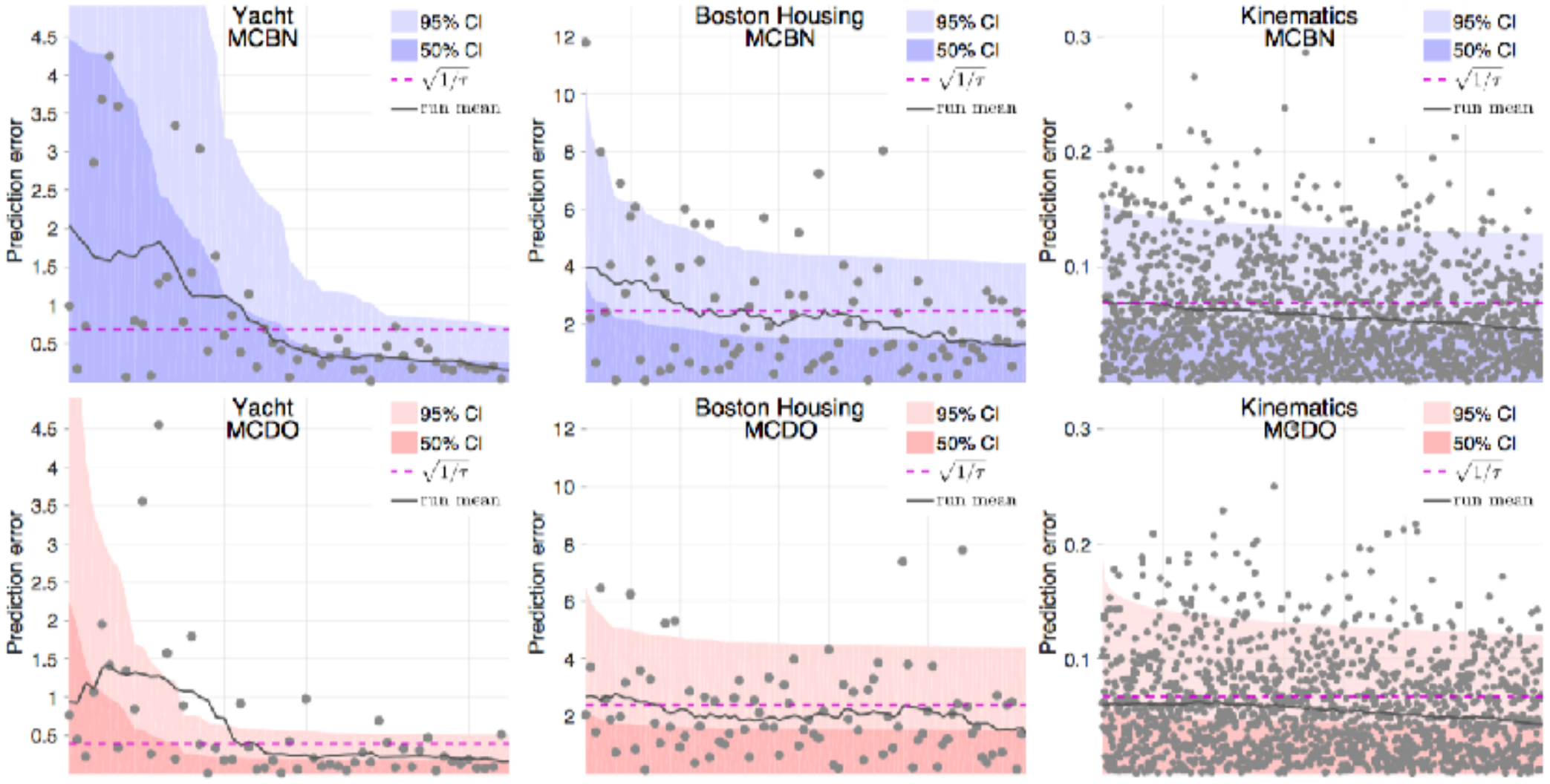
$$\approx \frac{1}{T} \sum_{t=1}^{T} f^{\widehat{\boldsymbol{\omega}}_t}(\mathbf{x})$$

$$\mathrm{Cov}_{p^*}[\mathbf{y}] = \mathbb{E}_{p^*}[\mathbf{y}^\top \mathbf{y}] - \mathbb{E}_{p^*}[\mathbf{y}]^\top \mathbb{E}_{p^*}[\mathbf{y}]$$

$$\approx \tau^{-1}\mathbf{I} + \mathrm{Cov}_{p^*}[\mathbf{y}]$$

Anonymous (2018)

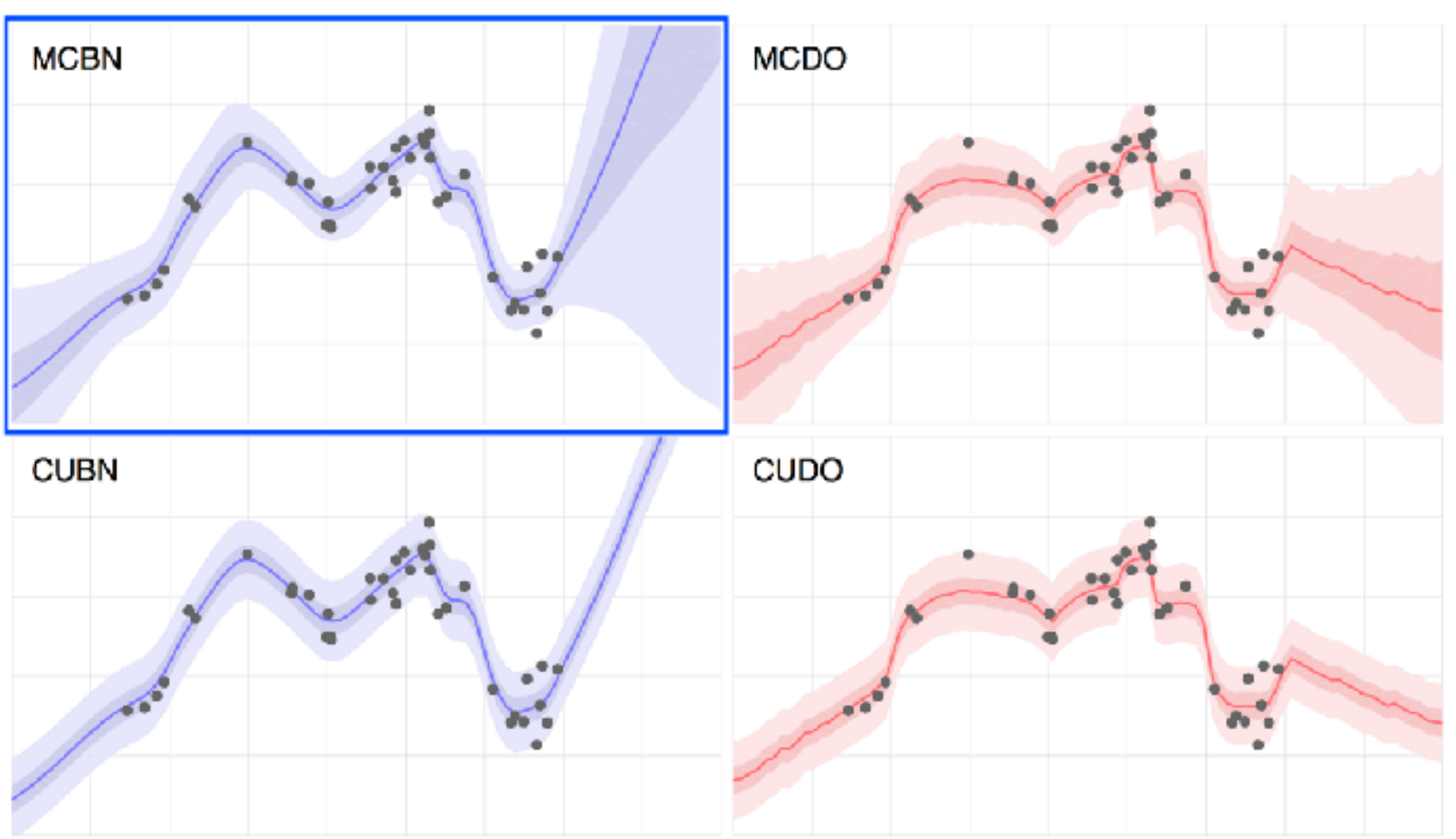
Monte Carlo Batch Normalization (MCBN)



MCBN: Monte Carlo batch normalization
MCDO: Monte Carlo drop-out

Anonymous (2018)

Monte Carlo Batch Normalization (MCBN)



MCBN: Monte Carlo batch normalization
MCDO: Monte Carlo drop-out
CUBN: constant uncertainty batch normalization
CUDO: constant uncertainty drop-out

B. Lakshminarayanan et al., Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles, 2017

Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles

Balaji Lakshminarayanan [1] Alexander Pritzel [1] Charles Blundell [1]

Abstract

Deep neural networks are powerful black box predictors that have recently achieved impressive performance on a wide spectrum of tasks. Quantifying predictive uncertainty in neural networks is a challenging and yet unsolved problem. Bayesian neural networks, which learn a distribution over weights, are currently the state-of-the-art for estimating predictive uncertainty; however these require significant modifications to the training procedure and are computationally expensive compared to standard (non-Bayesian) neural neural networks. We propose an alternative to Bayesian neural networks, that is simple to implement, readily parallelizable and yields high quality predictive uncertainty estimates. Through a series of experiments on classification and regression benchmarks, we demonstrate that our method produces well-calibrated uncertainty estimates which are as good or better than approximate Bayesian neural networks. To assess robustness to dataset shift, we evaluate the predictive uncertainty on test examples from known and unknown distributions, and show that our method is able to express higher uncertainty on unseen data. We demonstrate the scalability of our method by evaluating predictive uncertainty estimates on ImageNet.

1. Introduction

Deep learning methods such as convolutional neural networks and recurrent neural networks have achieved state-of-the-art performance on a wide variety of machine learning tasks and are becoming increasingly popular in domains such as computer vision (Krizhevsky et al., 2012), speech recognition (Hinton et al., 2012), natural language processing (Mikolov et al., 2013) and bioinformatics (LeCun et al., 2015). Despite impressive classification accuracies and mean squared errors in supervised learning problems, neural networks are poor at quantifying predictive uncertainty, and tend to be overconfident in their predictions. Evaluating the quality of predictive uncertainties is challenging as the true conditional probabilities of the data are usually not available. In this work, we shall focus upon two measures of the *quality of predictive uncertainty* from a neural network. Firstly, we shall examine *calibration* (Dawid, 1982; DeGroot and Fienberg, 1983). Formally, calibration is the discrepancy between subjective forecasts and (empirical) long-run frequencies. This is a frequentist notion of uncertainty: if a network claims that 90% of the time a particular label is the correct label, then, during evaluation, 90% of all labels ascribed probability 90% of being correct, should be the correct label. The quality of calibration can be measured by *proper scoring rules* (Gneiting and Raftery, 2007) such as log predictive probabilities and the Brier score (Brier, 1950). Interestingly, these two proper scoring rules are commonly used in deep learning, but without reference to their properties for incentivising calibration. Note that calibration is an orthogonal concern to accuracy: a network's prediction may be accurate and yet miscalibrated. The second notion of quality of predictive uncertainty we consider concerns generalisation of the predictive uncertainty to domain shift, that is, measuring if the network knows *what it knows*. For example, if a network trained on one dataset is evaluated on a completely different dataset, then the network should output high predictive uncertainty.

Well-calibrated predictions have a number of important practical uses and lie at the heart of many forecasting problems about the physical world (e.g., weather, earthquakes, medical diagnosis, etc). Robustness to misspecification is a common requirement in many real world applications as training data is incomplete or lags behind the most recent data upon which predictions are to be made. Calibrated predictions permit modularity: if all components of a system are well-calibrated, then the uncertainty of various predictions can easily be integrated as a common currency of uncertainty between different modules.

There has been a lot of recent interest in adapting neural networks to encompass uncertainty and probabilistic methods; the majority of this work revolves around a Bayesian formalism (Bernardo and Smith, 2009), whereby a prior distribution is specified upon the weights of a single neural network and then an approximation scheme is derived that infers the posterior distribution upon the weights of the neural net-

[1]DeepMind, London, United Kingdom. Correspondence to: Balaji Lakshminarayanan <balajiln@google.com>.

Lakshminarayanan et al. (2017)

Introduction

<u>Bayesian neural networks vs. non-Bayesian neural networks</u>

“Bayesian neural networks, which learn a distribution over weights, are currently the state-of-the-art for estimating predictive uncertainty; however these require significant modifications to the training procedure and are computationally expensive compared to standard (non-Bayesian) neural neural networks.

We propose **an alternative to Bayesian neural networks**, that is simple to implement, readily parallelisable and yields high quality predictive uncertainty estimates.”

Lakshminarayanan et al. (2017)

Deep ensembles for uncertainty estimation

Proper scoring rules

"Scoring rules measure the quality of predictive uncertainty. A **scoring rule** assigns a numerical score to a predictive distribution rewarding better calibrated predictions over worse. (…) It turns out many common neural network loss functions are proper scoring rules."

Lakshminarayanan et al. (2017)

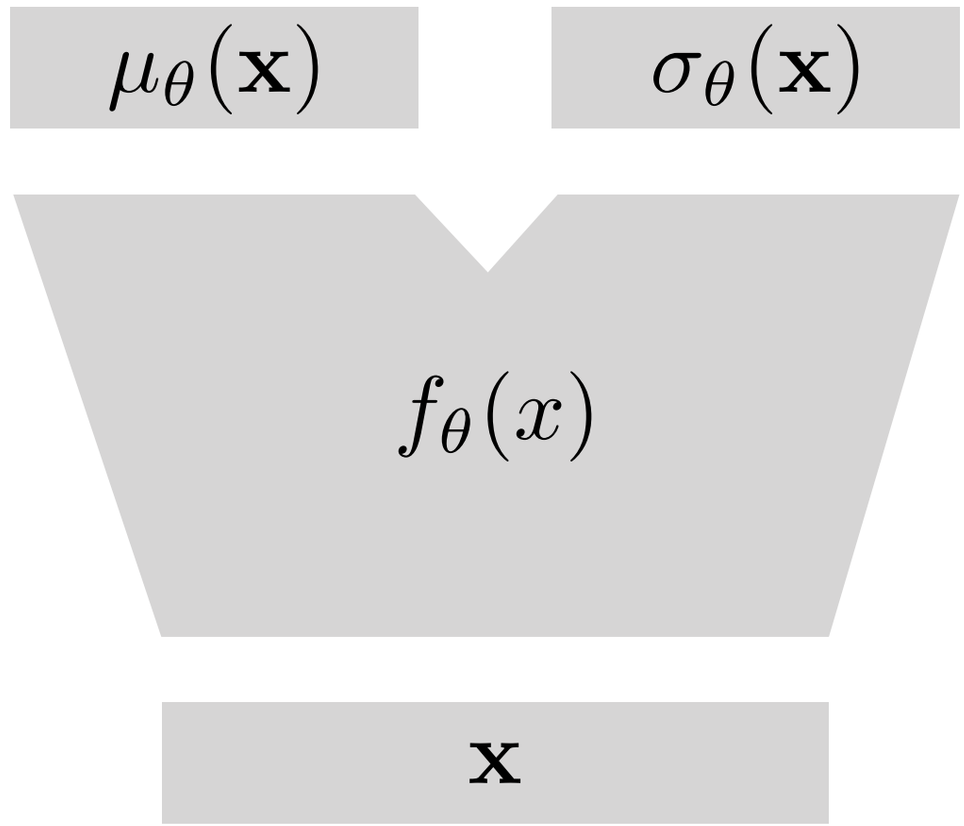
Deep ensembles for uncertainty estimation

