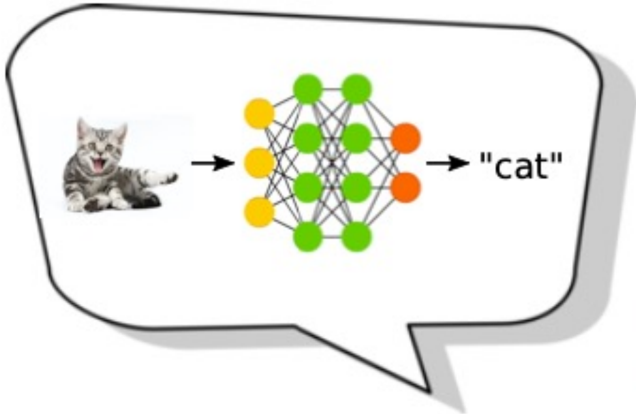
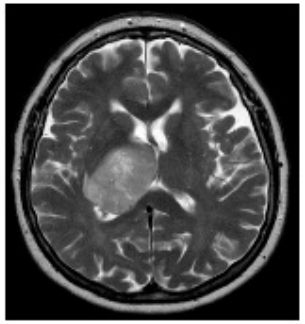
An Introduction to Bayesian Neural Networks

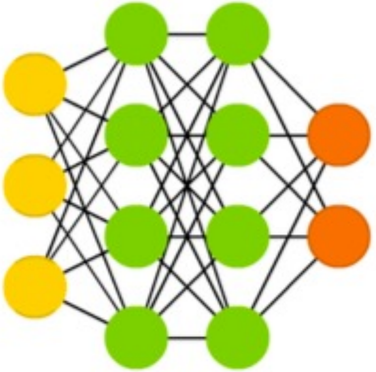
Yingzhen Li

yingzhen.li@imperial.ac.uk



Deep Learning





brain tumor
type "A"

are you sure? why?

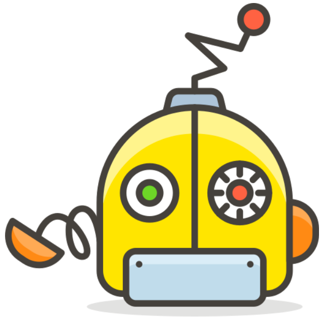
Do you know what
you don't know?

How confident are you?

Desiderata

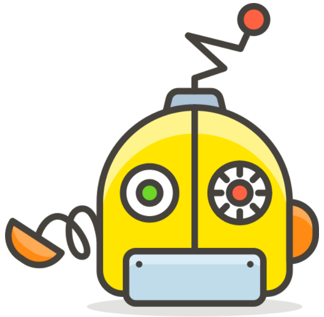
Here's this patient's health record:
…
Could you summarise it for me?

Here's a summary of the patient's health record you requested:
[point 1] with x% confidence (breakdown quantities)
…



Here's the conditions of this construction site:
…
Could you tell me what the potential safety issues are?

Here's potential safety issues that need to be look after:
[point 1] with x% confidence (breakdown quantities)
…



Bayesian Inference

$\pi(\theta) = p(\theta|data)$

$$P(\theta|\, data) = \frac{P(\theta)P(data\mid\theta)}{P(data)}$$

- $P(\theta)$: prior distribution
- $P(data \mid \theta)$: likelihood of θ given $data$
- $P(\theta|\, data)$: posterior distribution of θ given $data$
- $P(data)$: marginal likelihood/model evidence

$$P(data) = \int P(\theta)P(data \mid \theta)$$

Image courtesy of Sebastian Nowozin

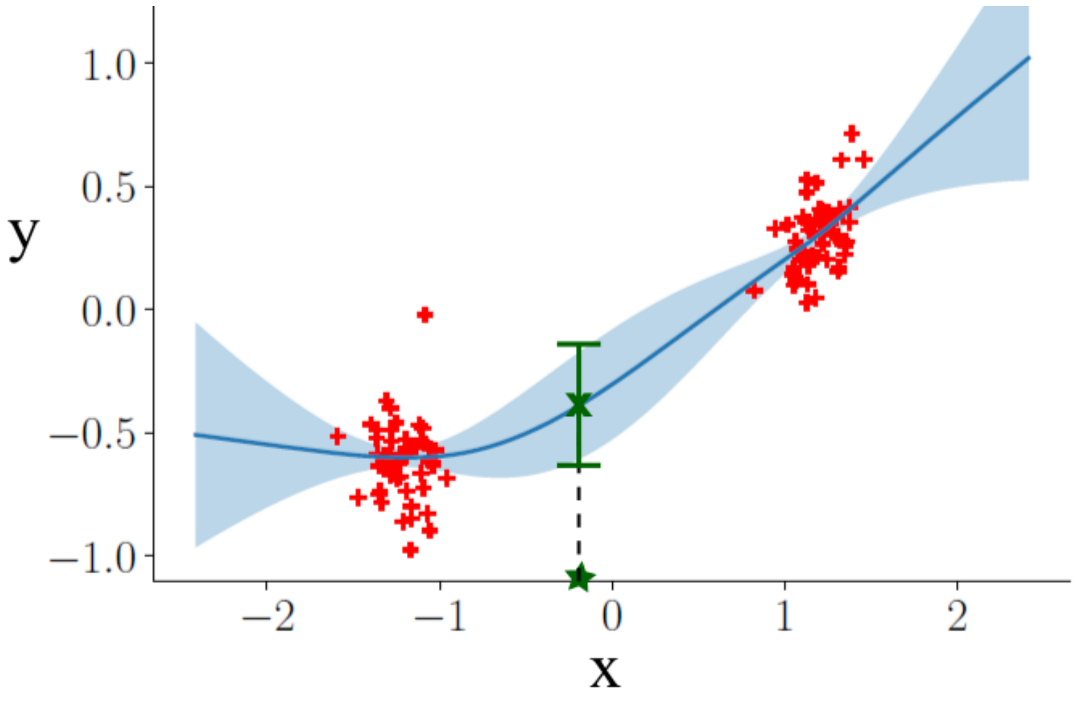
Bayesian Inference

- The central equation for Bayesian inference:

$$\int F(\theta)p(\theta|D)d\theta$$

"What is the prediction distribution of the test output given a test input?"

$F(\theta) = p(y|x,\theta)$,
D = observed datapoints

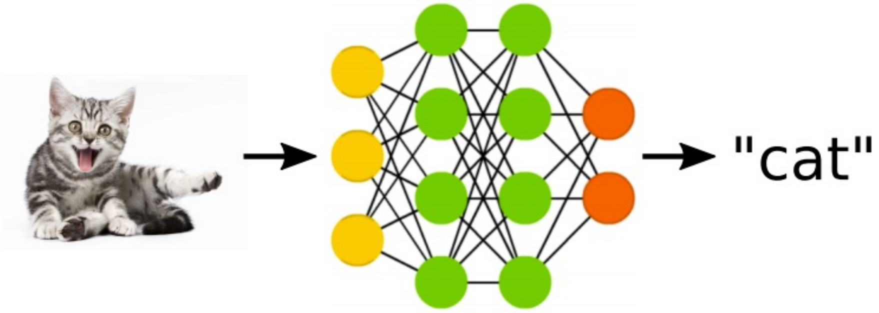


Bayesian Neural Network (BNN) 101

Classifying different types of animals:

- x: input image; y: output label
- Build a neural network with parameters θ:

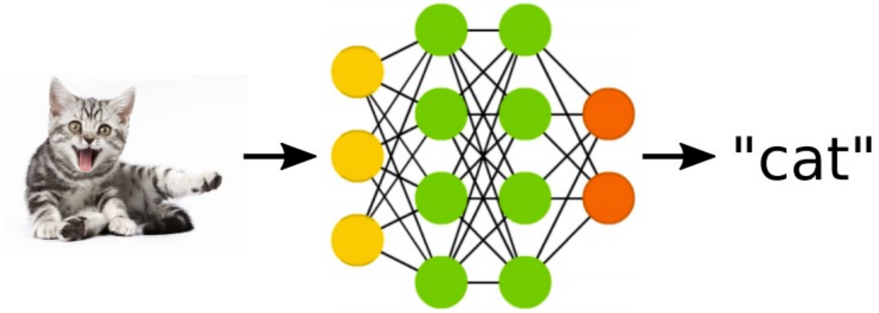
$$p(y|x,\theta) = softmax(f_\theta(x))$$



Bayesian Neural Network (BNN) 101

Classifying different types of animals:
- x: input image; y: output label
- Build a neural network with parameters θ:

$$p(y|x,\theta) = softmax(f_\theta(x))$$



A typical neural network (with non-linearity $g(\cdot)$):

$$f_\theta(x) = W^L g\left(W^{L-1}\, g\left(\dots g(W^1\, x + b^1)\right) + b^{L-1}\right) + b^L,$$

$$h^l = g(W^l\, h^{l-1} + b^l),\, h^1 = g(W^1\, x + b^1).$$

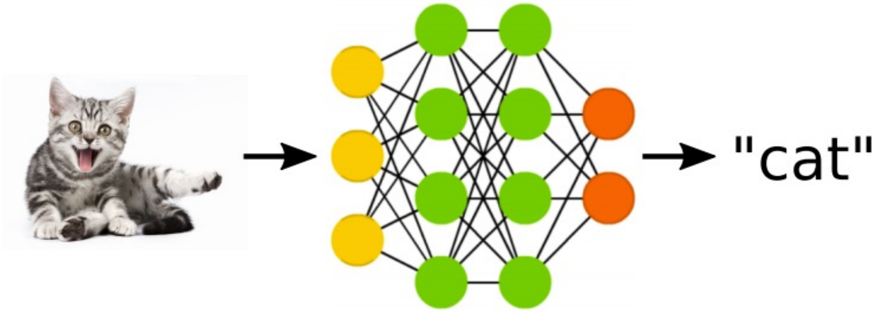
Neural network parameters: $\theta = \{W^l, b^l\}_{l=1}^{L}$

Bayesian Neural Network (BNN) 101

Classifying different types of animals:

- x: input image; y: output label
- Build a neural network with parameters θ:

$$p(y|x,\theta) = softmax(f_\theta(x))$$



Typical deep learning solution:

- Optimize θ to obtain a point estimates (MLE):

$$\theta^* = argmax \ \log p(D \mid \theta) \ ,$$
$$\log p(D \mid \theta) = \sum_{n=1}^{N} \log p(y_n \mid x_n, \theta) \ , D = \{(x_n, y_n)\}_{n=1}^{N}$$

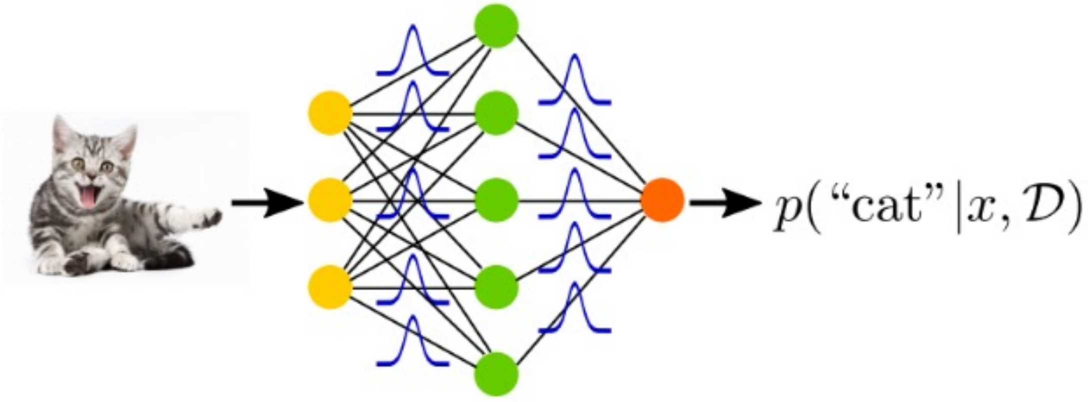
- Prediction: using $p(y^* \mid x^*, \theta^*)$

Bayesian Neural Network (BNN) 101

Classifying different types of animals:

- x: input image; y: output label
- Build a neural network with parameters θ:

$$p(y|x,\theta) = softmax(f_\theta(x))$$



Bayesian solution:

- Put a prior $p(\theta)$ on network parameters θ, e.g. Gaussian prior

$$p(\theta) = N(\theta; 0, \sigma^2 I)$$

- Compute the posterior distribution $p(\theta \mid D)$:

$$p(\theta \mid D) \propto p(D \mid \theta)\, p(\theta)$$

- Bayesian predictive inference:

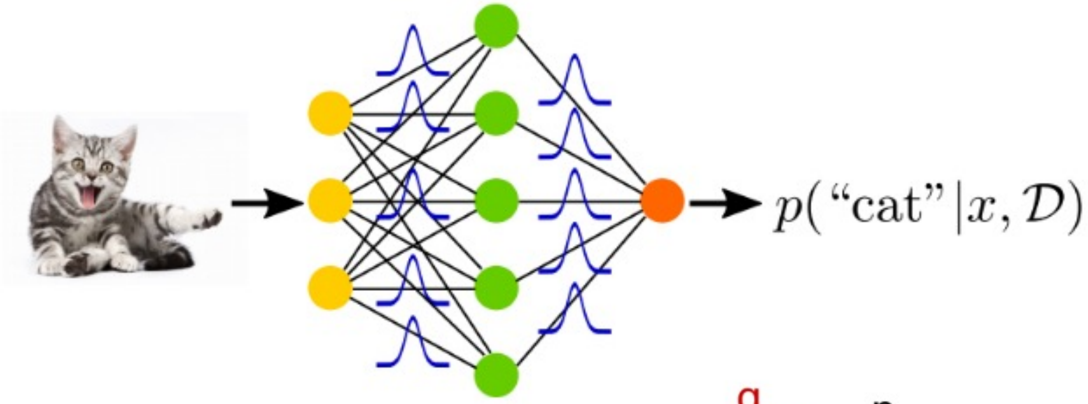
$$p(y^* \mid x^*, D) = E_{p(\theta \mid D)}[p(y^* \mid x^*, \theta)]$$

Bayesian Neural Network (BNN) 101

Classifying different types of animals:

- x: input image; y: output label
- Build a neural network with parameters θ:

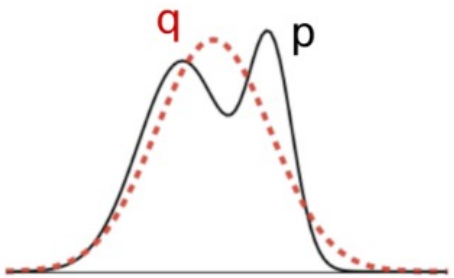
$$p(y|x,\theta) = softmax(f_\theta(x))$$



Approximate (Bayesian) inference solution:

- Exact posterior intractable, use approximate posterior:

$$q(\theta) \approx p(\theta \mid D)$$



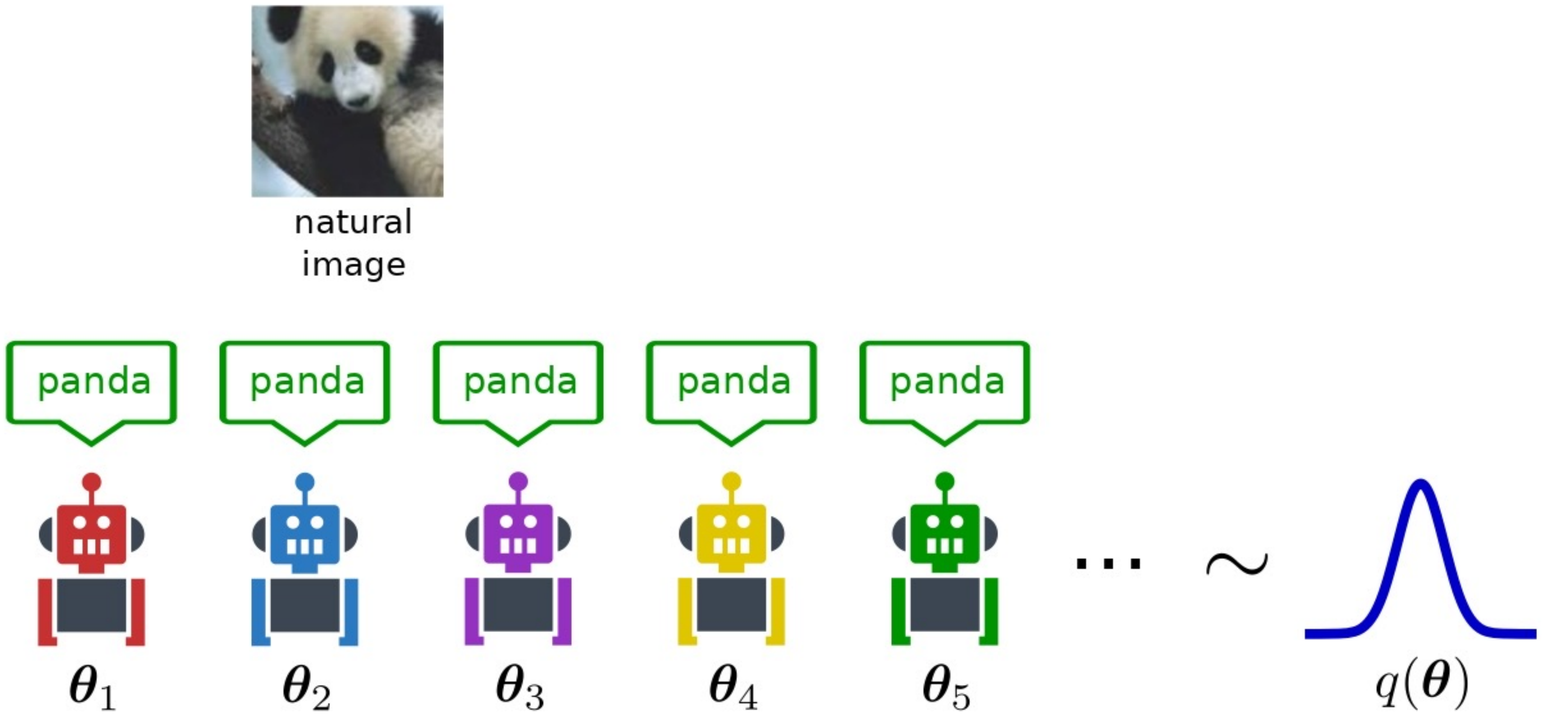
- Approximate Bayesian predictive inference:

$$p(y^* \mid x^*, D) \approx E_{q(\theta)}[p(y^* \mid x^*, \theta)]$$

- Monte Carlo approximation:

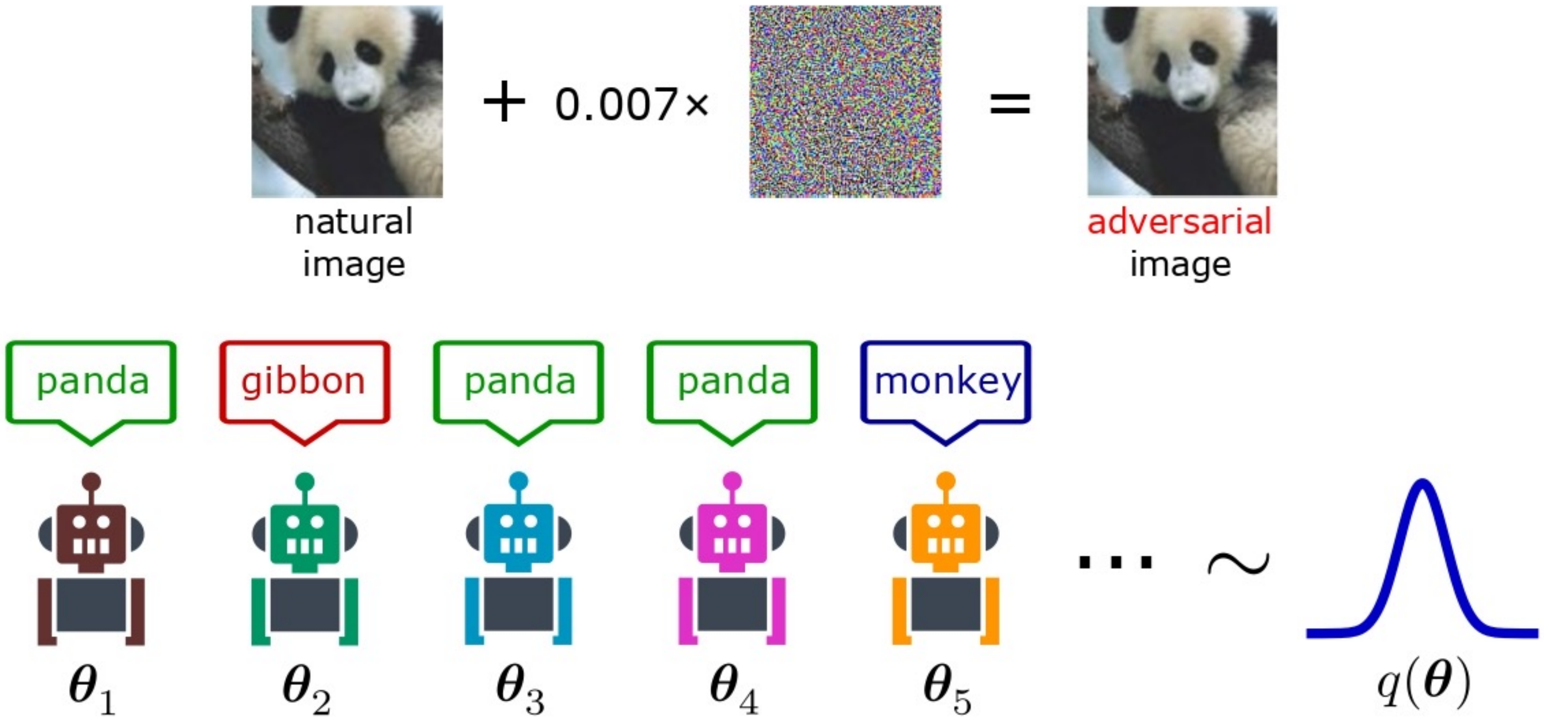
$$p(y^* \mid x^*, D) \approx \frac{1}{K}\sum_{k=1}^{K} p(y^* \mid x^*,\ \theta_k), \quad \theta_k \sim q(\theta)$$

