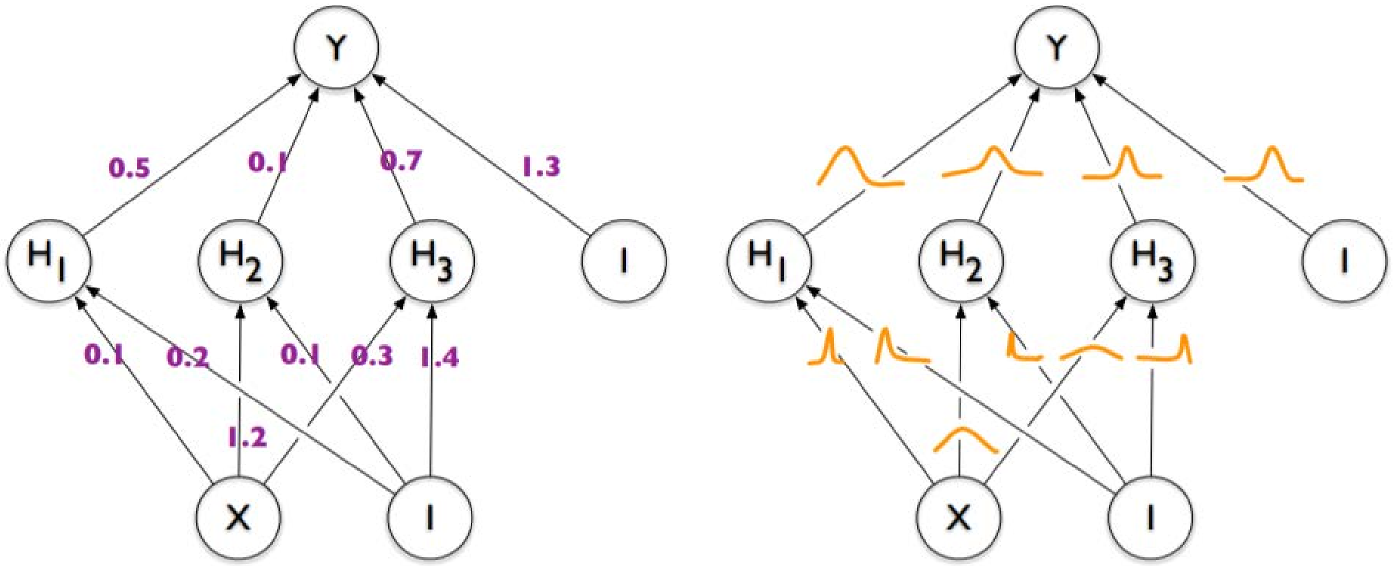
Uncertainty Estimation (UE) for CV

MIAN HUANG

Deterministic v.s Probabilistic



Ref: Weight uncertainty in neural networks. Charles Blundell et.al. ICML 2015.

Importance of uncertainty

1. High-risk tasks: Making mistakes is intolerable.

e.g. Medical images analysis, Autonomous driving, Aerospace engineering, etc.

2. Active learning: To determine which samples to be annotated.

3. Reinforcement learning: Exploitation-exploration dilemma.

Uncertainties in ML

Aleatoric/Data (*x*) uncertainty:

Inherent data noise.

Epistemic/Model (*w*) uncertainty:

Insufficient training data, out-of-distribution.

Data uncertainty

Homoscedastic (traditional): constant noise for each sample.

$$p(y \,|\, \mathbf{x}, \mathbf{w}) = \mathcal{N}(y;\, f(\mathbf{x}; \mathbf{w}),\, \sigma_y^2)$$

Heteroscedastic: various noises for different samples.

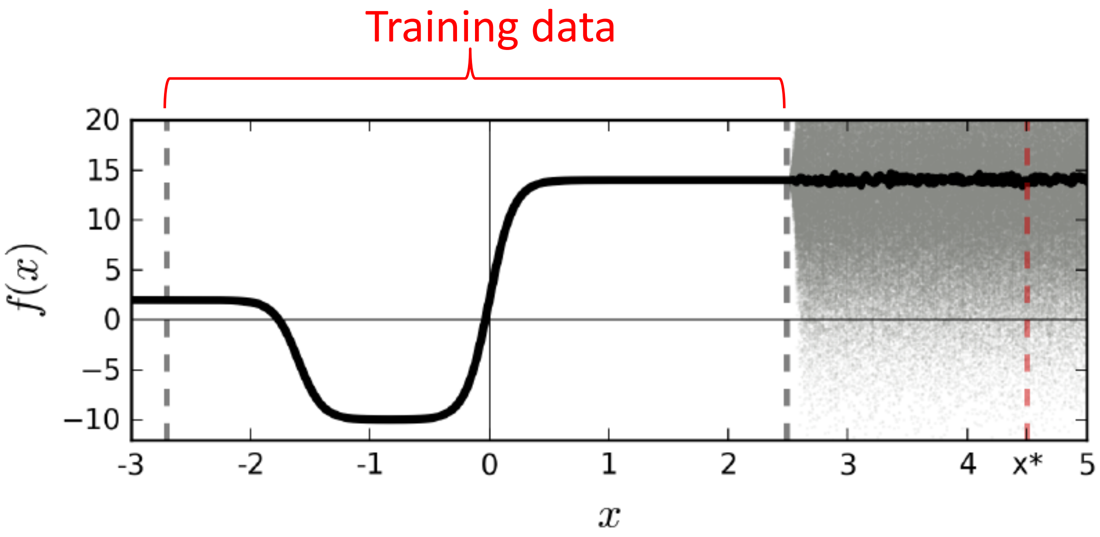
$$\mathcal{L}_{\text{NN}}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2\sigma(\mathbf{x}_i)^2} ||\mathbf{y}_i - \mathbf{f}(\mathbf{x}_i)||^2 + \frac{1}{2} \log \sigma(\mathbf{x}_i)^2$$