Ensemble

AA

Lakshminarayanan et al. (2017)

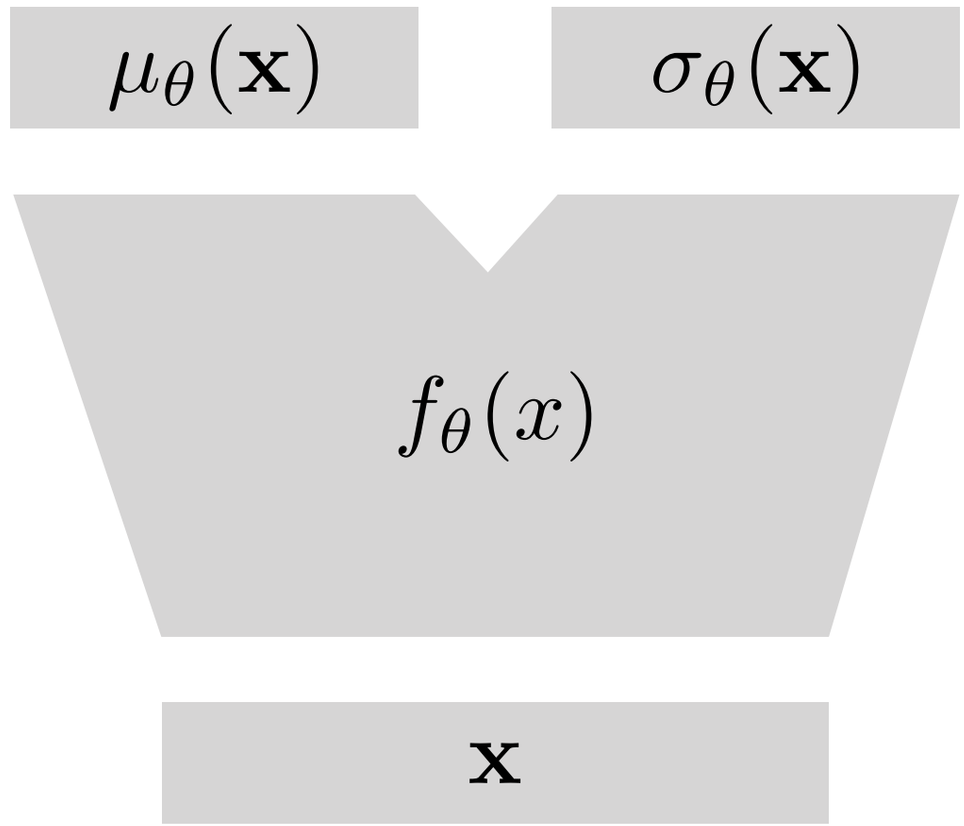
Density network



$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} \log \mathcal{N}(y_i; \mu_\theta(x_i), \sigma_\theta^2(x_i))$$

Lakshminarayanan et al. (2017)

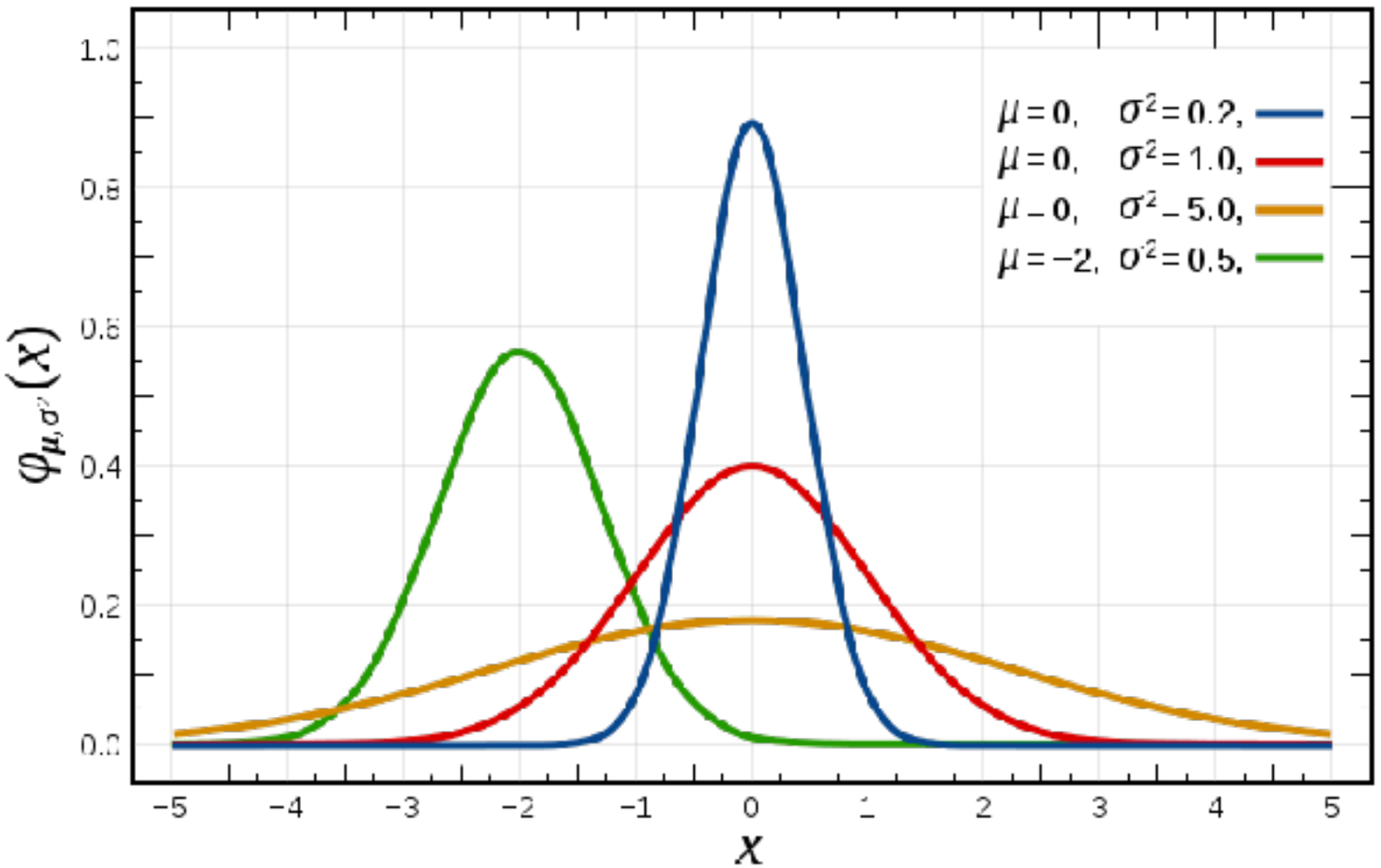
Density network



$$-\log p_\theta(y_n|\mathbf{x}_n) = \frac{\log \sigma_\theta^2(\mathbf{x})}{2} + \frac{(y-\mu_\theta(\mathbf{x}))^2}{2\sigma_\theta^2(\mathbf{x})} + C$$

Lakshminarayanan et al. (2017)

Density network



$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} \log \mathcal{N}(y_i; \mu_\theta(x_i), \sigma^2_\theta(x_i))$$

Lakshminarayanan et al. (2017)

Why density network?

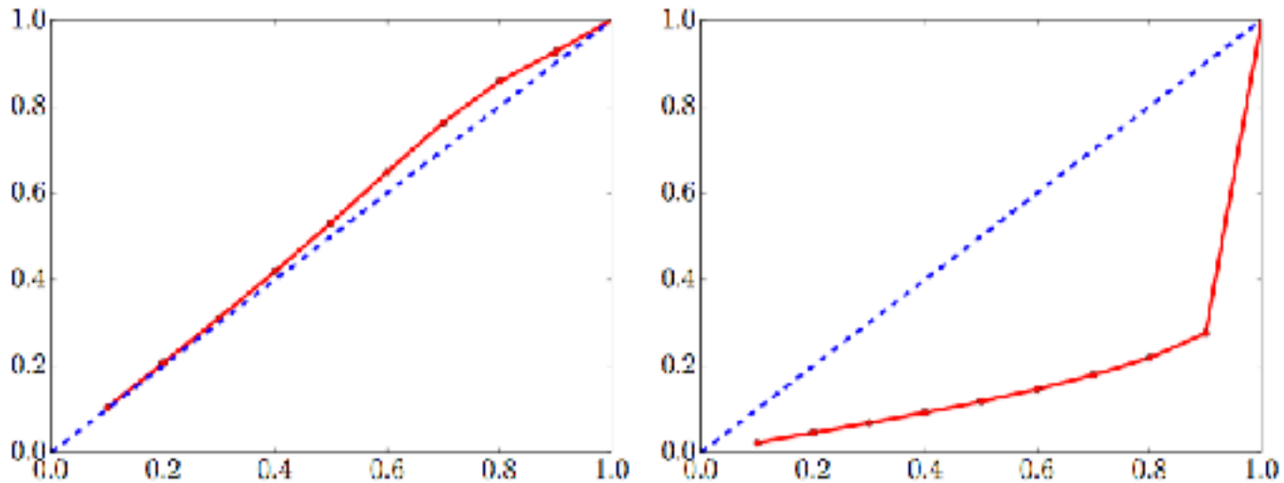
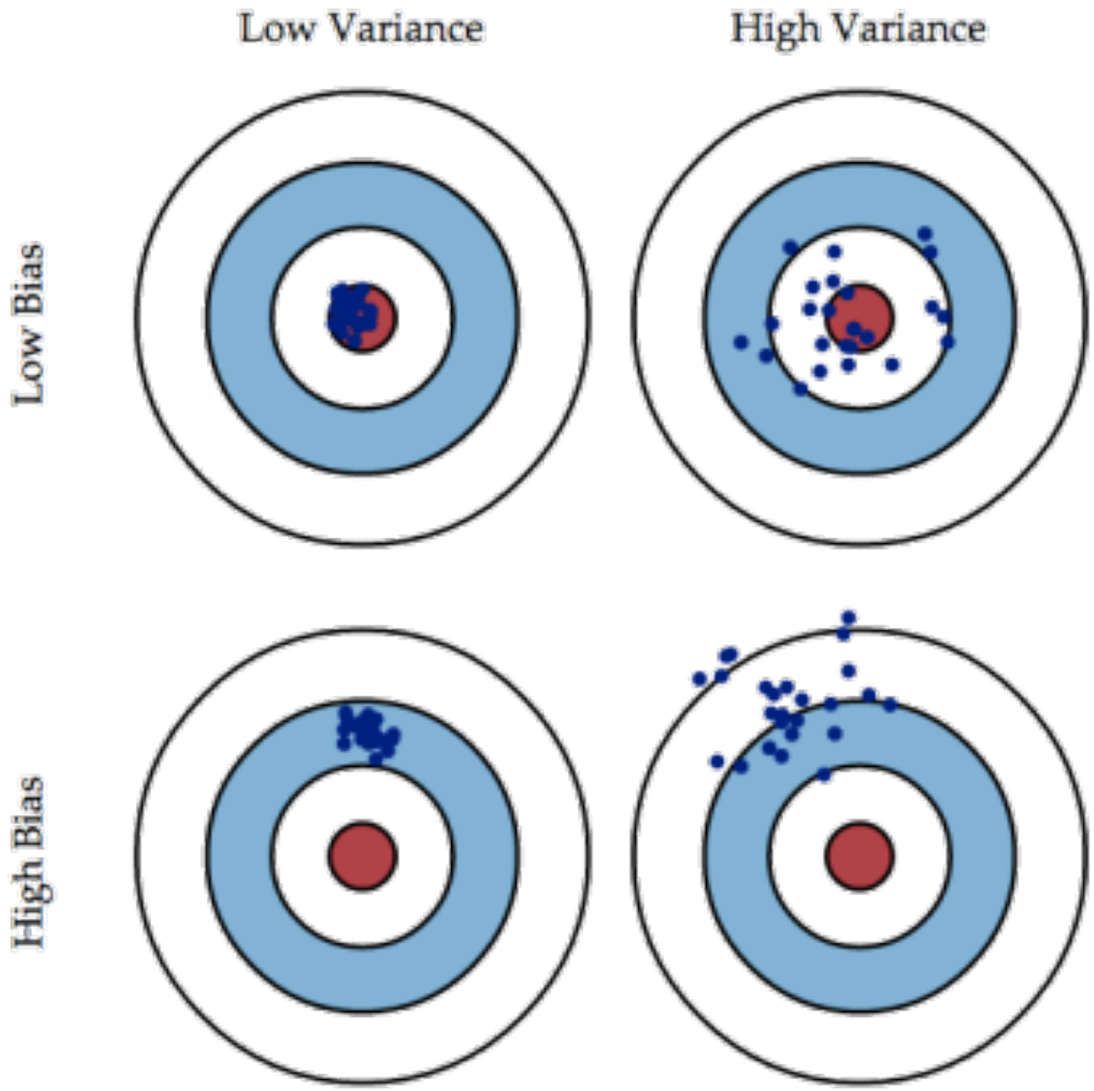


Figure 1. Calibration results on the Year Prediction MSD dataset: x-axis denotes the expected fraction and y-axis denotes the observed fraction; ideal output is the dashed blue line. Predicted variance (left) is significantly better calibrated than the empirical variance (right). See main text for further details.

Left side: predictive variance with a density network
Right side: ensemble of deterministic networks

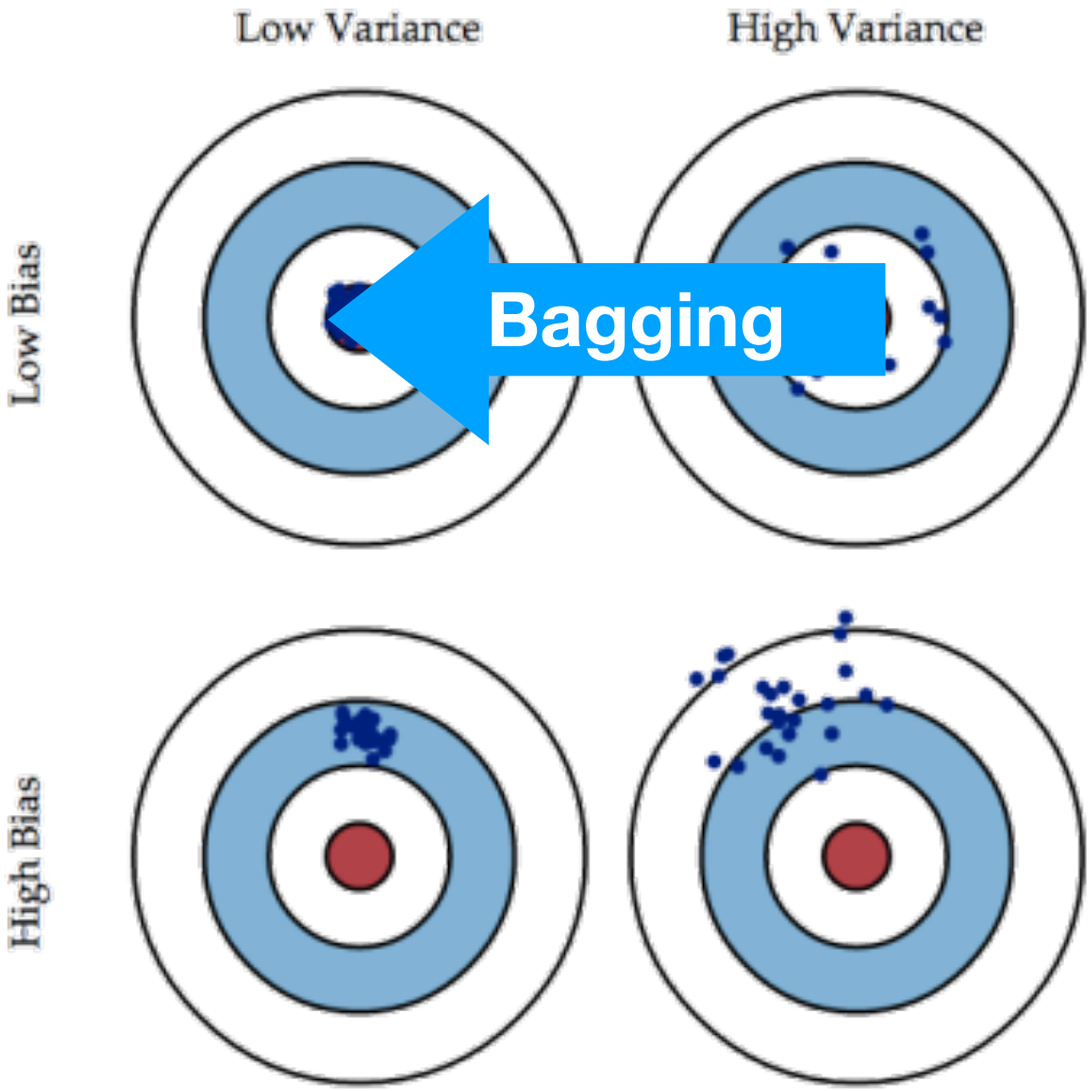
Lakshminarayanan et al. (2017)

Why ensemble?