Bayesian Neural Network (BNN) 101



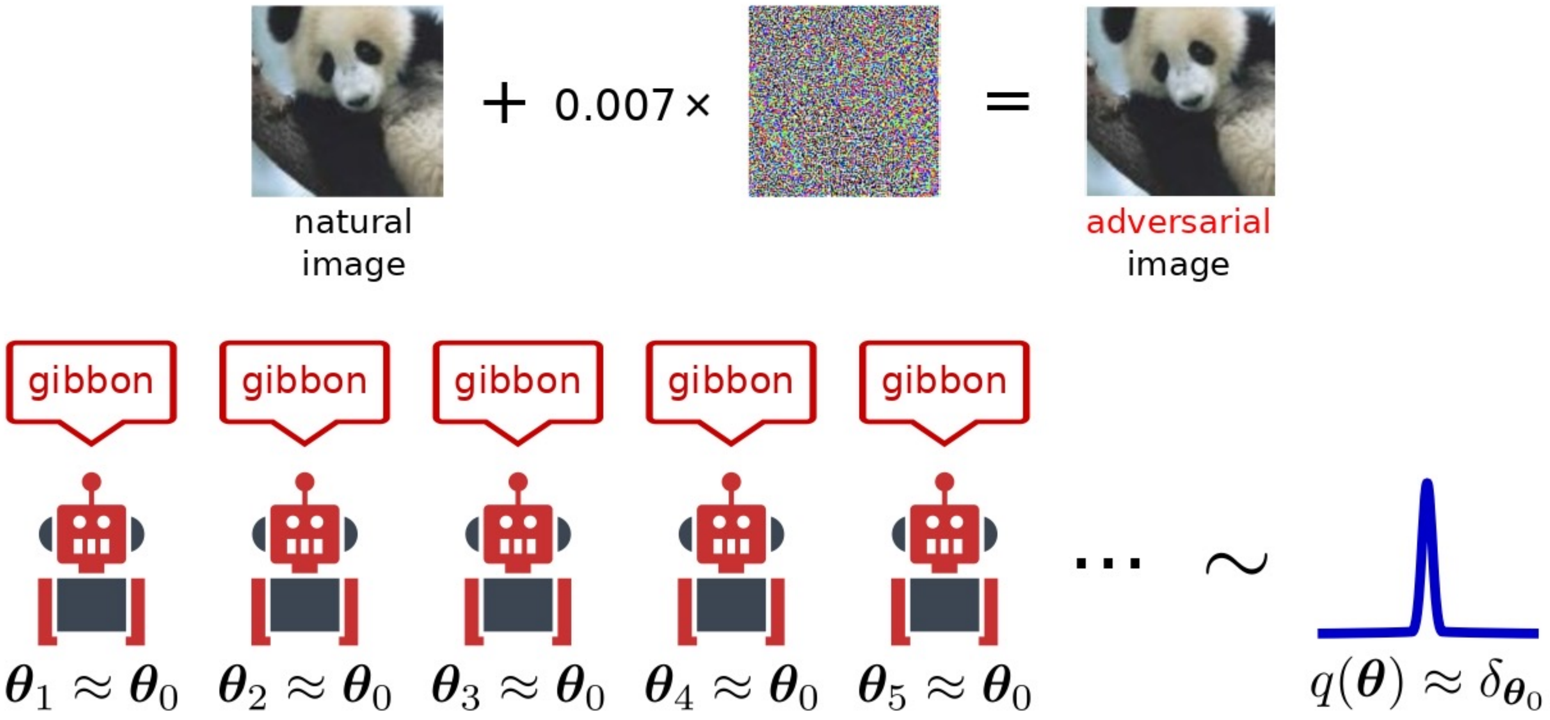
Prediction on in-distribution data:
ensemble over networks, using weights sampled from $q(\theta)$

Bayesian Neural Network (BNN) 101



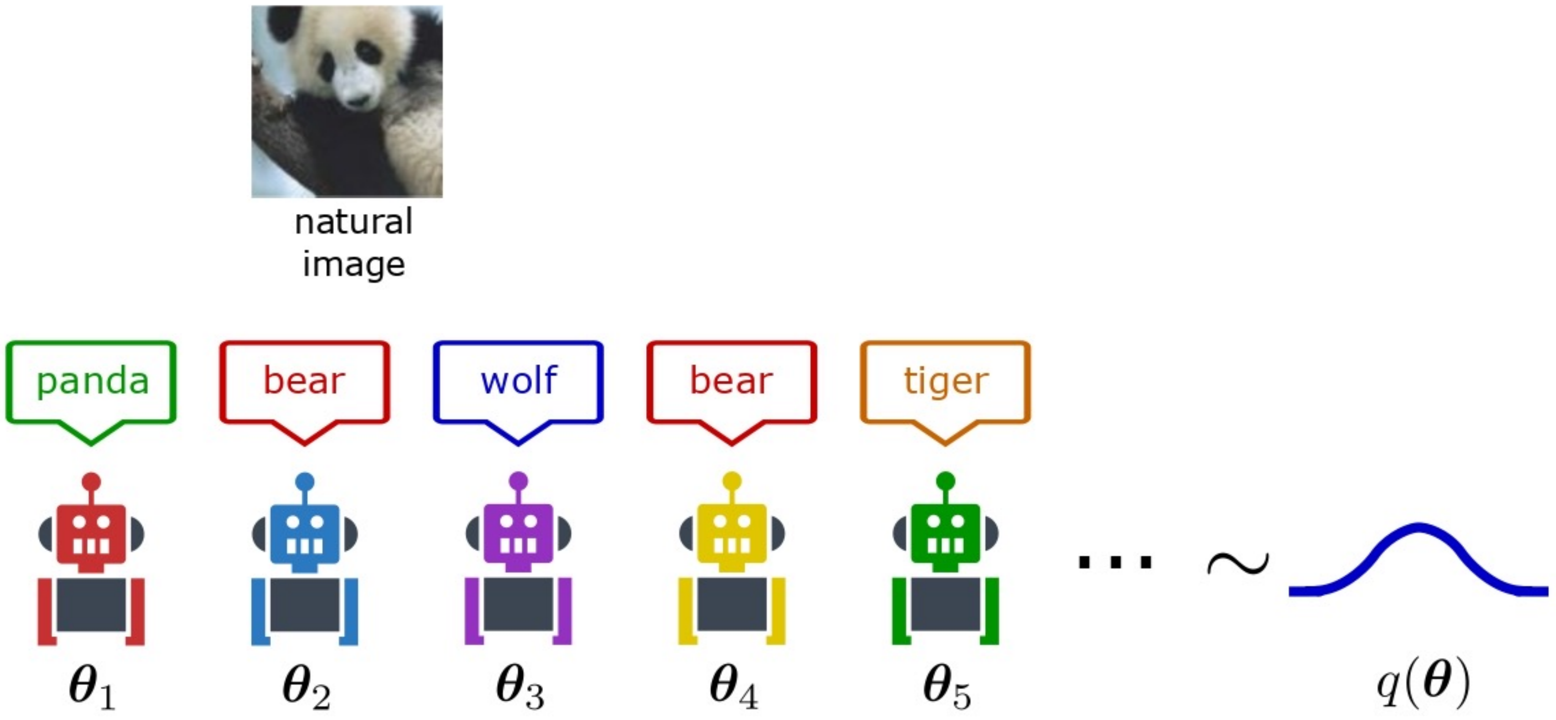
Prediction on OOD/noisy/adversarial data:
Disagreement (i.e. uncertainty) exists over networks sampled from $q(\theta)$

Bayesian Neural Network (BNN) 101



Prediction on OOD/noisy/adversarial data when $q(\theta)$ is over-confident:
Return confidently wrong answers (close to point estimate)