To learn the data-dependent noise parameter

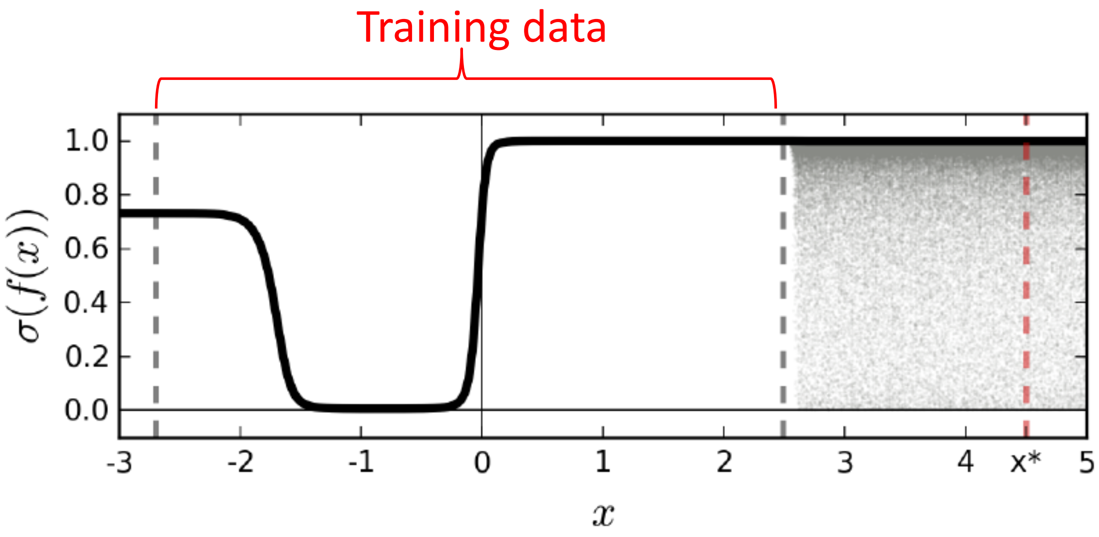
Residual ↑, Noise parameter ↑

Note: Here is only a point estimate, not Bayesian.

Model uncertainty



(a) Arbitrary function $f(\mathbf{x})$ as a function of data $\mathbf{x}$ (softmax *input*)



(b) $\sigma(f(\mathbf{x}))$ as a function of data $\mathbf{x}$ (softmax *output*)

Ref: Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. Yarin Gal and Zoubin Ghahramani. ICML 2016.

Model uncertainty

1. BNN posterior approximation.

Markov chain Monte-carlo (MCMC)

Variational inference: e.g. **MC-Dropout**

2. Model ensemble.

e.g. **Deep ensemble**

ICML 2016

Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning

Yarin Gal YG279@CAM.AC.UK
Zoubin Ghahramani ZG201@CAM.AC.UK
University of Cambridge

MC-Dropout

Bayesian prediction: $$p(\mathbf{y}|\mathbf{x}, \mathbf{X}, \mathbf{Y}) = \int p(\mathbf{y}|\mathbf{x}, \boldsymbol{\omega})p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})\mathrm{d}\boldsymbol{\omega}$$

Gaussian likelihood: $$p(\mathbf{y}|\mathbf{x}, \boldsymbol{\omega}) = \mathcal{N}\big(\mathbf{y}; \widehat{\mathbf{y}}(\mathbf{x}, \boldsymbol{\omega}), \tau^{-1}\mathbf{I}_D\big)$$

KL objective: $$-\int q(\boldsymbol{\omega}) \log p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega})\mathrm{d}\boldsymbol{\omega} + \mathrm{KL}(q(\boldsymbol{\omega})||p(\boldsymbol{\omega})).$$