Lakshminarayanan et al. (2017)

Why ensemble?

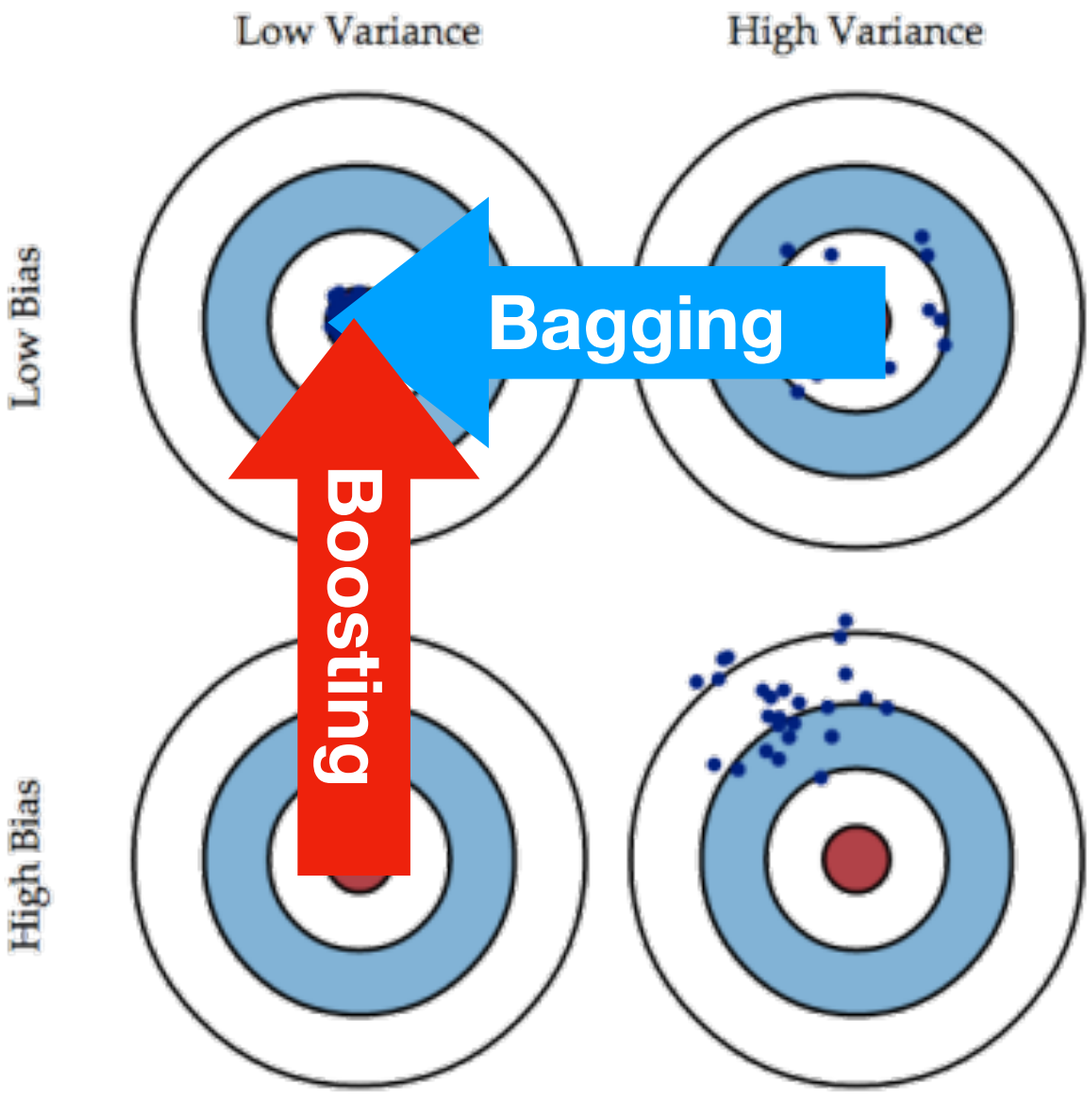


Bagging:

- Generate multiple sets of training data (with bootstrapping), multiple predictions and combine them with averaging.
- Reduce overall variance using multiple low-bias models
- Example: random forest

Lakshminarayanan et al. (2017)

Why ensemble?



Bagging:

- Generate multiple sets of training data (with bootstrapping), multiple predictions and combine them with averaging.
- Reduce overall variance using multiple low-bias models
- Example: random forest

Boosting:

- Generate multiple weak learners and combine them to make a strong learner.
- Reduce overall bias using multiple weak learners.
- Example: AdaBoost

Lakshminarayanan et al. (2017)

Adversarial training with a fast gradient sign method

$$\mathbf{x}' = \mathbf{x} + \epsilon \operatorname{sign}\Big(\nabla_{\mathbf{x}}\, \ell(\theta, \mathbf{x}, y)\Big)$$

“Adversarial training can be also be interpreted as a computationally efficient solution to **smooth the predictive distributions** by increasing the likelihood of the target around an neighborhood of the observed training examples.”

Lakshminarayanan et al. (2017)

Proposed method

Algorithm 1 Pseudocode of the training procedure for our method

1: Initialise $\theta_1, \theta_2, \ldots, \theta_M$ randomly
2: **for** $m = 1 : M$ **do** **Train M different models** ▷ *train networks independently in parallel*
3: Sample data point n_m randomly for each net ▷ *single n_m for clarity, minibatch in practice*
4: Generate adversarial example using $\mathbf{x}'_{n_m} = \mathbf{x}_{n_m} + \epsilon \operatorname{sign}\left(\nabla_{\mathbf{x}_{n_m}} \ell(\theta_m, \mathbf{x}_{n_m}, y_{n_m})\right)$
5: Minimise $\ell(\theta_m, \mathbf{x}_{n_m}, y_{n_m}) + \ell(\theta_m, \mathbf{x}'_{n_m}, y_{n_m})$ w.r.t. θ_m ▷ *adversarial training*

Lakshminarayanan et al. (2017)

Proposed method

Algorithm 1 Pseudocode of the training procedure for our method

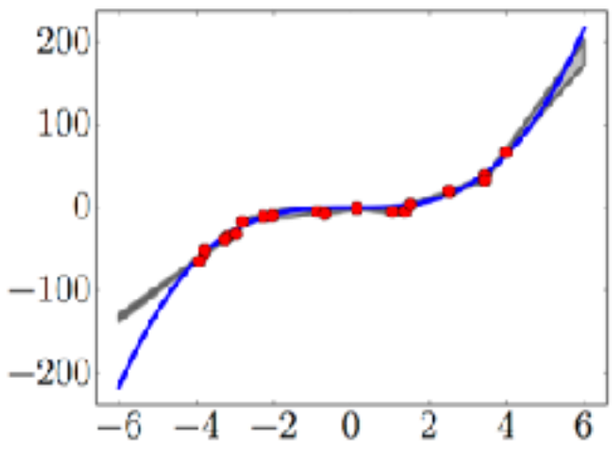
1: Initialise $\theta_1, \theta_2, \ldots, \theta_M$ randomly
2: **for** $m = 1 : M$ **do** **Train M different models** ▷ *train networks independently in parallel*
3: Sample data point n_m randomly for each net ▷ *single n_m for clarity, minibatch in practice*
4: Generate adversarial example using $\mathbf{x}'_{n_m} = \mathbf{x}_{n_m} + \epsilon \operatorname{sign}\left(\nabla_{\mathbf{x}_{n_m}} \ell(\theta_m, \mathbf{x}_{n_m}, y_{n_m})\right)$
5: Minimise $\ell(\theta_m, \mathbf{x}_{n_m}, y_{n_m}) + \ell(\theta_m, \mathbf{x}'_{n_m}, y_{n_m})$ w.r.t. θ_m ▷ *adversarial training*

Treat the ensemble as a uniformly-weighted mixture model and combine the prediction as:

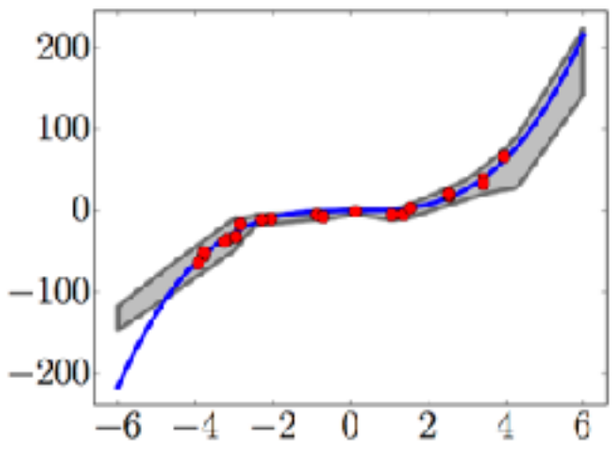
$$p(y|x) = \frac{1}{M} \sum_{m=1}^{M} p_{\theta_m}(y|x, \theta_m)$$

Lakshminarayanan et al. (2017)

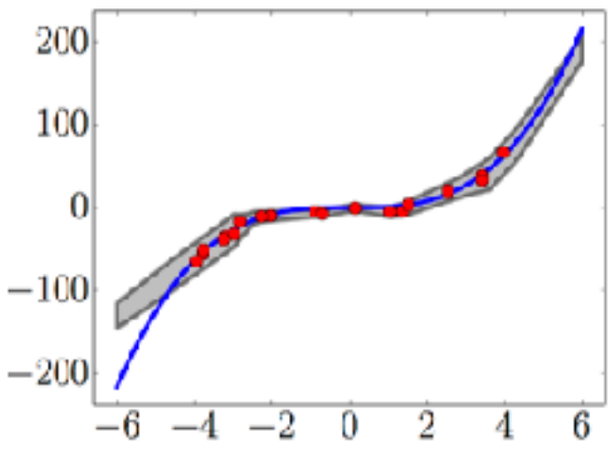
Experiments



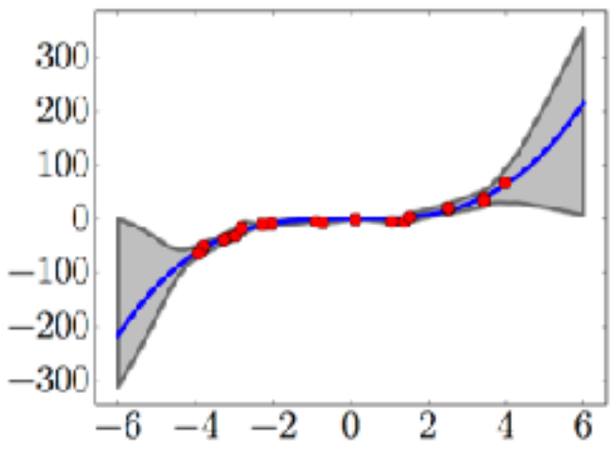
Empirical variance (5)



Density network (1)



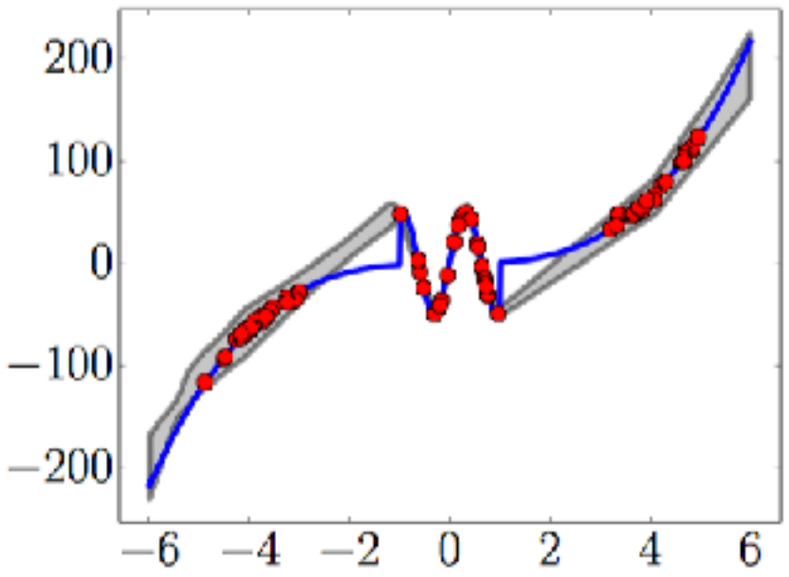
Adversarial training



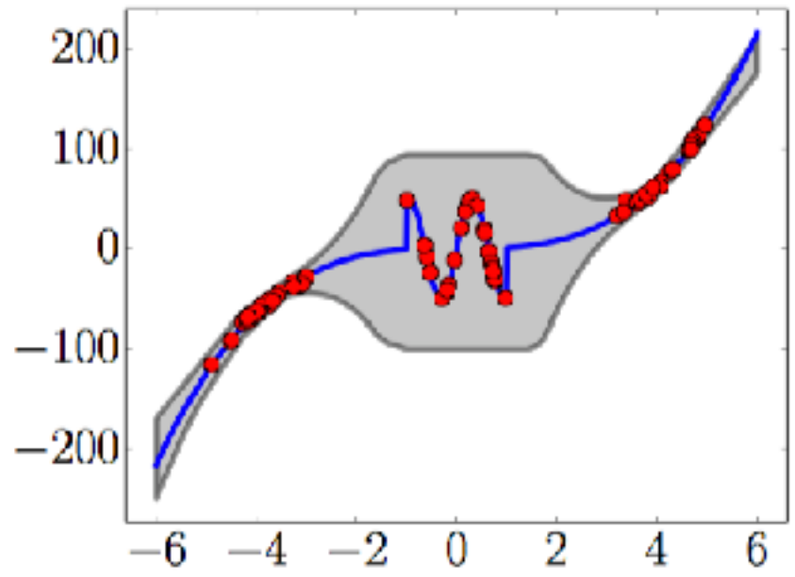
Deep ensemble (5)

Lakshminarayanan et al. (2017)