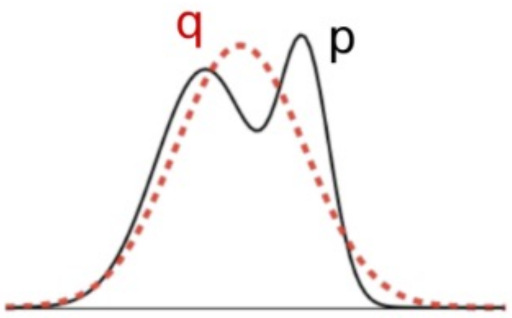
Bayesian Neural Network (BNN) 101



Prediction on in-distribution data when $q(\theta)$ is under-confident:
Low accuracy in prediction tasks (less desirable)

Approximate Inference in BNNs

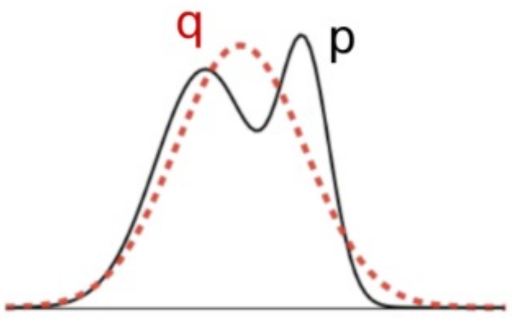
- Key steps of approximate inference in BNNs
  1. Construct the $q(\theta) \approx p(\theta \mid D)$ distribution
     - Simple distributions: e.g. Mean-field Gaussian
     - Structured approximations, e.g. low-rank Gaussians
     - Others (non-Gaussian)



See my NeurIPS 2020 tutorial on approximate inference

Approximate Inference in BNNs

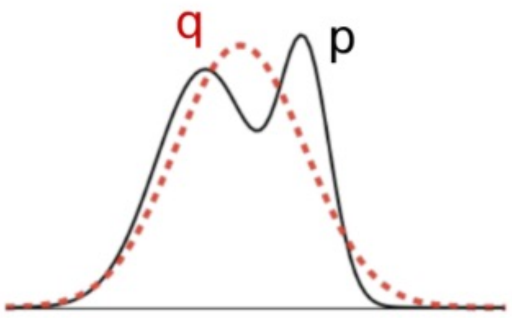
- Key steps of approximate inference in BNNs
  1. Construct the $q(\theta) \approx p(\theta \mid D)$ distribution
     - Simple distributions: e.g. Mean-field Gaussian
     - Structured approximations, e.g. low-rank Gaussians
     - Others (non-Gaussian)
  2. Fit the $q(\theta)$ distribution
     - E.g. with variational inference

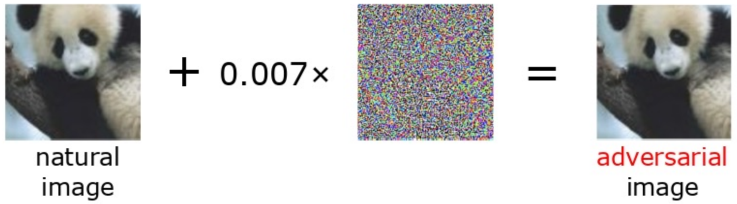


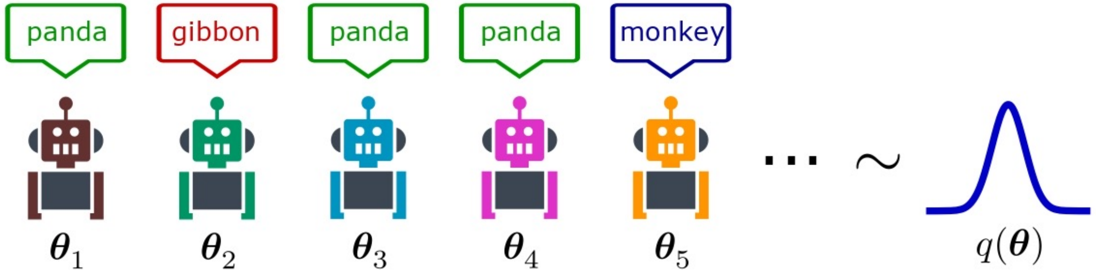
See my NeurIPS 2020 tutorial on approximate inference

Approximate Inference in BNNs

- Key steps of approximate inference in BNNs
  1. Construct the $q(\theta) \approx p(\theta \mid D)$ distribution
     - Simple distributions: e.g. Mean-field Gaussian
     - Structured approximations, e.g. low-rank Gaussians
     - Others (non-Gaussian)
  2. Fit the $q(\theta)$ distribution
     - E.g. with variational inference
  3. Compute prediction with Monte Carlo approximations







See my NeurIPS 2020 tutorial on approximate inference

Today’s agenda

- Lecture on Basics: MFVI for BNNs
- Hands-on tutorial on BNNs
  - i.e., programming exercises
  - Also some case studies

Part I: Basics

- **Variational inference**
- **Bayes-by-backprop**

Bayesian Inference

$$P(\theta \mid D) = \frac{P(\theta)P(D \mid \theta)}{P(D)}$$

- $P(\theta)$: prior
- $P(D \mid \theta)$: likelihood
- $P(\theta| D)$: posterior
- $P(D)$: marginal

Variational Inference (VI)

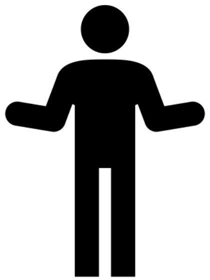
The posterior

$$p(\theta|D) = p(D\ |\theta)p(\theta)/p(D)$$

The variational distribution

$$q_{\phi}(\theta)$$

Inference as Optimization

$p(\theta|D)$ $\rightarrow$  $\leftarrow$ $q_{\phi}(\theta)$

Kullback-Leibler (KL) divergence

Kullback-Leibler Divergence

$$KL[q(\theta)||p(\theta)] \ = \int q(\theta) \log \frac{q(\theta)}{p(\theta)} \ d\theta = E_{q(\theta)}[\log \frac{q(\theta)}{p(\theta)}]$$

- When $p = q$, KL is 0
- Otherwise, KL > 0
- It measures how similar are these two distributions

Let’s Derive the Objective of VI

- Minimize $KL[q(\theta)||p(\theta|D)]$

$$KL[q(\theta)||p(\theta|D)] = -\mathrm{E}_{\mathrm{q}(\theta)}\left[\log\frac{p(\theta|D)}{q(\theta)}\right]$$

Let's Derive the Objective of VI

- Minimize $KL[q(\theta)||p(\theta|D)]$

$$KL[q(\theta)||p(\theta|D)] = -E_{q(\theta)}\left[\log\frac{p(\theta|D)}{q(\theta)}\right]$$

$$= -E_{q(\theta)}\left[\log\frac{\mathrm{p}(\theta,D)}{p(D)q(\theta)}\right] = -E_{q(\theta)}\left[\log\frac{\mathrm{p}(\theta,D)}{q(\theta)} - \log p(D)\right]$$

Let's Derive the Objective of VI

- Minimize $KL[q(\theta)||p(\theta|D)]$

$$KL[q(\theta)||p(\theta|D)] = -E_{q(\theta)}\left[\log\frac{p(\theta|D)}{q(\theta)}\right]$$

$$= -E_{q(\theta)}\left[\log\frac{\mathrm{p}(\theta,D)}{p(D)q(\theta)}\right] = -E_{q(\theta)}\left[\log\frac{\mathrm{p}(\theta,D)}{q(\theta)} - \log p(D)\right]$$

$$= \boxed{\log p(D)} - E_{q(\theta)}\left[\log\frac{p(\theta,D)}{q(\theta)}\right]$$