$q(\boldsymbol{\omega})$

$$\underbrace{\mathbf{W}_i}_{1\times K_i} = \underbrace{\mathbf{M}_i}_{1\times K_i} \cdot \underbrace{\mathrm{diag}([\overset{1\times K_i}{\mathbf{z}_{i,j}}]_{j=1}^{K_i})}_{K_i\times K_i}$$

$$\mathbf{z}_{i,j} \sim \mathrm{Bernoulli}(p_i) \text{ for } \underset{\text{layer}}{i} = 1, \ldots, L, \; \underset{\text{unit}}{j} = 1, \ldots, K_{i-1}$$

Deep ensemble (bootstrapping)

Mean → prediction, variance → uncertainty

Parallel training

Algorithm 1 Pseudocode of the training procedure for our method

$\triangleright$ *Let each neural network parametrize a distribution over the outputs, i.e.* $p_\theta(y|\mathbf{x})$. *Use a proper scoring rule as the training criterion* $\ell(\theta, \mathbf{x}, y)$. *Recommended default values are* $M = 5$ *and* $\boldsymbol{\epsilon} = 1\%$ *of the input range of the corresponding dimension (e.g 2.55 if input range is [0,255]).*

Initialize $\theta_1, \theta_2, \ldots, \theta_M$ randomly

for $m = 1 : M$ **do** $\triangleright$ *train networks independently in parallel*

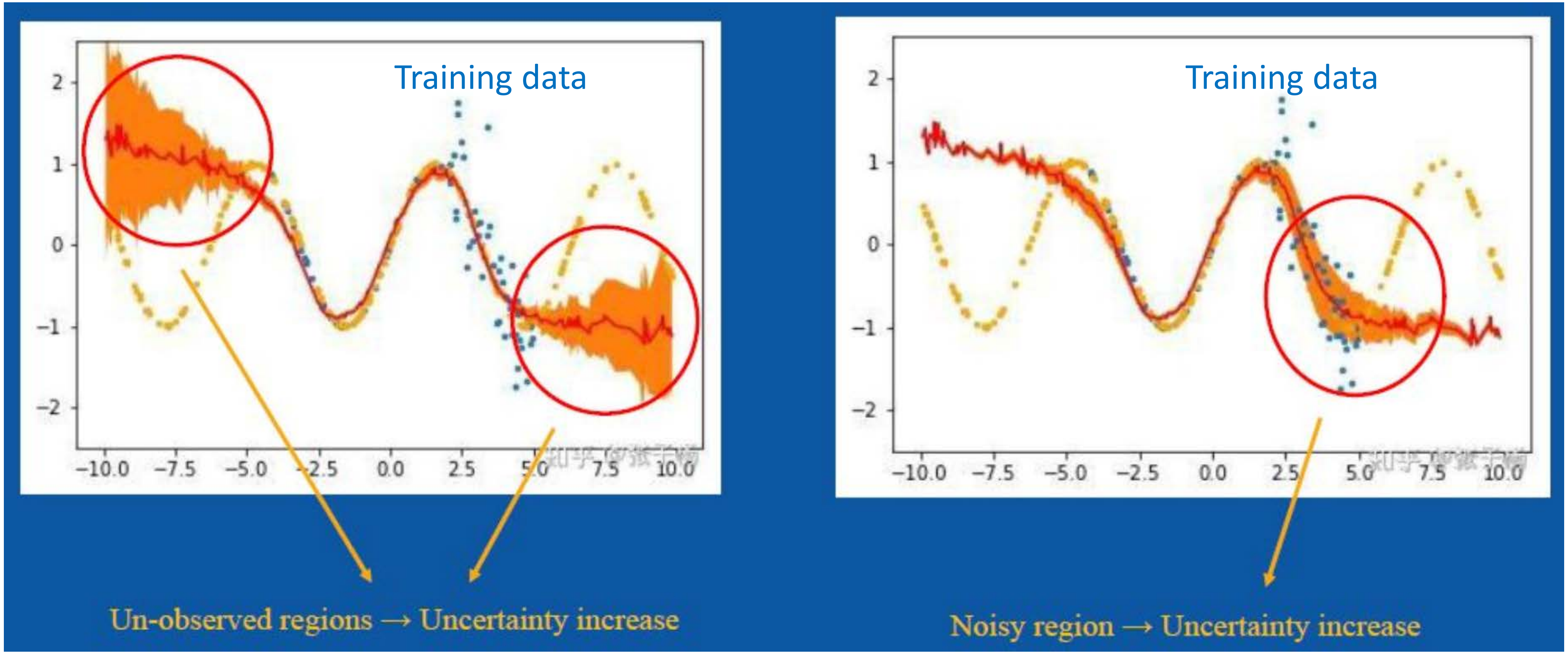
  Sample data point n_m randomly for each net $\triangleright$ *single* n_m *for clarity, minibatch in practice*

  Generate adversarial example using $\mathbf{x}'_{n_m} = \mathbf{x}_{n_m} + \boldsymbol{\epsilon}\,\mathrm{sign}\big(\nabla_{\mathbf{x}_{n_m}} \ell(\theta_m, \mathbf{x}_{n_m}, y_{n_m})\big)$

  Minimize $\ell(\theta_m, \mathbf{x}_{n_m}, y_{n_m}) + \ell(\theta_m, \mathbf{x}'_{n_m}, y_{n_m})$ w.r.t. θ_m $\triangleright$ *adversarial training (optional)*

Ref: Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. Balaji et.al. NIPS 2017

Epistemic (model) v.s Aleatoric (data)



UE development

UE has been studied for a long period.