Experiments



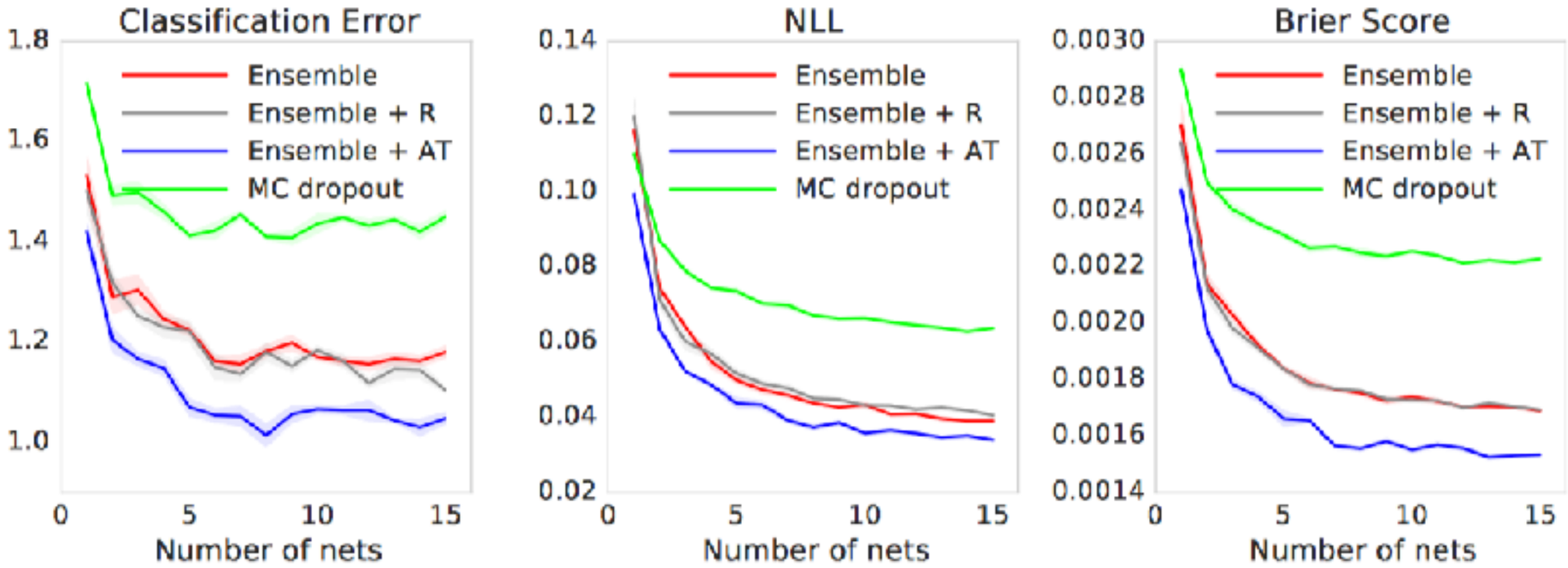
Empirical variance (5)



Deep ensemble (5)

Lakshminarayanan et al. (2017)

Results on MNIST



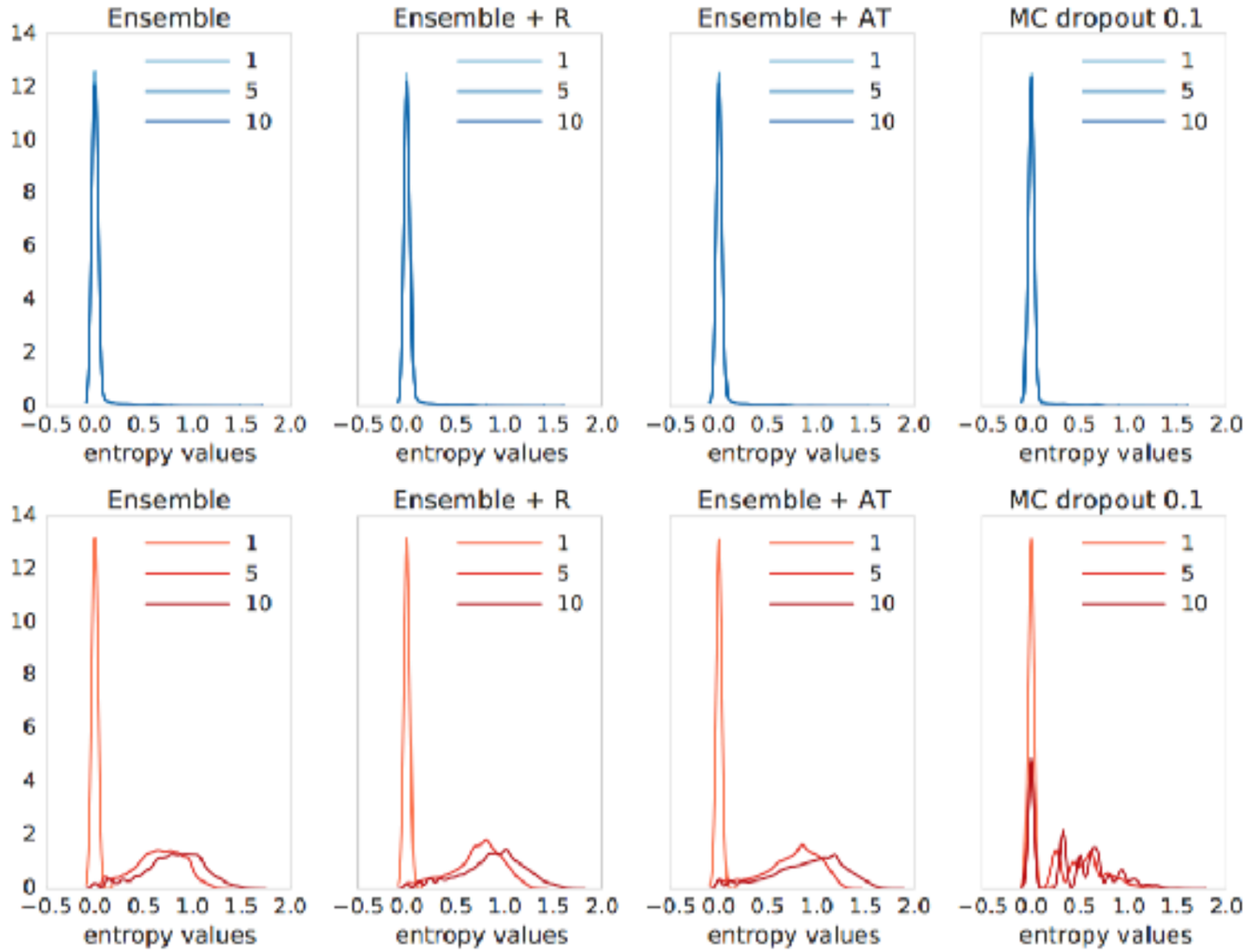
Lakshminarayanan et al. (2017)

Uncertainty evaluation

Overconfident predictions on unseen classes pose a challenge for reliable deployment of deep learning models in real world applications.

Lakshminarayanan et al. (2017)

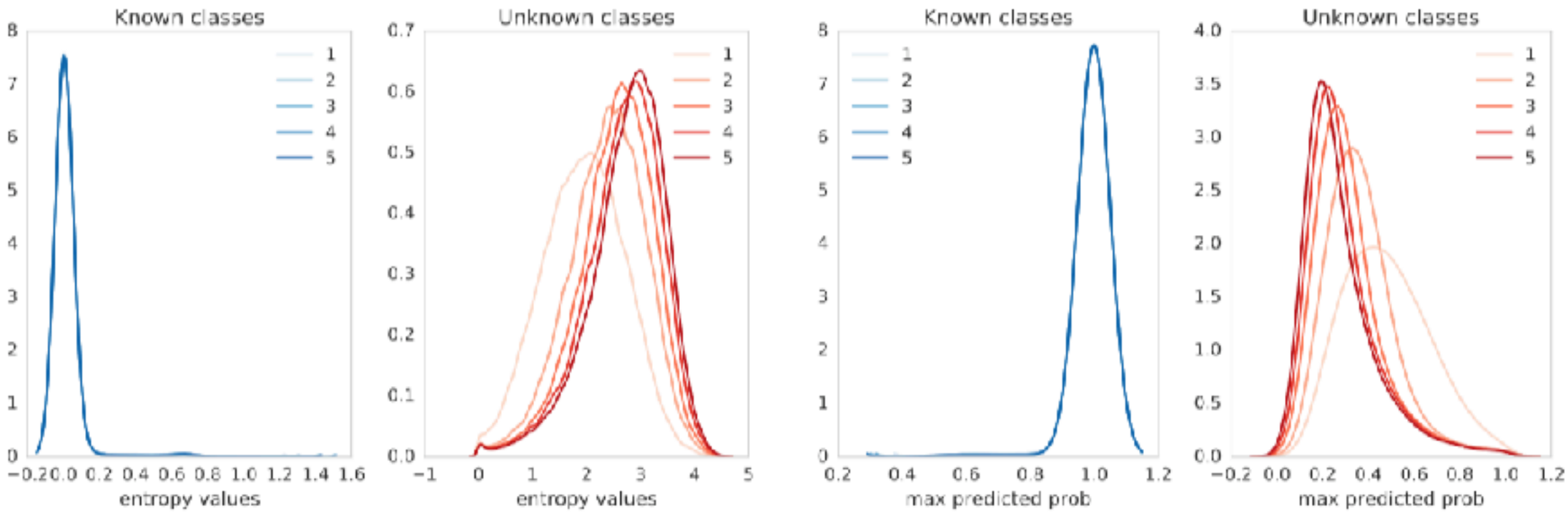
Uncertainty evaluation



MNIST-NotMNIST

Lakshminarayanan et al. (2017)

Uncertainty evaluation



Dog-nonDogs

A. Kendal and Y. Gal, What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?, 2017

What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?

Alex Kendall
University of Cambridge
agk34@cam.ac.uk

Yarin Gal
University of Cambridge
yg279@cam.ac.uk

Abstract

There are two major types of uncertainty one can model. *Aleatoric* uncertainty captures noise inherent in the observations. On the other hand, *epistemic* uncertainty accounts for uncertainty in the model – uncertainty which can be explained away given enough data. Traditionally it has been difficult to model epistemic uncertainty in computer vision, but with new Bayesian deep learning tools this is now possible. We study the benefits of modeling epistemic vs. aleatoric uncertainty in Bayesian deep learning models for vision tasks. For this we present a Bayesian deep learning framework combining input-dependent aleatoric uncertainty together with epistemic uncertainty. We study models under the framework with per-pixel semantic segmentation and depth regression tasks. Further, our explicit uncertainty formulation leads to new loss functions for these tasks, which can be interpreted as learned attenuation. This makes the loss more robust to noisy data, also giving new state-of-the-art results on segmentation and depth regression benchmarks.

1 Introduction

Understanding what a model does not know is a critical part of many machine learning systems. Today, deep learning algorithms are able to learn powerful representations which can map high dimensional data to an array of outputs. However these mappings are often taken blindly and assumed to be accurate, which is not always the case. In two recent examples this has had disastrous consequences. In May 2016 there was the first fatality from an assisted driving system, caused by the perception system confusing the white side of a trailer for bright sky [1]. In a second recent example, an image classification system erroneously identified two African Americans as gorillas [2], raising concerns of racial discrimination. If both these algorithms were able to assign a high level of uncertainty to their erroneous predictions, then the system may have been able to make better decisions and likely avoid disaster.

Quantifying uncertainty in computer vision applications can be largely divided into regression settings such as depth regression, and classification settings such as semantic segmentation. Existing approaches to model uncertainty in such settings in computer vision include particle filtering and conditional random fields [3, 4]. However many modern applications mandate the use of *deep learning* to achieve state-of-the-art performance [5], with most deep learning models not able to represent uncertainty. Deep learning does not allow for uncertainty representation in regression settings for example, and deep learning classification models often give normalised score vectors, which do not necessarily capture model uncertainty. For both settings uncertainty can be captured with *Bayesian deep learning* approaches – which offer a practical framework for understanding uncertainty with deep learning models [6].

In Bayesian modeling, there are two main types of uncertainty one can model [7]. *Aleatoric* uncertainty captures noise inherent in the observations. This could be for example sensor noise or motion noise, resulting in uncertainty which cannot be reduced even if more data were to be collected. On the other hand, *epistemic* uncertainty accounts for uncertainty in the model parameters – uncertainty

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

Kendal & Gal (2017)

Aleatoric & epistemic uncertainties

<u>Two types of uncertainty</u>

1. **Aleatoric uncertainty**: captures noise inherent in the observation
2. **Epistemic uncertainty**: accounts for uncertainty in the model - uncertainty which can be explained away given enough data.

Kendal & Gal (2017)

Aleatoric & epistemic uncertainties

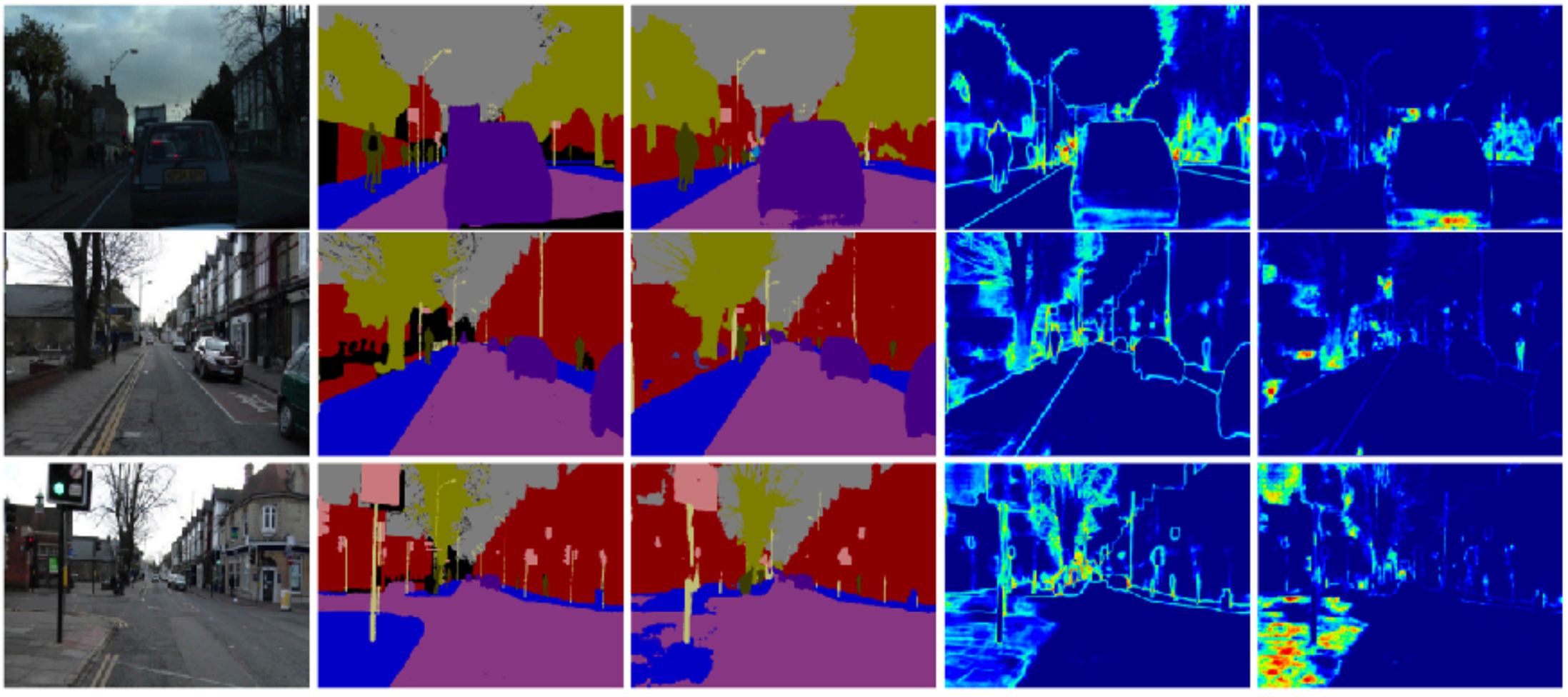
Two types of uncertainty

1. **Aleatoric uncertainty**: is modeled by placing a distribution over the output of the model

2. **Epistemic uncertainty**: is modeled by placing a prior distribution over a model's weights, and then trying to capture how much these weights vary given some data.