Model Evidence

Let’s Derive the Objective of VI

- Minimize $KL[q(\theta)||p(\theta|D)]$

$$KL[q(\theta)||p(\theta|D)] = \log p(D) - E_{q(\theta)}\left[\log \frac{p(\theta, D)}{q(\theta)}\right]$$

Maximize $E_{q(\theta)}\left[\log \frac{p(\theta,D)}{q(\theta)}\right]$

Let’s Derive the Objective of VI

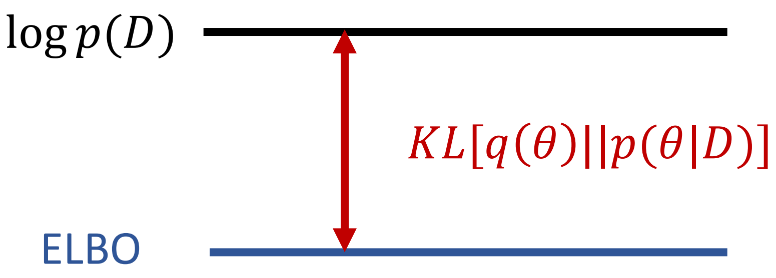
Minimize $KL[q(\theta)||p(\theta|D)]$

$$KL[q(\theta)||p(\theta|D)] = \boxed{\log p(D)} - E_{q(\theta)}\left[\log \frac{p(\theta, D)}{q(\theta)}\right]$$

Model Evidence

Maximize $L = E_{q(\theta)}\left[\log \frac{p(\theta,D)}{q(\theta)}\right]$

Evidence Lower Bound (ELBO)



“Model Evidence = ELBO + KL”

Variational Inference (VI)

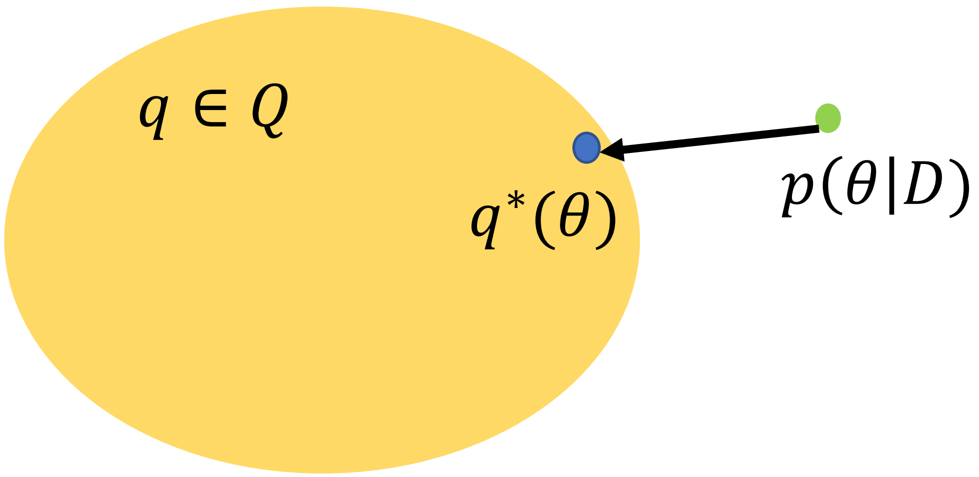
The posterior

$$p(\theta|D) = p(D|\theta)p(\theta)/p(D)$$

The variational distribution

$$q_\phi(\theta)$$

$$L = E_{q_{\phi(\theta)}}\left[\log \frac{\mathrm{p(D},\theta)}{q_\phi(\theta)}\right] = \log p(D) - KL[\, q_\phi(\theta)||p(\theta)]$$



Variational Inference (VI)

- Rewriting the ELBO:

$$\log p(D) \geq L = \underbrace{E_{q_{\phi(\theta)}}[\log p(D|\theta)]}_{\text{Data fitting term}} - \underbrace{KL[q_{\phi}(\theta)||p(\theta)]}_{\text{KL regulariser}}$$

(Negative) Data fitting term:

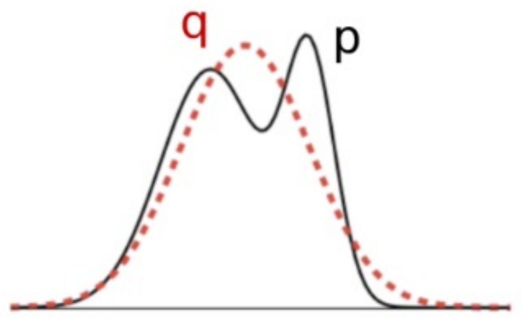
- Like the usual DL loss you'll use for training neural networks
- …except that now the network's weights are sampled from q

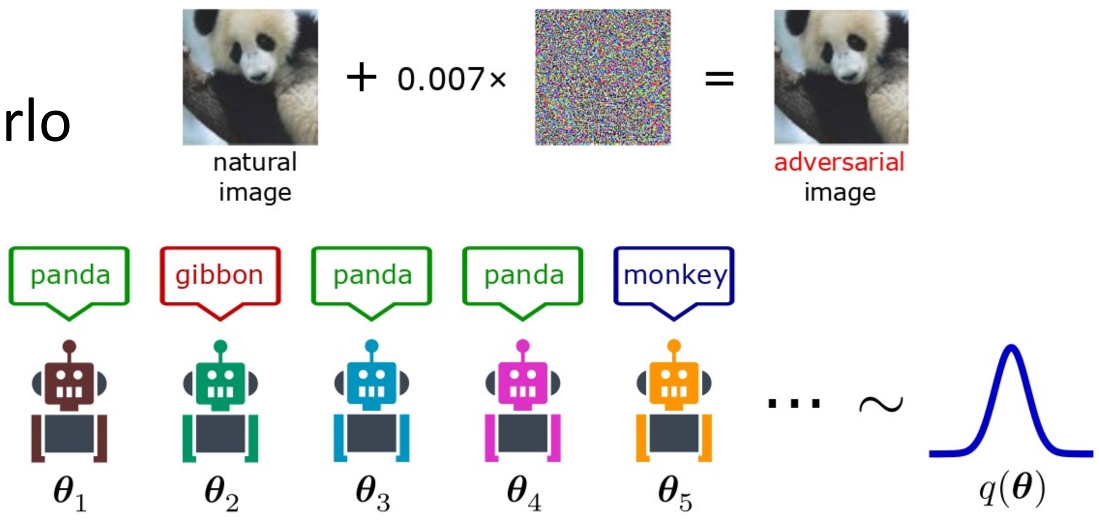
KL regulariser:

- Make the q distribution closer to the prior
- Regularises the approximate posterior, especially when using e.g., Gaussian prior

Approximate Inference in BNNs

- Key steps of approximate inference in BNNs
  1. Construct the $q(\theta) \approx p(\theta \mid D)$ distribution
     - Simple distributions: e.g. Mean-field Gaussian
     - Structured approximations, e.g. low-rank Gaussians
     - Others (non-Gaussian)
  2. Fit the $q(\theta)$ distribution
     - E.g. with variational inference
  3. Compute prediction with Monte Carlo approximations