But UE was only applied on CV until recent years (2017).

NIPS 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

Epistemic uncertainty estimation:

$$\widehat{\mathbf{W}} \sim q(\mathbf{W})$$

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

Combine epistemic and aleatoric (fixed Gaussian) uncertainties

High uncertainty → small effect on loss

$$\mathcal{L}_{BNN}(\theta) = \frac{1}{D}\sum_i \frac{1}{2}\hat{\sigma}_i^{-2}||\mathbf{y}_i - \hat{\mathbf{y}}_i||^2 + \frac{1}{2}\log\hat{\sigma}_i^2$$

UE for semantic segmentation

CamVid	IoU
SegNet [28]	46.4
FCN-8 [29]	57.0
DeepLab-LFOV [24]	61.6
Bayesian SegNet [22]	63.1
Dilation8 [30]	65.3
Dilation8 + FSO [31]	66.1
DenseNet [20]	66.9
This work:	
DenseNet (Our Implementation)	67.1
+ Aleatoric Uncertainty	67.4
+ Epistemic Uncertainty	67.2
+ Aleatoric & Epistemic	**67.5**

(a) CamVid dataset for road scene segmentation.

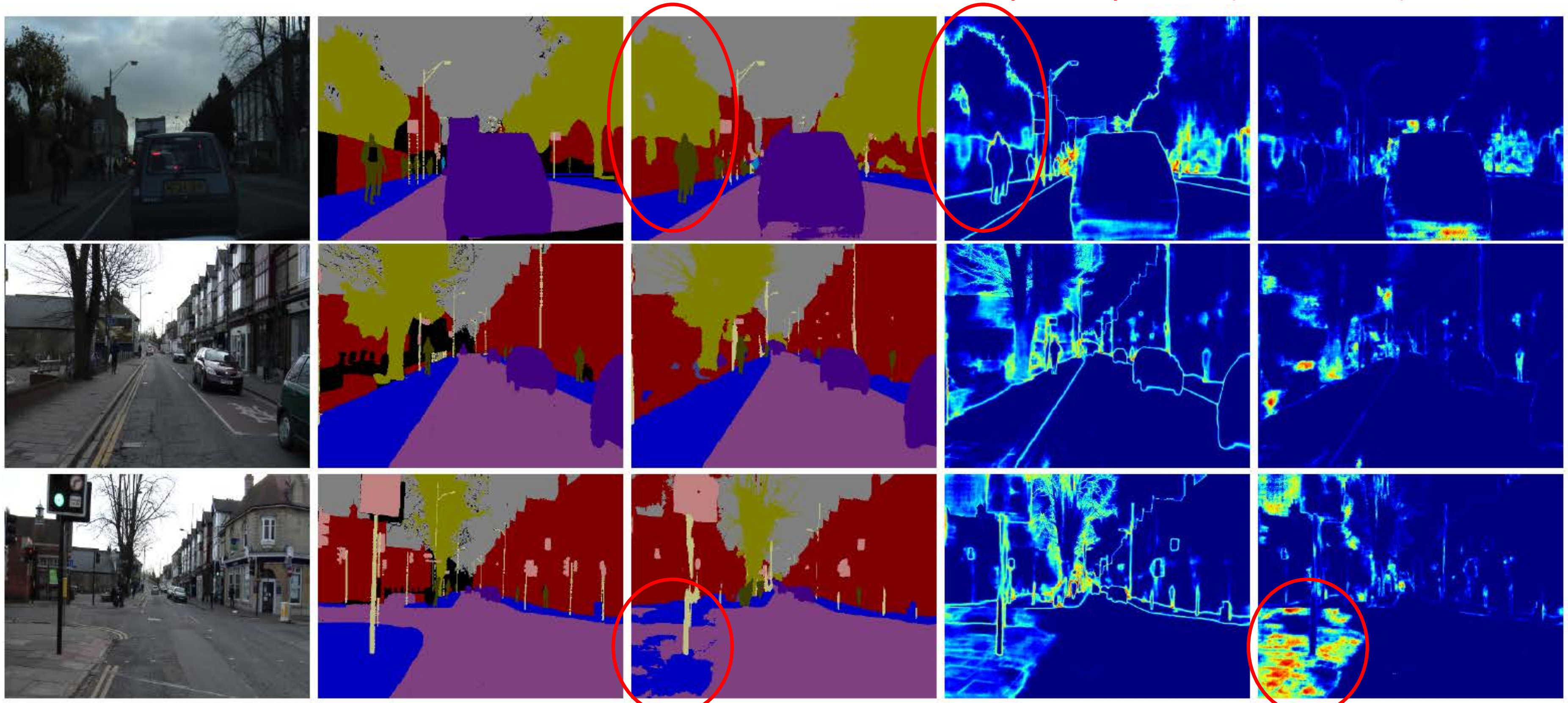
NYUv2 40-class	Accuracy	IoU
SegNet [28]	66.1	23.6
FCN-8 [29]	61.8	31.6
Bayesian SegNet [22]	68.0	32.4
Eigen and Fergus [32]	65.6	34.1
This work:		
DeepLabLargeFOV	70.1	36.5
+ Aleatoric Uncertainty	70.4	37.1
+ Epistemic Uncertainty	70.2	36.7
+ Aleatoric & Epistemic	**70.6**	**37.3**