Kendal & Gal (2017)

Aleatoric & epistemic uncertainties



(a) Input Image (b) Ground Truth (c) Semantic Segmentation (d) Aleatoric Uncertainty (e) Epistemic Uncertainty

Kendal & Gal (2017)

Aleatoric & epistemic uncertainties

$$\hat{y}_{\hat{\mathbf{W}}}(\mathbf{x}) \qquad \hat{\sigma}^2_{\hat{\mathbf{W}}}(\mathbf{x})$$

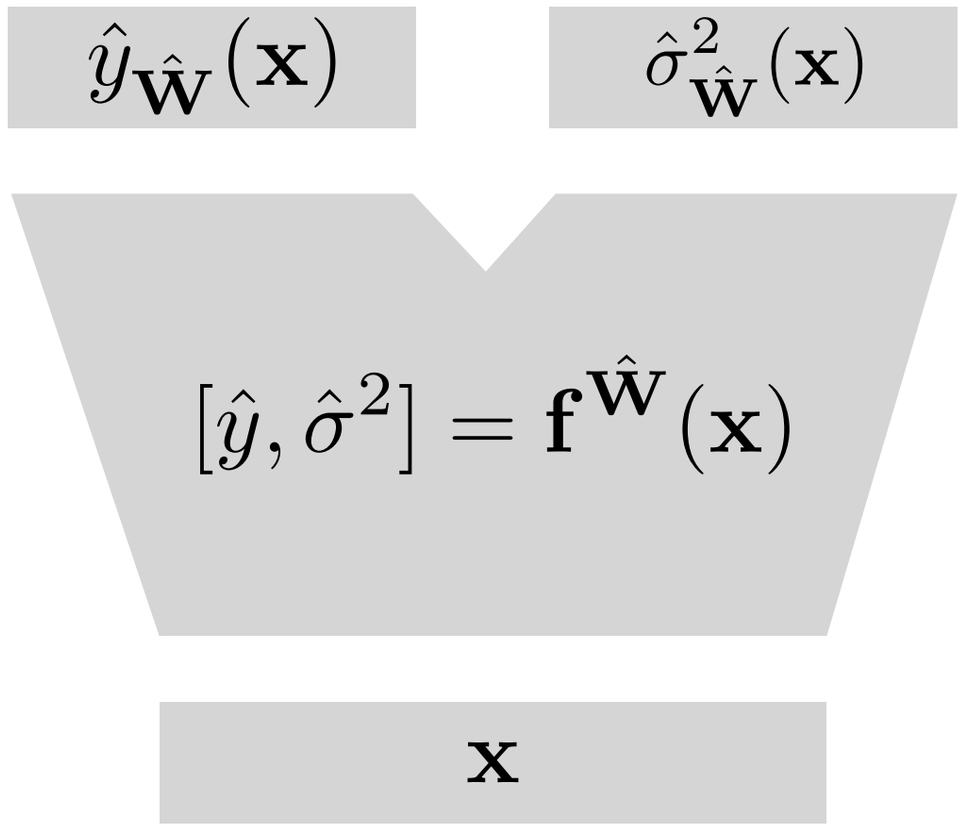
$$[\hat{y}, \hat{\sigma}^2] = \mathbf{f}^{\hat{\mathbf{W}}}(\mathbf{x}) \qquad \hat{\mathbf{W}} \sim q(\mathbf{W})$$

$$\mathbf{x}$$

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} \log \mathcal{N}(y_i; \hat{y}_{\hat{\mathbf{W}}}(\mathbf{x}), \hat{\sigma}^2_{\hat{\mathbf{W}}}(\mathbf{x}))$$

Kendal & Gal (2017)

Heteroscedastic uncertainty as **loss attenuation**

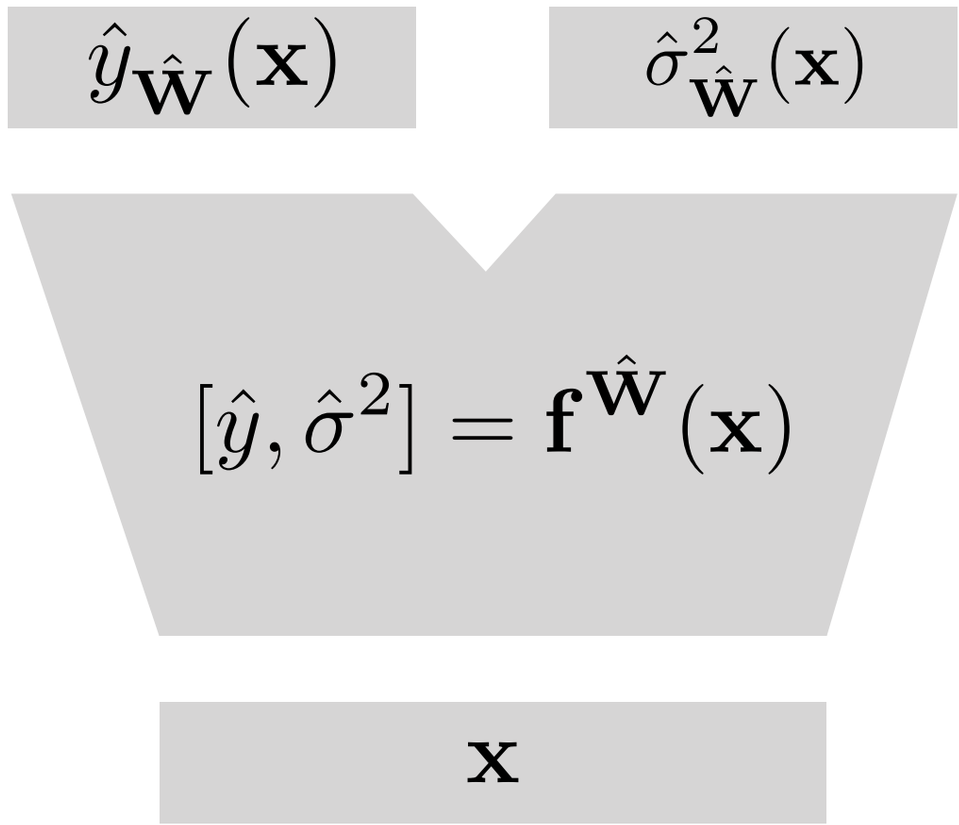


$$\hat{\mathbf{x}}_{i,t} = \mathbf{f}_i^{\mathbf{W}} + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, (\sigma_i^{\mathbf{W}})^2)$$

$$\mathcal{L}_x = \frac{1}{T} \sum_{i,t} (-\hat{x}_{i,t,c} + \log \sum_{c'} \exp \hat{x}_{i,t,c'})$$

Kendal & Gal (2017)

Aleatoric & epistemic uncertainties



$$\mathrm{Var}(y) \approx \underbrace{\frac{1}{T}\sum_{t=1}^{T}\hat{y}_t^2 - \left(\sum_{t=1}^{T}\hat{y}_t\right)^2}_{\text{Epistemic unct.}} + \underbrace{\frac{1}{T}\sum_{t=1}^{T}\hat{\sigma}_t^2}_{\text{Aleatoric unct,}}$$

Kendal & Gal (2017)

Aleatoric & epistemic uncertainties

Heteroscedastic uncertainty as learned loss attenuation

The predictive uncertainty acts as a robust regression function by allowing the network to learn to **attenuate** the effect from erroneous labels. This makes the model m ore robust to noisy data: inputs for which the model learned to predict high uncertainty will have a smaller effect on the loss.

Kendal & Gal (2017)

Aleatoric & epistemic uncertainties

Classification loss

For classification task, a NN predicts a vector of unaries for each pixel i, which when passed through a softmax operation, forms a probability vector p. We change the model by placing a Gaussian distribution over the unaries vector:

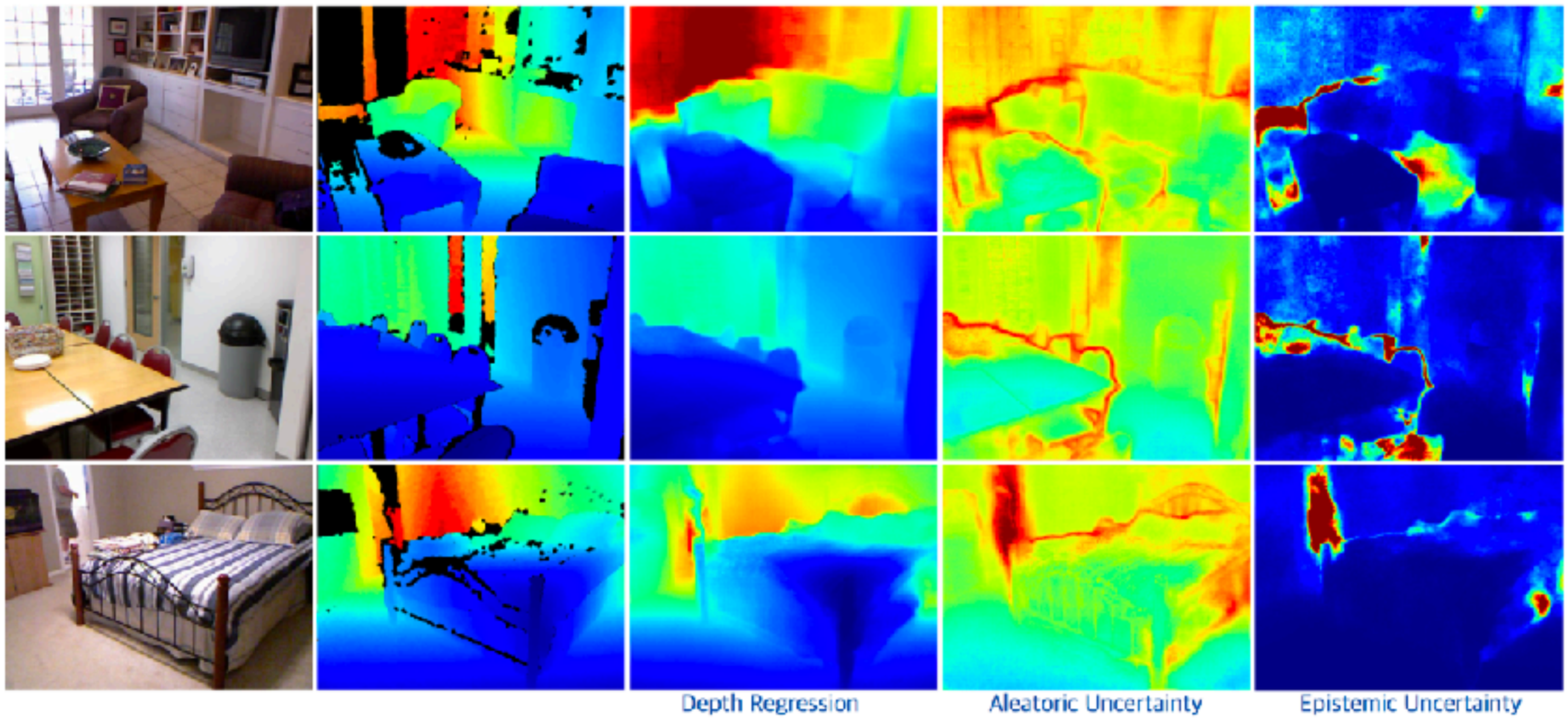
$$\hat{\mathbf{x}}_i|\mathbf{W} \sim \mathcal{N}(\mathbf{f}_i^{\mathbf{W}}, (\sigma_i^{\mathbf{W}})^2)$$
$$\hat{\mathbf{p}}_i = \text{Softmax}(\hat{\mathbf{x}}_i).$$

We can rewrite the above and obtain the following numerically-stable stochastic loss:

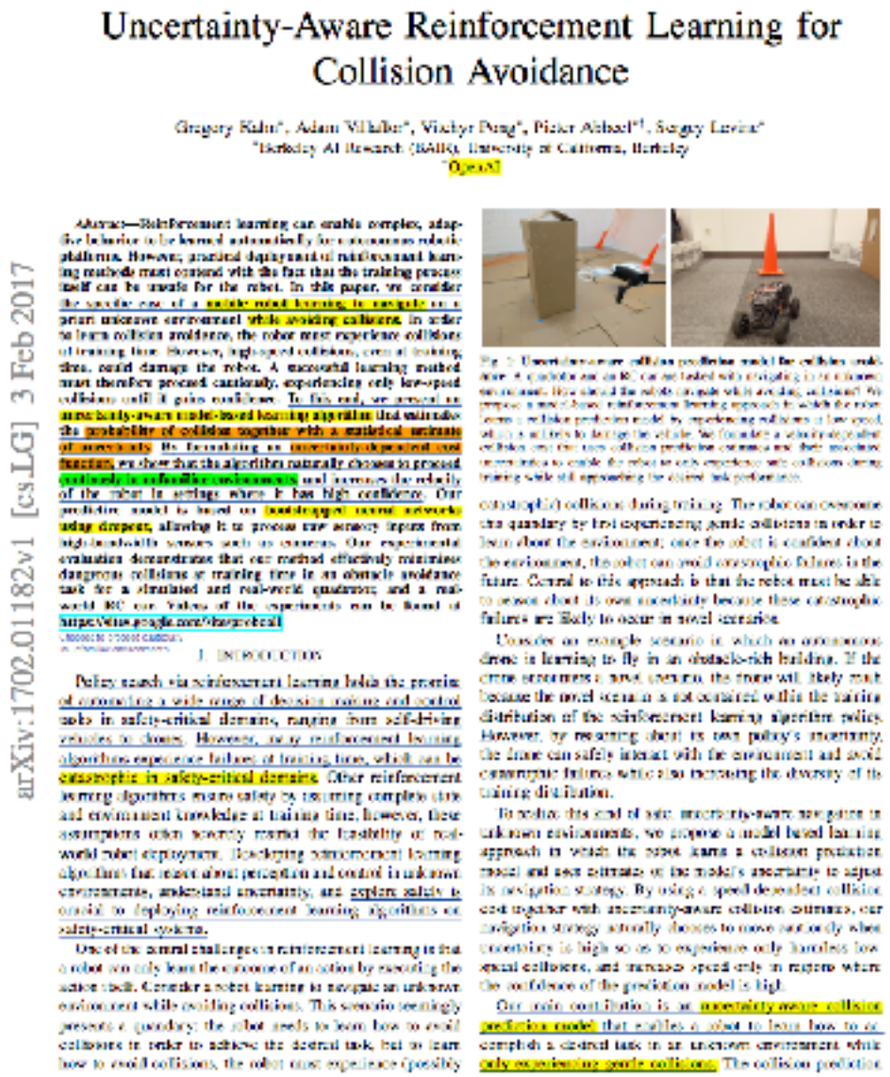
$$\hat{\mathbf{x}}_{i,t} = \mathbf{f}_i^{\mathbf{W}} + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0,\, (\sigma_i^{\mathbf{W}})^2)$$
$$\mathcal{L}_x = \frac{1}{T} \sum_{i,t} (-\hat{x}_{i,t,c} + \log \sum_{c'} \exp \hat{x}_{i,t,c'})$$