See my NeurIPS 2020 tutorial on approximate inference

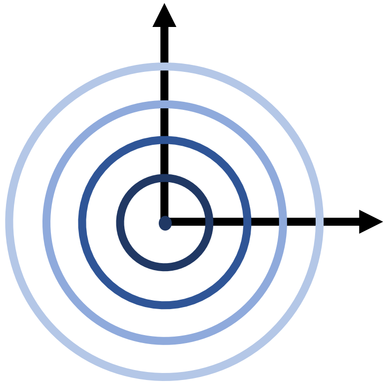
Approximate Inference in BNNs

- Step 1: construct the $q(\theta) \approx p(\theta \mid D)$ distribution
  - Example: Mean-field Gaussian distribution:

$$q(\theta) = \prod_{l=1}^{L} q\left(W^{l}\right) q(b^{l})$$

$$q(W_l) = \prod_{ij} q(W_{ij}^{l}), \qquad q\left(W_{ij}^{l}\right) = N(W_{ij}^{l}; M_{ij}^{l}, V_{ij}^{l})$$

$$q\left(b^{l}\right) = \prod_i q\left(b_i^{l}\right), \;\; q\left(b_i^{l}\right) = N(b_i^{l}; m_i^{l}, v_i^{l})$$



  - Variational parameters: $\phi = \left\{M_{ij}^{l}, \log V_{ij}^{l}, m_i^{l}, \log v_i^{l}\right\}_{l=1}^{L}$

Blundell et al. Weight Uncertainty in Neural Networks. ICML 2015

Approximate Inference in BNNs

- Step 2: fit the $q(\theta)$ distribution:
  - Variational inference: $\phi^* = argmax\ L(\phi)$

$$L(\phi) = E_{q_\phi(\theta)}[\log p(D \mid \theta)] - KL\left[q_\phi(\theta) \parallel p(\theta)\right]$$

Blundell et al. Weight Uncertainty in Neural Networks. ICML 2015

Approximate Inference in BNNs

- Step 2: fit the $q(\theta)$ distribution:
  - Variational inference: $\phi^* = argmax\ L(\phi)$

$$L(\phi) = E_{q_\phi(\theta)}[\log p(D \mid \theta)] - KL\left[q_\phi(\theta) \,\|\, p(\theta)\right]$$

  - First scalable technique: Stochastic optimization
    - i.i.d. assumption of data: $\log p(D \mid \theta) = \sum_{n=1}^{N} \log p(y_n \mid x_n, \theta)$
    - Enable mini-batch training with $\{(x_m, y_m)\} \sim D^M$:

$$L(\phi) \approx \frac{N}{M} \sum_{m=1}^{M} E_{q(\theta)}[\log p(y_m \mid x_m, \theta)] - KL[q(\theta) \,||\, p(\theta)]$$

Blundell et al. Weight Uncertainty in Neural Networks. ICML 2015

Approximate Inference in BNNs

- Step 2: fit the $q(\theta)$ distribution:
  - Variational inference: $\phi^* = argmax\, L(\phi)$

$$L(\phi) = E_{q_\phi(\theta)}[\log p(D \mid \theta)] - KL\big[q_\phi(\theta) \,\|\, p(\theta)\big]$$

  - First scalable technique: Stochastic optimization
    - i.i.d. assumption of data: $\log p(D \mid \theta) = \sum_{n=1}^{N} \log p(y_n \mid x_n, \theta)$
    - Enable mini-batch training with $\{(x_m, y_m)\} \sim D^M$:

$$L(\phi) \approx \boxed{\frac{N}{M}} \sum_{m=1}^{M} E_{q(\theta)}[\log p(y_m \mid x_m, \theta)] - KL[q(\theta) \,||\, p(\theta)]$$

reweighting to ensure calibrated
posterior concentration

Blundell et al. Weight Uncertainty in Neural Networks. ICML 2015

Approximate Inference in BNNs

- Step 2: fit the $q(\theta)$ distribution:
  - 2nd scalable technique: Monte Carlo sampling
    - $E_{q(\theta)}[\log p(y \mid x,\ \theta)]$ intractable even with Gaussian $q(\theta)$
    - Solution: Monte Carlo estimate:

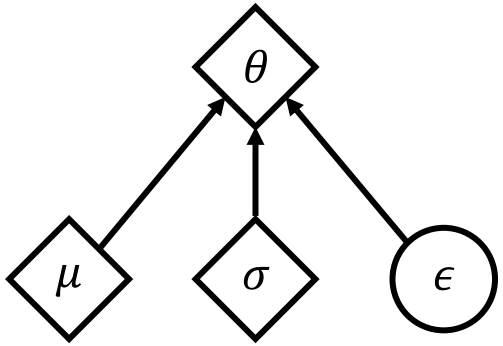
$$E_{q(\theta)}[\log p(y \mid x, \theta)] \approx \frac{1}{K} \sum_{k}^{K} \log p(y \mid x, \theta_k), \qquad \theta_k \sim q(\theta)$$

Blundell et al. Weight Uncertainty in Neural Networks. ICML 2015

Approximate Inference in BNNs

- Step 2: fit the $q(\theta)$ distribution:
  - 2nd scalable technique: Monte Carlo sampling
    - $E_{q(\theta)}[\log p(y \mid x,\ \theta)]$ intractable even with Gaussian $q(\theta)$
    - Solution: Monte Carlo estimate:

$$E_{q(\theta)}[\log p(y \mid x, \theta)] \approx \frac{1}{K} \sum_{k}^{K} \log p(y \mid x, \theta_k)\,, \qquad \theta_k \sim q(\theta)$$



- Reparameterization trick to sample mean-field Gaussians:

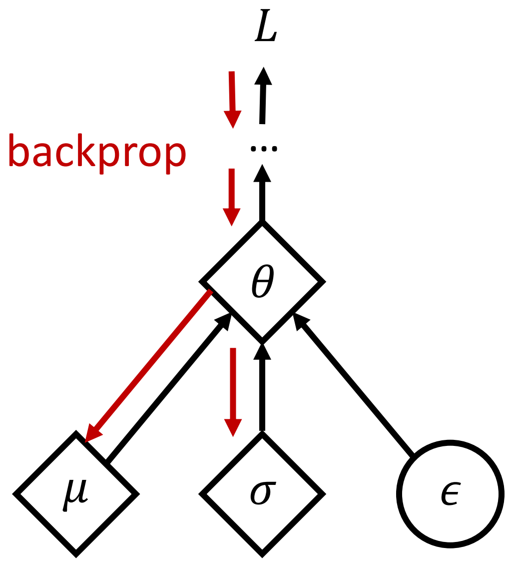
$$\theta_k \sim q(\theta) \Leftrightarrow \theta_k = m_\theta + \sigma_\theta \odot \epsilon_k,\ \ \epsilon_k \sim N(0, I)$$

Approximate Inference in BNNs

- Step 2: fit the $q(\theta)$ distribution:
  - 2nd scalable technique: Monte Carlo sampling
    - $E_{q(\theta)}[\log p(y \mid x, \theta)]$ intractable even with Gaussian $q(\theta)$
    - Solution: Monte Carlo estimate:

$$E_{q(\theta)}[\log p(y \mid x, \theta)] \approx \frac{1}{K} \sum_{k}^{K} \log p(y \mid x, \theta_k), \qquad \theta_k \sim q(\theta)$$

    - Reparameterization trick to sample mean-field Gaussians:
      $\theta_k \sim q(\theta) \Leftrightarrow \theta_k = m_\theta + \sigma_\theta \odot \epsilon_k,\ \epsilon_k \sim N(0, I)$



$$\Rightarrow\ E_{q(\theta)}[\log p(y \mid x, \theta)] \approx \frac{1}{K} \sum_{k}^{K} \log p(y \mid x, \theta_k = m_\theta + \sigma_\theta \epsilon_k), \epsilon_k \sim N(0, I)$$