(b) NYUv2 40-class dataset for indoor scenes.

Table 1: **Semantic segmentation performance.** Modeling both aleatoric and epistemic uncertainty gives a notable improvement in segmentation accuracy over state of the art baselines.

UE for semantic segmentation

Semantic boundary, noisy label (dominant)



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

OOD data (dominant)

UE for depth estimation

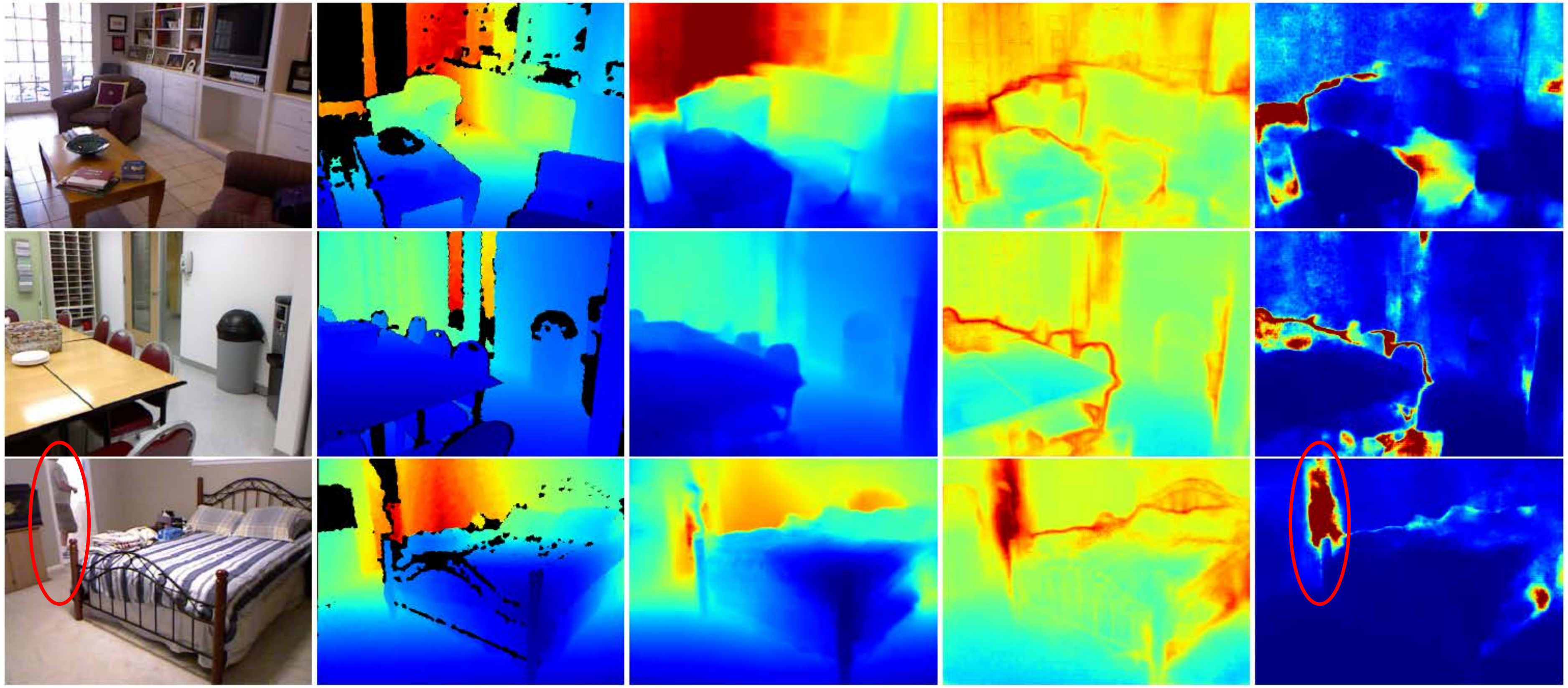


Figure 5: NYUv2 Depth results. From left: input image, ground truth, depth regression, aleatoric uncertainty, and epistemic uncertainty.