Kendal & Gal (2017)

Results



G. Khan et al., Uncertainty-Aware Reinforcement Learning from Collision Avoidance, 2016



Uncertainty-Aware Reinforcement Learning for Collision Avoidance

Gregory Kahn*, Adam Villaflor*, Vitchyr Pong*, Pieter Abbeel*†, Sergey Levine*
*Berkeley AI Research (BAIR), University of California, Berkeley
†OpenAI

Abstract—Reinforcement learning can enable complex, adaptive behavior to be learned automatically for autonomous robotic platforms. However, practical deployment of reinforcement learning methods must contend with the fact that the training process itself can be unsafe for the robot. In this paper, we consider the specific case of a mobile robot learning to navigate in a priori unknown environments while avoiding collisions. In order to learn collision avoidance, the robot must experience collisions at training time. However, high-speed collisions, even at training time, could damage the robot. A successful learning method must therefore proceed cautiously, experiencing only low-speed collisions until it gains confidence. To this end, we present an uncertainty-aware model-based learning algorithm that estimates the probability of collision together with a statistical estimate of uncertainty. By formulating an uncertainty-dependent cost function, we show that the algorithm naturally chooses to proceed cautiously in unfamiliar environments, and increases the velocity of the robot in settings where it has high confidence. Our predictive model is based on bootstrapped neural networks using dropout, allowing it to process raw sensory inputs from high-dimensional sensors such as cameras. Our experimental evaluation demonstrates that our method effectively minimizes dangerous collisions at training time in an obstacle avoidance task for a simulated and real-world quadrotor, and a real-world RC car. Videos of the experiments can be found at https://sites.google.com/view/gpcollision

I. INTRODUCTION

Policy search via reinforcement learning holds the promise of automating a wide range of decision-making and control tasks in safety-critical domains, ranging from self-driving vehicles to drones. However, as in reinforcement learning algorithms experience failures at training time, which can be catastrophic in safety-critical domains. Other reinforcement learning algorithms ensure safety by assuming complete prior and environment knowledge at training time; however, these assumptions often severely restrict the feasibility of real-world robot deployment. Developing reinforcement learning algorithms that reason about perception and control in unknown environments, understand uncertainty, and explore safely to avoid catastrophic failures is critical to deploying reinforcement learning algorithms on safety-critical systems.

One of the central challenges in reinforcement learning is that a robot can only learn the outcome of an action by executing the action itself. Consider a robot learning to navigate an unknown environment while avoiding collisions. This scenario seemingly presents a paradox: the robot needs to learn how to avoid collisions in order to achieve the desired task, but in order to learn how to avoid collisions, the robot must experience (possibly catastrophic) collisions during training. The robot can overcome this quandary by first experiencing gentle collisions in order to learn about the environment; once the robot is confident about the environment, the robot can avoid catastrophic failures in the future. Central to this approach is that the robot must be able to reason about its own uncertainty because these catastrophic failures are likely to occur in novel scenarios.

Consider an example scenario in which an autonomous drone is learning to fly in an obstacle-rich building. If the drone encounters a novel scenario, the drone will likely crash because the novel scenario is not contained within the training distribution of the reinforcement learning algorithm policy. However, by reasoning about its own policy's uncertainty, the drone can safely interact with the environment and avoid catastrophic failures while also increasing the diversity of its training distribution.

To realize this end-to-end uncertainty-aware navigation in unknown environments, we propose a model-based learning approach in which the robot learns a collision prediction model and uses estimates of the model's uncertainty to adjust its navigation strategy. By using a speed-dependent collision cost together with uncertainty-aware collision estimates, our navigation strategy actually decides to move fast only when uncertainty is high so as to experience only harmless low-speed collisions, and increases speed only in regions where the confidence of the prediction model is high.

Our main contribution is an uncertainty-aware collision prediction model that enables a robot to learn how to accomplish a desired task in an unknown environment while only experiencing gentle collisions. The collision prediction

Khan et al. (2016)

Uncertainty-Aware Reinforcement Learning

Introduction

"In this paper, we consider the specific case of a mobile robot learning to navigate an a priori unknown environment while avoiding collisions. We present **an uncertainty-aware model-based learning algorithm** that estimates the **probability of collision together with a statistical estimate of uncertainty**. By formulating an uncertainty-dependent cost function, we show that the algorithm naturally chooses to proceed cautiously in unfamiliar environments, and increases the velocity of the robot in settings where it has high confidence."

Khan et al. (2016)

Uncertainty-Aware Reinforcement Learning

Uncertainty-aware collision prediction model

Khan et al. (2016)

Uncertainty-Aware Reinforcement Learning

Collision prediction with uncertainty

The collision prediction model takes the current state and observation as an input and outputs the probability of a collision.

The risk-averse collision estimator is the sum of predicted collision probability and the predictive variance of the mode..

Khan et al. (2016)

Uncertainty-Aware Reinforcement Learning

"Uncertainty is based on **bootstrapped** neural networks using **dropout**."

Bootstrapping?

- Generate multiple datasets using sampling with replacement.
- The **intuition** behind bootstrapping is that, by generating multiple populations and training one model per population, the models will agree in high-density areas (low uncertainty) and disagree in low-density areas (high uncertainty).

Dropout?

- "Dropout can be viewed as an economical approximation of an ensemble method (such as bootstrapping) in which each sampled dropout mask corresponds to a different model."

Khan et al. (2016)

Uncertainty-Aware Reinforcement Learning

Algorithm 1 Neural net training with bootstrapping and dropout

1: **input:** dataset $\mathcal{D} = \{\mathbf{x}_t^{(i)}, \mathbf{u}_{t:t+H}^{(i)}, \mathbf{o}_t^{(i)}\}$, neural network model NN
2: **for** $b = 1$ to B **do** **Train B different models**
3: Sample a dataset of subsequences $\mathcal{D}^{(b)}$ from the full dataset $\mathcal{D}$ with replacement
4: Initialize neural network $\text{NN}^{(b)}$ with random weights
5: **for** number of SGD iterations **do**
6: Sample datapoint $(\mathbf{x}_t, \mathbf{u}_{t:t+H}, \mathbf{o}_t)$ from $\mathcal{D}^{(b)}$
7: Sample $\text{NN}_d^{(b)}$ by masking the units in $\text{NN}^{(b)}$ using dropout
8: Run forward pass on $\text{NN}_d^{(b)}$ using $(\mathbf{x}_t, \mathbf{u}_{t:t+H}, \mathbf{o}_t)$
9: Run backward pass on $\text{NN}_d^{(b)}$ to get gradient $g_d^{(b)}$
10: Update model $\text{NN}^{(b)}$ parameters using $g_d^{(b)}$
11: **end for**
12: **end for**

Safe Visual Navigation via Deep Learning and Novelty Detection

Charles Richter and Nicholas Roy
Massachusetts Institute of Technology
Cambridge, MA, USA

***Abstract*—Robots that use learned perceptual models in the real world must be able to safely handle cases where they are forced to make decisions in scenarios that are unlike any of their training examples. However, state-of-the-art deep learning methods are known to produce erratic or unsafe predictions when faced with novel inputs. Furthermore, recent ensemble, bootstrap and dropout methods for quantifying neural network uncertainty may not efficiently provide accurate uncertainty estimates when queried with inputs that are very different from their training data. Rather than unconditionally trusting the predictions of a neural network for unpredictable real-world data, we use an autoencoder to recognize when a query is novel, and revert to a safe prior behavior. With this capability, we can deploy an autonomous deep learning system in arbitrary environments, without concern for whether it has received the appropriate training. We demonstrate our method with a vision-guided robot that can leverage its deep neural network to navigate 50% faster than a safe baseline policy in familiar types of environments, while reverting to the prior behavior in novel environments so that it can safely collect additional training data and continually improve. A video illustrating our approach is available at: http://groups.csail.mit.edu/rrg/videos/safe_visual_navigation.**

I. Introduction

In many ways, cameras are ideal sensors for mobile robot applications since they provide rich contextual information that is required for many different types of navigation and decision making problems. They are much smaller, lighter and cheaper than LIDAR, and work in a wide variety of indoor and outdoor lighting conditions, unlike most infrared RGBD sensors. However, cameras pose substantial computational challenges to extract information in usable forms. For example, the range of dense, accurate depth information from visual SLAM may be limited (like RGBD) to only a few meters. Therefore, it is reasonable to assume that mobile robots will be able to build geometric maps from images, but that those maps will be very limited in range. Range limitations, in turn, restrict the speed at which a robot can travel, and impede its ability to anticipate more distant structures in the environment.

Nevertheless, camera images contain information about the context and structure of the world far beyond the limited range of accurate geometric inference. For instance, if a robot observes a straight empty hallway ahead, it might reasonably learn from experience and visual appearance that it can safely travel at high speed down the hallway, even if it cannot infer the exact geometry of the entire length of the hallway. If we can extract this type of appearance-based information by other means, we can improve navigation performance.

Deep learning has been shown to be well suited to extracting useful information for navigation from raw sensor data such as camera images [13, 27]. However, unlike visual SLAM algorithms that infer geometry using domain-invariant principles, neural networks adapt their representation to domain-specific training datasets. Therefore, they may make unreliable predictions when queried with inputs far from the distribution of the training data [4]. Yet, we would still like to deploy robots using neural networks "in the wild," where safety is critical but real-world data will almost certainly differ from the training dataset, violating the common i.i.d. assumption in machine learning. Moreover, conventional neural networks provide no principled measure of how certain they are, or how well-trained they are to make a given prediction.

Recognizing this limitation, some work has attempted to quantify uncertainty with Bayesian neural networks [20, 5], by training ensembles of networks on re-sampled versions of the training dataset [26, 17, 8, 18], or by using dropout to train an implicit family of models [9]. While these methods are equipped to model the variance *within* the training data, there has yet to emerge an established technique for reliably producing appropriate uncertainty estimates for regions of input space very far from the training data. Recent techniques appear to be sensitive to particular choices of network architecture and residual randomness from initialization [10, 25], and unlike Bayesian techniques such as Gaussian processes, these methods do not predictably converge to a prior mean and variance far from the training data. Furthermore, these methods require tens or hundreds of network evaluations to approximate the output distribution, compromising runtime efficiency. Given these limitations, we believe that determining the trustworthiness of a neural network prediction may be more effectively addressed as a novelty-detection problem.

In this paper, we demonstrate safe, high-speed visual navigation of an autonomous mobile robot, guided by a neural network in environments that may or may not resemble the training environment(s). We use a conventional feedforward neural network to predict collisions based on images observed by the robot, and we use an autoencoder to judge whether those images are similar enough to the training data for the resulting neural network predictions to be trusted. In fact, Pomerleau [28] utilized a similar form of novelty detection when using a neural network to control an autonomous vehicle, which we extend in several ways. First, if our novelty detector classifies a planning scenario as novel, we revert to

Richer & Roy (2017)

Introduction

State-of-the-art deep learning methods are known to produce erratic or unsafe predictions when faced with **novel inputs**. Furthermore, recent ensemble, bootstrap and dropout methods for quantifying neural network uncertainty may not efficiently provide accurate uncertainty estimates when queried with inputs that are very different from their training data.

We use a conventional feedforward neural network to **predict collisions** based on images observed by the robot, and we use an **autoencoder to judge whether those images are similar enough** to the training data for the resulting neural network predictions to be trusted.

Richer & Roy (2017)

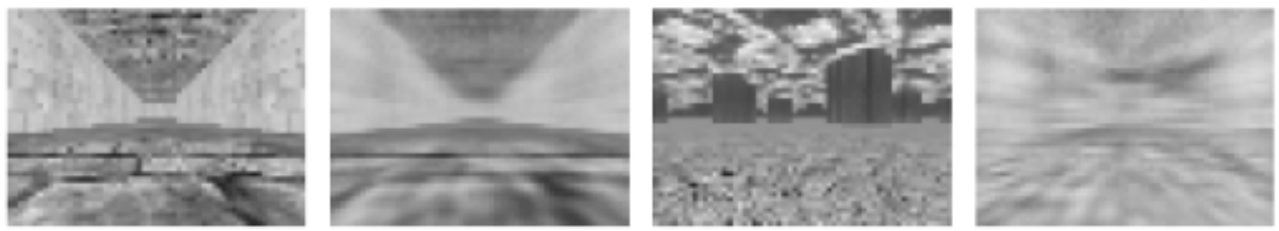
Novelty detection

(a) Input. (b) Output. (c) Input. (d) Output.

Use the **reconstruction error** as a measure of novelty.

Richer & Roy (2017)

Novelty detection



(a) Input. (b) Output. (c) Input. (d) Output.

Fig. 4: (a) A familiar input image of a simulated hallway, drawn from the training distribution, and (b) the associated accurate autoencoder reconstruction. (c) A novel input image of a simulated "cylinder forest," *not* drawn from the training distribution, and (d) the resulting inaccurate reconstruction.

Use the **reconstruction error** as a measure of novelty.

Richer & Roy (2017)

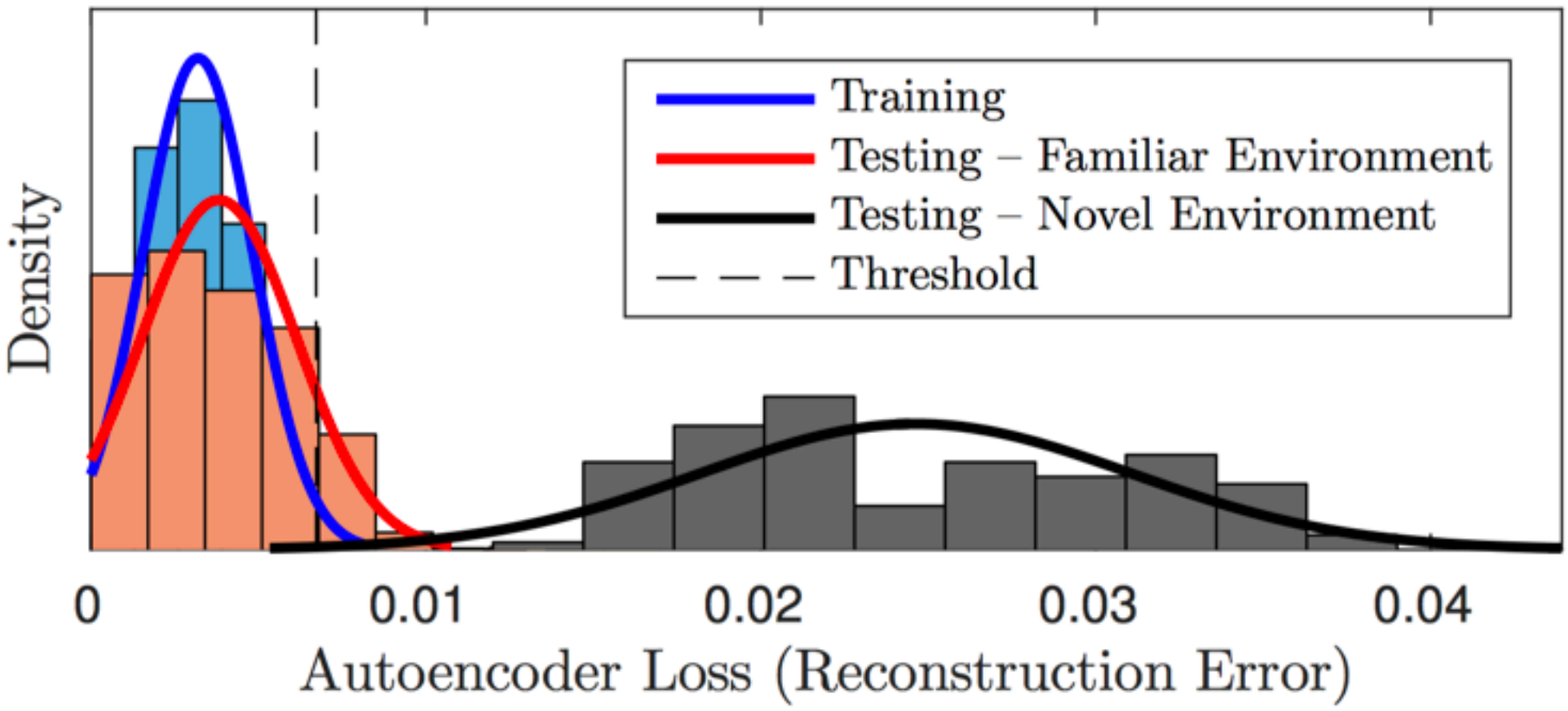
Novelty detection

(a) Familiar Environment.