CVPR 2020

Scalable Uncertainty for Computer Vision with Functional Variational Inference

Eduardo D C Carvalho* Ronald Clark* Andrea Nicastro* Paul H J Kelly*

Motivation & Contributions

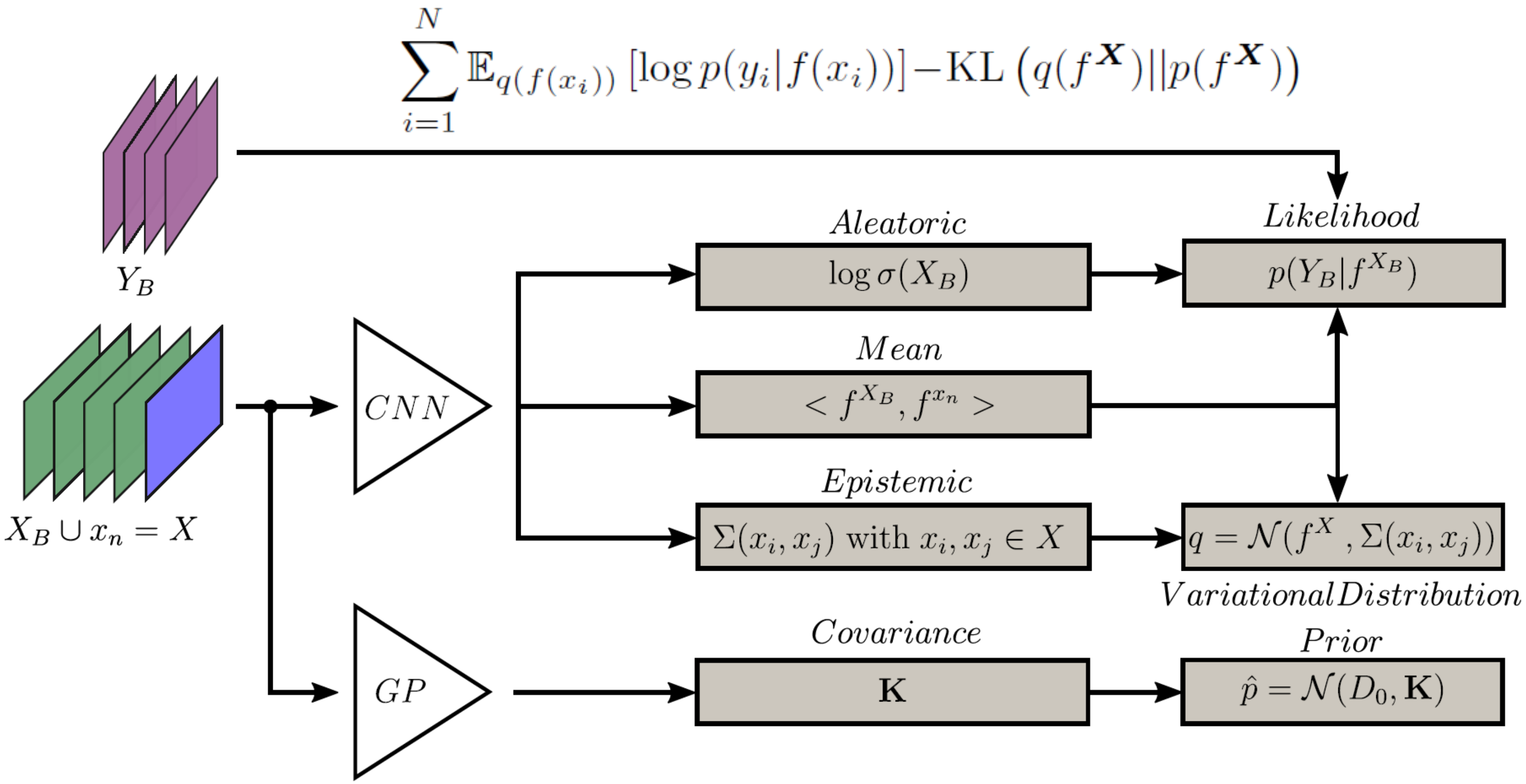
MC-Dropout requires multiple forward passes at test time to obtain model uncertainty.

Proposed Bayesian DL with fast one forward pass.

Obtained real-time uncertainty (both epistemic and aleatoric) estimation.

Scalable to high-d tasks.

Overview



Based on Functional VI, we use a parameterized GP as variational family and Bayesian CNN priors.