(b) Novel Environment.

Richer & Roy (2017)

Novelty detection



Richer & Roy (2017)

Learning to predict collision

$$p(c|\hat{m}_t, i_t, a_t) = \begin{cases} f_c(c|i_t, a_t) & \text{if } f_n(i_t) = 0 \\ f_p(c|\hat{m}_t, a_t) & \text{if } f_n(i_t) = 1. \end{cases}$$

where

c : collision

$\hat{m}_t$: estimated map

i_t : input image

a_t : action

$f_c(c|i_t, a_t)$: neural net trained to **predict collision**

$f_p(c|\hat{m}_t, a_t)$: prior estimate of collision probability

$f_n(i_t)$: **novelty detection**

Richer & Roy (2017)

Experiments

"Using an autoencoder as a **measure of uncertainty** in our collision prediction network, we can transition intelligently between the high performance of the learned model and the safe, conservative performance of a simple prior, depending on whether the system has been trained on the relevant data."

Richer & Roy (2017)

Experiments

“In the **hallway training environment**, we achieved a mean speed of **3.26 m/s** and a top speed over **5.03 m/s**. This result significantly exceeds the maximum speeds achieved when driving in this environment under the prior estimate of collision probability before performing any learning.”

“On the other hand, in the **novel environment**, for which our model was untrained, the novelty detector correctly identified every image as being unfamiliar. In the novel environment, we achieved a mean speed of **2.49 m/s** and a maximum speed of **3.17 m/s**.”

S. Choi et al., Uncertainty-Aware Learning from Demonstration Using Mixture Density Networks with Sampling-Free Variance Modeling, 2017

Uncertainty-Aware Learning from Demonstration Using Mixture Density Networks with Sampling-Free Variance Modeling

Sungjoon Choi, Kyungjae Lee, Sungbin Lim, and Songhwai Oh

Abstract— In this paper, we propose an uncertainty-aware learning from demonstration method by presenting a novel uncertainty estimation method utilizing a mixture density network appropriate for modeling complex and noisy human behaviors. The proposed uncertainty acquisition can be done with a single forward path without Monte Carlo sampling and is suitable for real-time robotics applications. Then, we show that it can be decomposed into *explained* variance and *unexplained* variance where the connections between *aleatoric* and *epistemic* uncertainties are addressed. The properties of the proposed uncertainty measure are analyzed through three different synthetic examples, *absence of data*, *heavy measurement noise*, and *composition of functions* scenarios. We show that each case can be distinguished using the proposed uncertainty measure and presented an uncertainty-aware learning from demonstration method for autonomous driving using this property. The proposed uncertainty-aware learning from demonstration method outperforms other compared methods in terms of safety using a complex real-world driving dataset.

I. Introduction

Recently, deep learning has been successfully applied to a diverse range of research areas including computer vision [1], natural language processing [2], and robotics [3]. When deep networks are kept in a cyber environment without interacting with an actual physical system, mis-predictions or malfunctioning of the system may not cause catastrophic disasters. However, when using deep learning methods in a real physical system involving human beings, such as an autonomous car, safety issues must be considered appropriately [4].

In May 2016, in fact, a fatal car accident had occurred due to the malfunctioning of a low-level image processing component of an advanced assisted driving system (ADAS) due to its inability to discriminate the white side of a trailer from a bright sky [5]. In this regard, Kendall and Gal [6] proposed an uncertainty modeling method for deep learning estimating both *aleatoric* and *epistemic* uncertainties indicating noise inherent in the data generating process and uncertainty in the predictive model which captures the ignorance about the model. However, computationally-heavy Monte Carlo sampling is required, making it unsuitable for real-time applications.

In this paper we present a novel uncertainty estimation method for a regression task using a deep neural network and its application to learning from demonstration (LfD). Specifically, a mixture density network (MDN) [7] is used to model the underlying process which is more appropriate for describing complex distributions [8], e.g., human demonstrations. We first present an uncertainty modeling method when making a prediction with an MDN which can be acquired with a single MDN forward path without Monte Carlo sampling. This sampling-free property makes it suitable for real-time robotic applications compared to existing uncertainty modeling methods that require multiple models [9] or sampling [6], [10], [11]. Furthermore, as an MDN is appropriate for modeling complex distributions [12] compared to a density network used in [6], [12] or a standard neural network for regression, the experimental results on autonomous driving tell us that it can better model the underlying policy of a driver given complex and noisy demonstrations.

The main contributions of this paper are twofold. We first present a sampling-free uncertainty estimation method utilizing an MDN and show that it can be decomposed into two parts, *explained* and *unexplained* variances which indicate our ignorance about the model and measurement noise, respectively. The properties of the proposed uncertainty modeling method are analyzed through three different cases: *absence of data*, *heavy measurement noise*, and *composition of functions* scenarios. Using the analysis, we further propose an uncertainty-aware learning from demonstration (LfD) method. We first train an aggressive controller in a simulated environment with an MDN and use the *explained* variance of an MDN to switch its mode to a conservative rule-based controller. When applied to a complex real-world driving dataset from the US Highway 101 [13], the proposed uncertainty-aware LfD outperforms compared methods in terms of safety against out-of-distribution inputs.

The remainder of this paper is composed as follows. Related work and preliminaries regarding modeling uncertainty in deep learning are introduced in Section II and Section III. The proposed uncertainty modeling method with an MDN is presented in Section IV and analyzed in Section V. Finally, in Section VI, we present an uncertainty-aware learning from demonstration method and successfully apply to an autonomous driving task using a real-world driving dataset by deploying and controlling a virtual car on the road.

II. Related Work

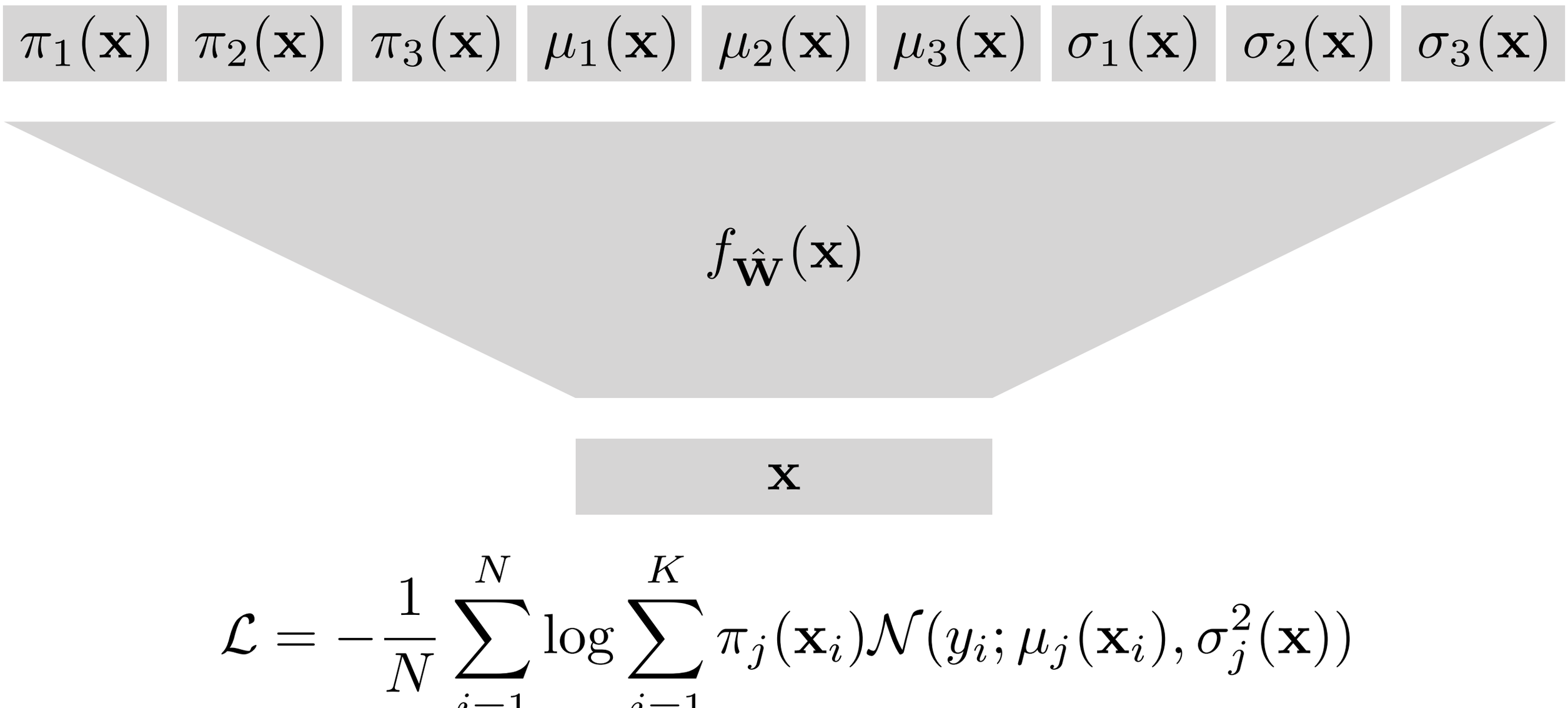
Despite the heavy successes in deep learning, practical methods for estimating uncertainties in the predictions with deep networks have only recently become actively studied. In the seminal study of Gal and Ghahramani [10], a practical method of estimating the predictive variance of a deep neural network is proposed by computing the sample mean and variance of stochastic forward paths, i.e., dropout [14]. The main contribution of [10] is to present a connection between an approximate Bayesian network and a sparse Gaussian process. This method is often referred to as the Monte Carlo

S. Choi, K. Lee, and S. Oh are with the Department of Electrical and Computer Engineering and ASRI, Seoul National University, Seoul 08826, Korea (emails: {sungjoon.choi, kyungjae.lee, songhwai.oh}@cpslab.snu.ac.kr). S. Lim is with the Department of Mathematics, Korea University (email: songbin@korea.ac.kr).

Will be presented at ICRA 2018

Choi et al. (2017)

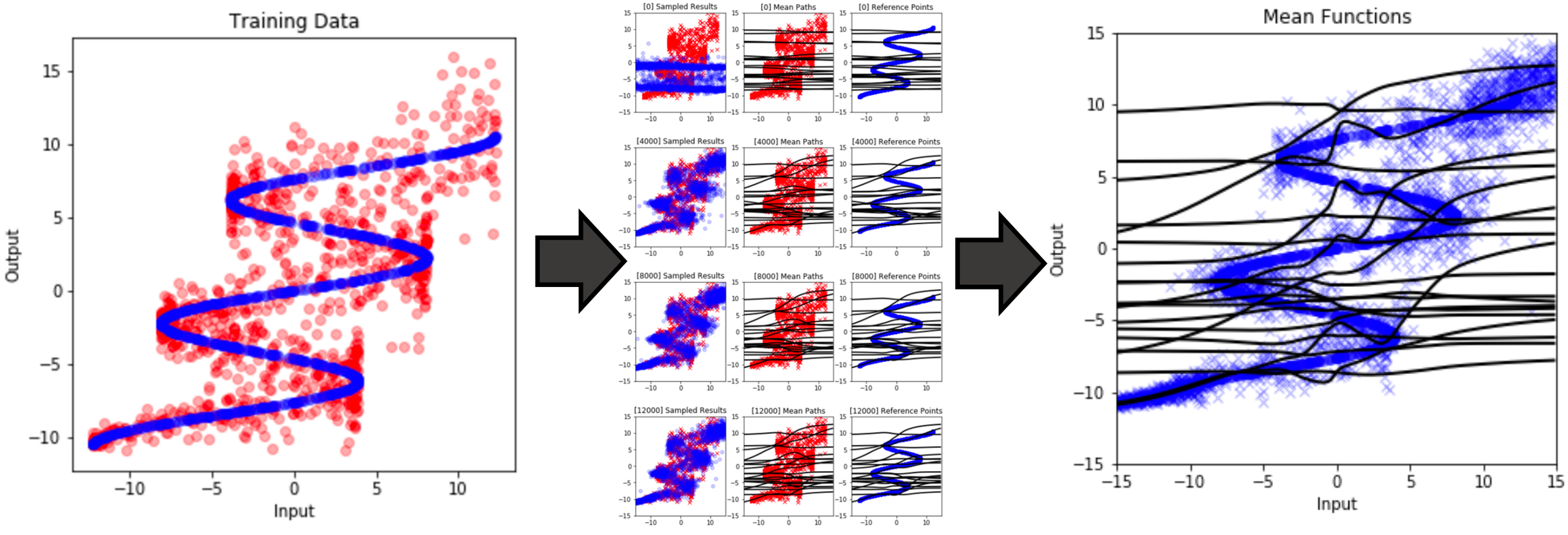
Mixture density networks



$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\log\sum_{j=1}^{K}\pi_j(\mathbf{x}_i)\mathcal{N}(y_i;\mu_j(\mathbf{x}_i),\sigma_j^2(\mathbf{x}))$$

Choi et al. (2017)

Mixture density networks



Choi et al. (2017)

Explained and unexplained variance

unexplained variance

$$\mathbb{V}(\mathbf{y}|\mathbf{x}) = \sum_{j=1}^{K} \pi_j(\mathbf{x})\Sigma_j(\mathbf{x}) + \sum_{j=1}^{K} \pi_j(\mathbf{x}) \left\| \mu_j(\mathbf{x}) - \sum_{k=1}^{K} \pi_k(\mathbf{x})\mu_k(\mathbf{x}) \right\|^2 .$$

explained variance

"We propose a sampling-free variance modeling method using a mixture density network which can be decomposed into *explained variance* and *unexplained variance*."

Choi et al. (2017)

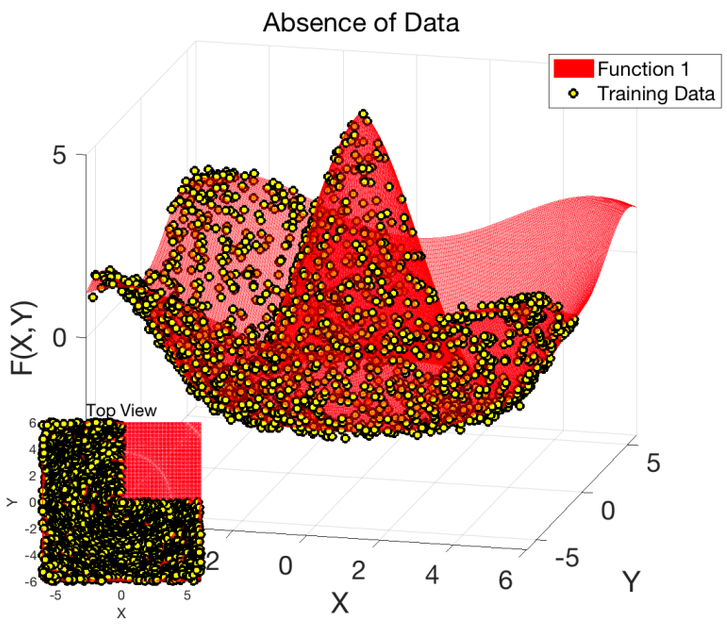
Explained and unexplained variance

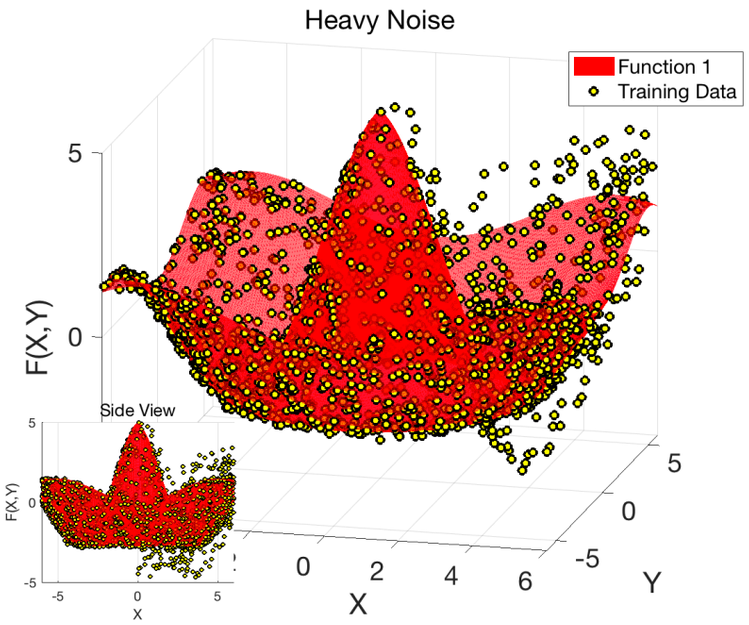
$$\mathbb{V}(\mathbf{y}|\mathbf{x}) = \overbrace{\sum_{j=1}^{K} \pi_j(\mathbf{x})\Sigma_j(\mathbf{x})}^{\textit{unexplained}\text{ variance}} + \underbrace{\sum_{j=1}^{K} \pi_j(\mathbf{x}) \left\| \mu_j(\mathbf{x}) - \sum_{k=1}^{K} \pi_k(\mathbf{x})\mu_k(\mathbf{x}) \right\|^2}_{\textit{explained}\text{ variance}}.$$

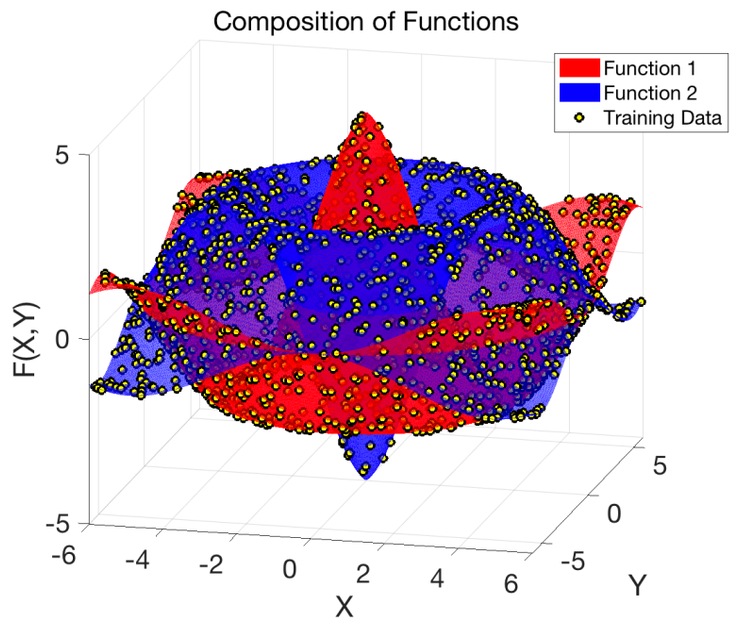
"In particular, *explained* variance represents model uncertainty whereas *unexplained* variance indicates the uncertainty inherent in the process, e.g., measurement noise."

Choi et al. (2017)

Analysis with Synthetic Examples



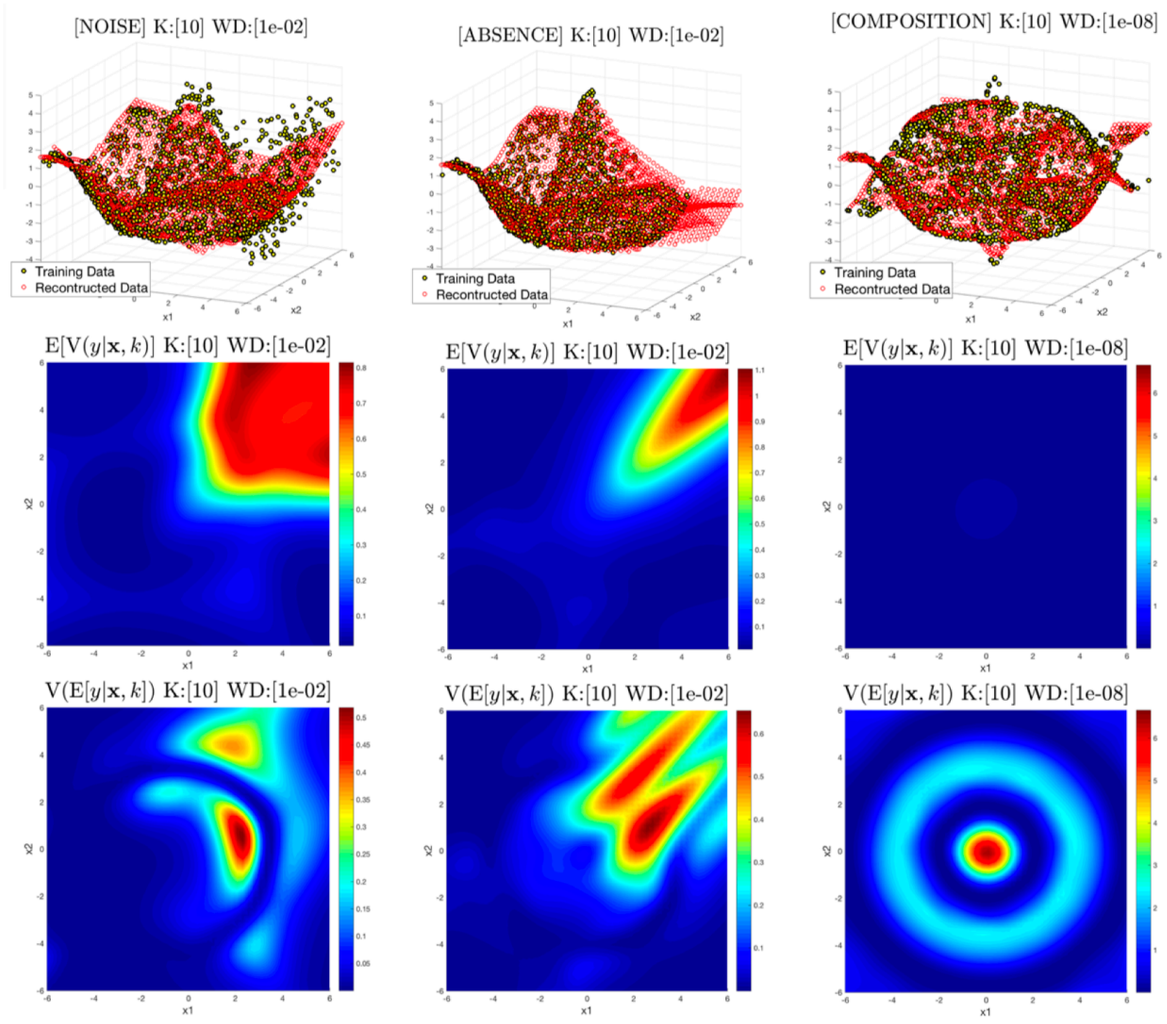




The proposed uncertainty modeling method is analyzed in three different synthetic examples: 1) absence of data, 2) heavy noise, and 3) composition of functions.

Choi et al. (2017)

Analysis with Synthetic Examples



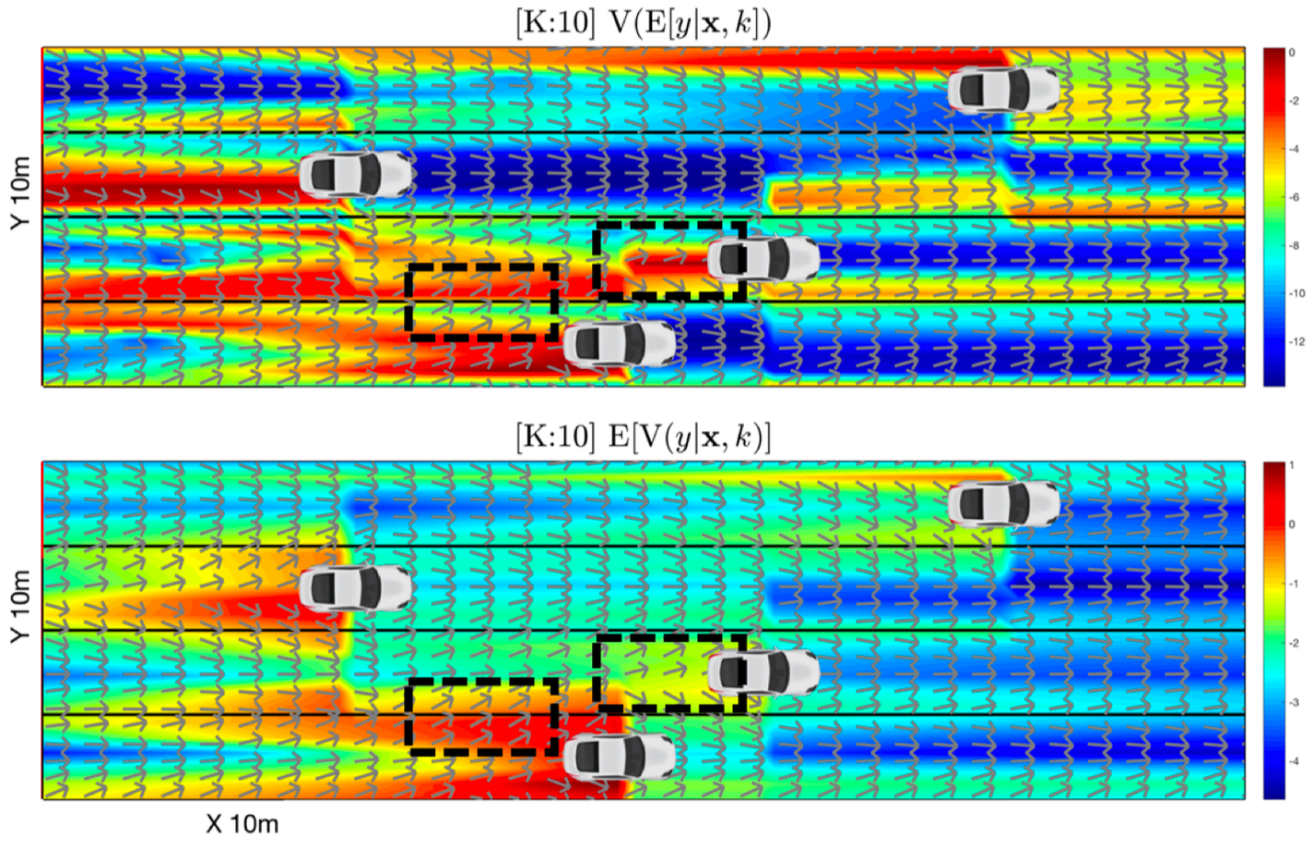
	$\mathbb{E}[\mathbb{V}(y\|\mathbf{x},k)]$	$\mathbb{V}(\mathbb{E}[y\|\mathbf{x},k])$
Heavy noise	High	High or Low
Absence of data	High	High
Composition of functions	Low	High

Choi et al. (2017)

Explained and unexplained variance

Explained Variance

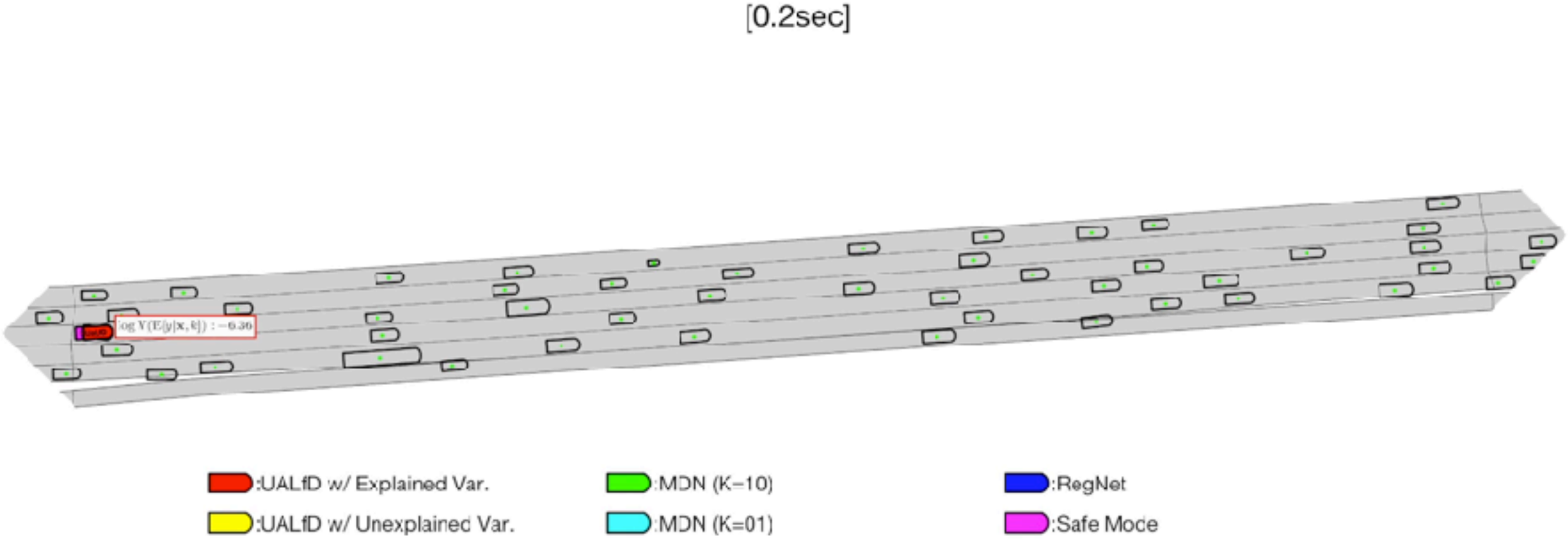
Unexplained Variance



We present uncertainty-aware learning from demonstration by using the *explained variance* as a switching criterion between trained policy and rule-based safe mode.

Choi et al. (2017)

Driving experiments



Thank you for your attention.

Contact information (sungjoon.choi@cpslab.snu.ac.kr)