Functional VI objective [1]: $$\sum_{i=1}^{N} \mathbb{E}_{q(f(x_i))}\left[\log p(y_i|f(x_i))\right] - \mathrm{KL}\left(q(f^{\mathbf{X}})||p(f^{\mathbf{X}})\right)$$

Function space,
rather than weight space

Discrete likelihood for classification:
(Rescaled Boltzmann distribution)

$$p(y_k|f(x)) = \frac{\exp\left(f_k'(x)\right)}{\sum_{k=1}^{K} \exp\left(f_k'(x)\right)}$$

$$f_k'(x) = f_k(x)/\sigma_k^2(x)$$

Scaling to high-d tasks:

$$\begin{pmatrix} f(x_i) \\ f(x_j) \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} K(x_i,x_i) & K(x_i,x_j) \\ K(x_i,x_j) & K(x_j,x_j) \end{pmatrix}\right)$$

Multi-output GP as prior
Objective solved in closed-form

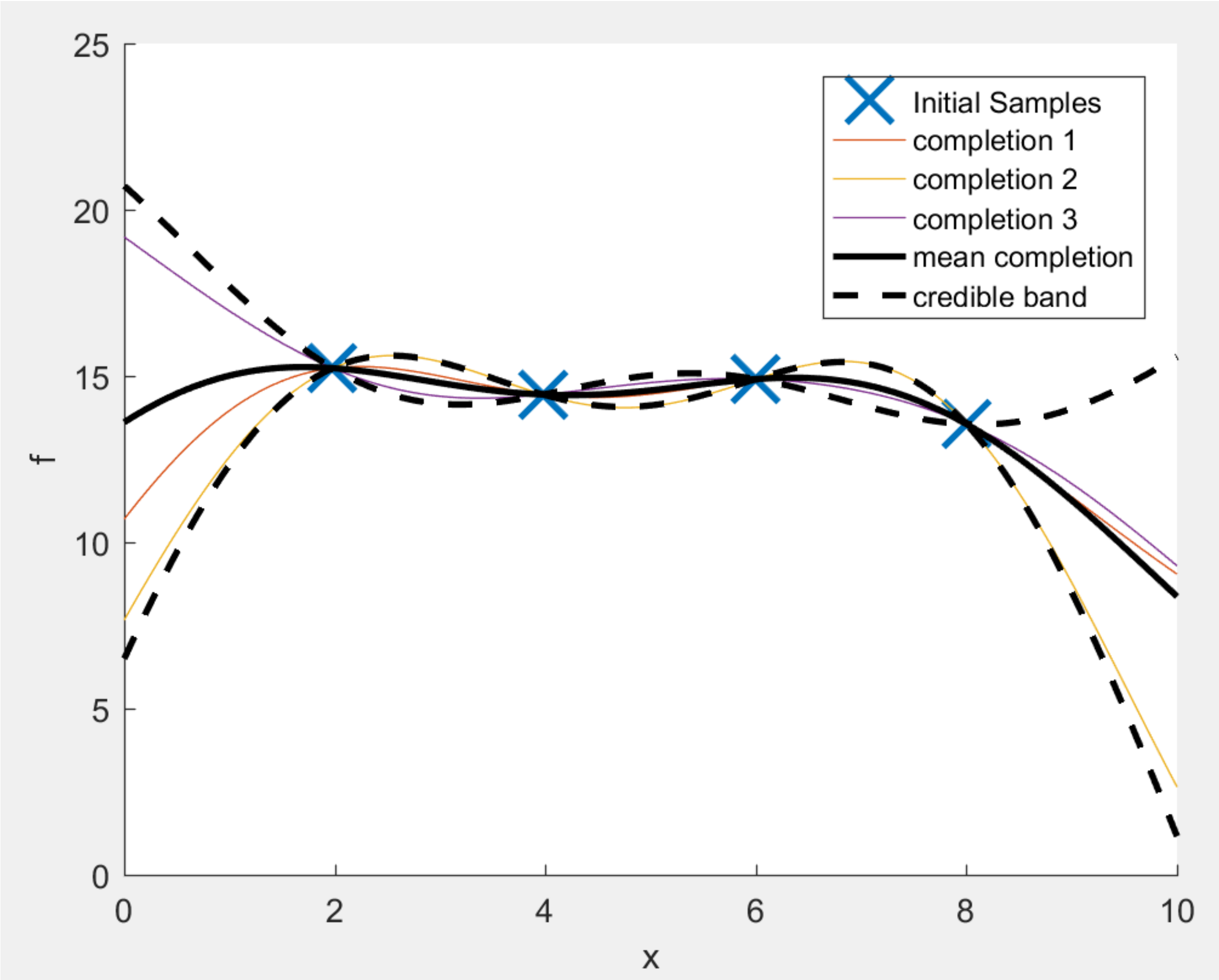
$$\begin{pmatrix} K_{1,1} & \cdots & K_{B,1} \\ \vdots & \ddots & \vdots \\ K_{B,1} & \cdots & K_{B,B} \end{pmatrix} \qquad \begin{pmatrix} \boldsymbol{K}_{:n,:n} & \boldsymbol{K}_{:n,n+1} \\ \boldsymbol{K}_{:n,:n+1}^T & K_{n+1,n+1} \end{pmatrix}$$

Reduce complexity

Epistemic covariance kernel:
(see detail in paper)

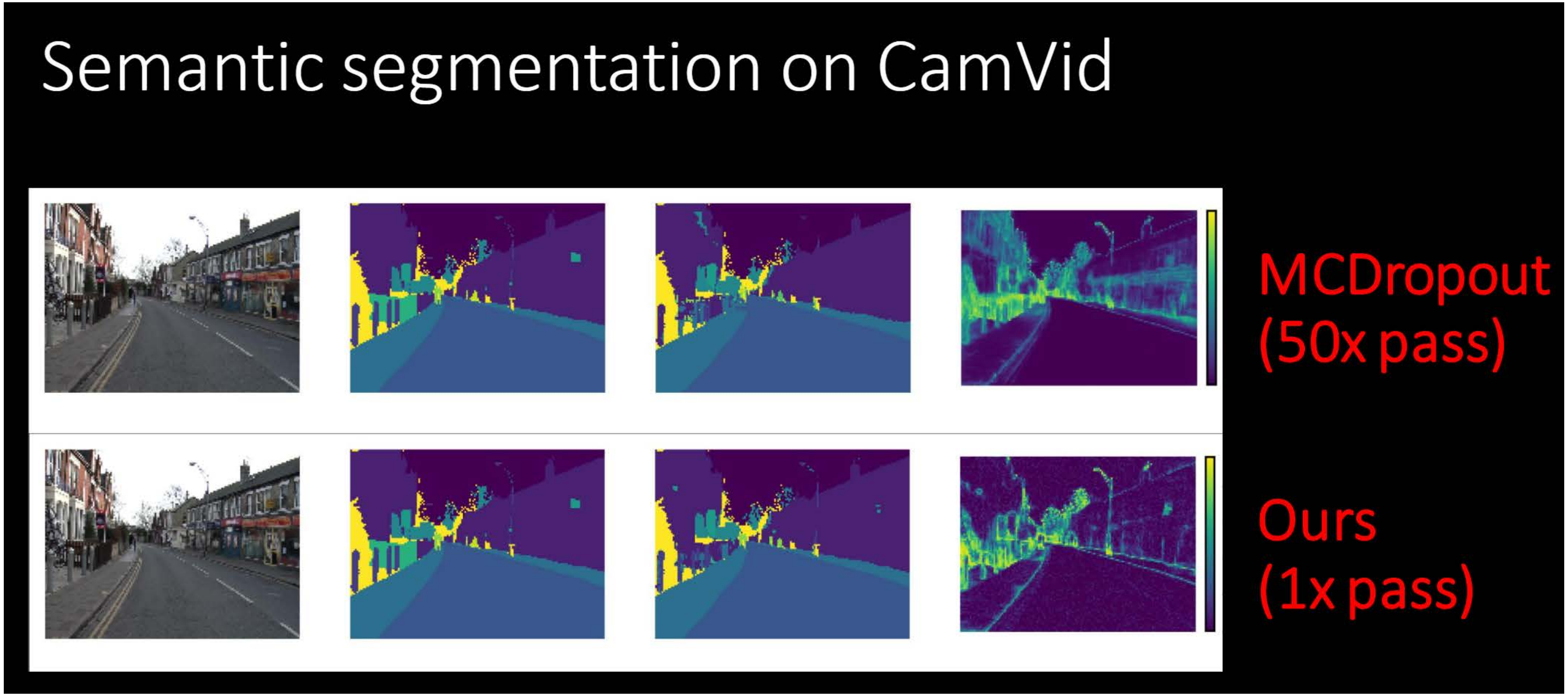
$$\Sigma(x_i,x_j) = \frac{1}{L}\sum_{k=1}^{L} g_k(x_i) \odot g_k(x_j) + D(x_i,x_j)\delta(x_i,x_j)$$

Ref: [1] FUNCTIONAL VARIATIONAL BAYESIAN NEURAL NETWORKS. Shengyang Sun et.al. ICLR 2019.



Initial Samples
completion 1
completion 2
completion 3
mean completion
credible band
f
x
0
5
10
15
20
25
0
2
4
6
8
10

Qualitative results



Quantitative results

Table 1. Results from training and testing on CamVid.

	IoU	**Accuracy**
Deterministic-Boltzmann	0.568	0.895
MCDropout-Boltzmann	0.556	0.893
Ours-Boltzmann	**0.623**	**0.905**

Table 2. Mean calibration score, computed with 10 equally spaced intervals, averaged over all test set examples. Lower is better.

	Mean Calibration
MCDropout-Boltzmann	0.058
Ours-Boltzmann	**0.053**

Table 6. Semantic segmentation on CamVid. Inference time comparison over 100 independent runs.

	mean $\pm$ std (ms)
Deterministic-Boltzmann	111.64 ± 0.27
MCDropout-Boltzmann	5763.63 ± 1.95
Ours-Boltzmann	128.59 ± 1.86

Thank you