Blundell et al. Weight Uncertainty in Neural Networks. ICML 2015

Approximate Inference in BNNs

- Combining both steps:

$$L(\phi) \approx \frac{N}{M} \sum_{m=1}^{M} \frac{1}{K} \sum_{k=1}^{K} \log p(y_m \mid x_m, \theta_k) - \underline{KL[q(\theta) \,||\, p(\theta)]}, \theta_k \sim q(\theta)$$

analytic between two Gaussians
(if not, can also be estimated with Monte Carlo)

In regression:
$p(y \mid x, \theta) = N(f_\theta(x), \sigma^2)$

In classification:
$p(y \mid x, \theta) = Categorical(logit = f_\theta(x))$

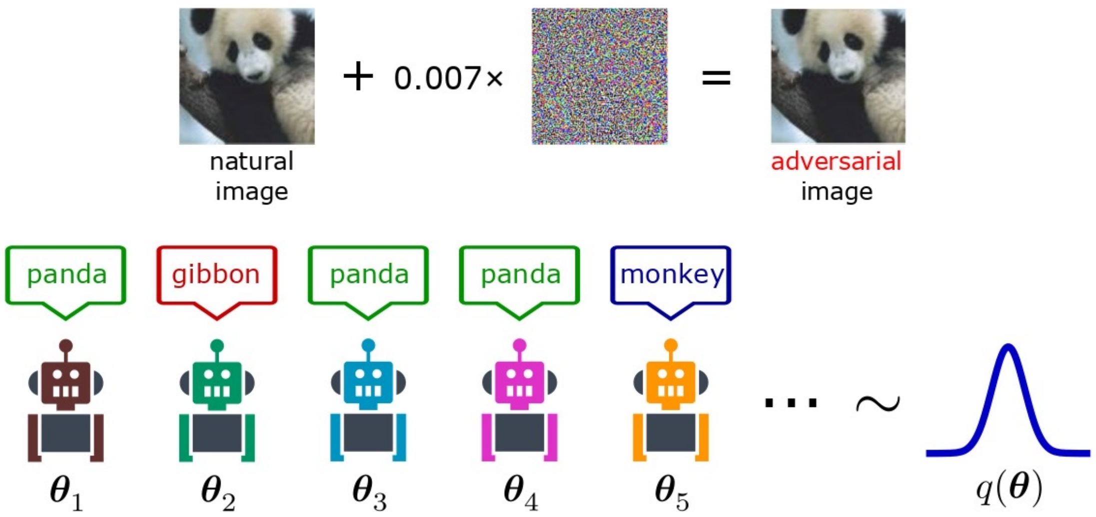
Blundell et al. Weight Uncertainty in Neural Networks. ICML 2015

Approximate Inference in BNNs

- Step 3: compute prediction with Monte Carlo approximations:

$$p(y^* \mid x^*, D) \approx \frac{1}{K} \sum_{k=1}^{K} p(y^* \mid x^*, \ \theta_k), \quad \underline{\theta_k \sim q(\theta)}$$

Mean-field Gaussian case:
$\theta_k = m_\theta + \sigma_\theta \odot \epsilon_k, \ \epsilon_k \sim N(0, I)$



Blundell et al. Weight Uncertainty in Neural Networks. ICML 2015

Part II: Bayesian MLPs

- **Implement various BNN methods for MLP architectures**
- **Regression example test**
- **Case study 1: Bayesian Optimisation with UCB**

https://bit.ly/probai2023_bnn_regression

Instructions for using this Google Colab notebook:

- Make sure you have signed in with your Google account;
- Click “File > Save a copy in Drive” to create your own copy;
- Let’s play around with the demo using your own copy!

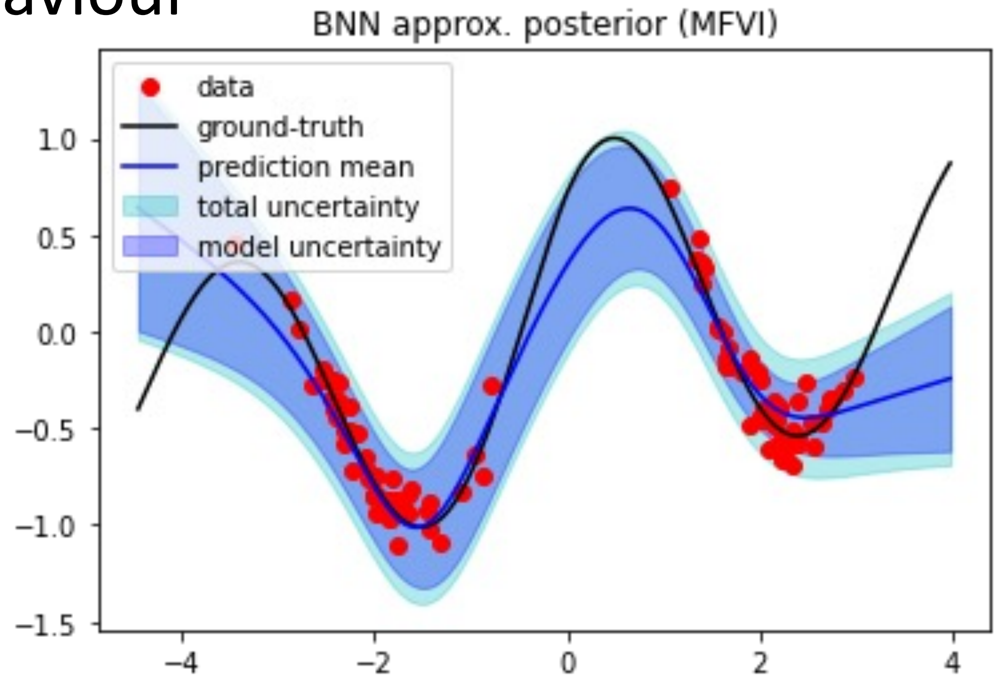
Findings with MFVI for Bayesian MLPs

Findings with MFVI for Bayesian MLPs

- MFVI tends to underfit
  - Initialisation matters
  - Tuning the beta parameter also helps

Findings with MFVI for Bayesian MLPs

- MFVI tends to underfit
  - Initialisation matters
  - Tuning the beta parameter also helps
- Uncertainty behaviour



Using other q distributions?

- Using more complicated q distributions?
  - Pros: more flexible approximations ⇒ better posterior approximations (?)
  - Cons: higher time & space complexities

Using other q distributions?

- Using more complicated q distributions?
  - Pros: more flexible approximations $\Rightarrow$ better posterior approximations (?)
  - Cons: higher time & space complexities

- We will look at 2 alternatives:
  - "Last-layer BNN": Full covariance Gaussian approximations for the last layer
  - MC-Dropout: adding dropout layers and run them in both train & test time

“Last-layer BNN”

- Use deterministic layers for all but the last layer
- For the last layer: Use Full-covariance Gaussian approximate posterior:

$$q(\theta^l) = \delta(W^l = M^l, b^l = m^l), l = 1, \dots, L-1,$$

$$q(\theta^L) = N(vec(\theta^L); vec(\mu^L), \Sigma), \theta^L = \{W^L, b^L\}$$

- For regression this is equivalent to Bayesian linear regression (BLR) with NN-based non-linear features

$x \longrightarrow$

$\longrightarrow f_{\theta^{1:L-1}}(x) \longrightarrow$ BLR $\longrightarrow p(y \mid x, \theta)$

"Last-layer BNN"

- Use deterministic layers for all but the last layer
- For the last layer: Use Full-covariance Gaussian approximate posterior
- For regression this is equivalent to Bayesian linear regression (BLR) with NN-based non-linear features

x →

→ $f_{\theta^{1:L-1}}(x)$ → BLR → $p(y \mid x, \theta)$

$$L = E_q[\log p(D \mid \theta)] - KL[q(\theta^L) \parallel p(\theta^L)]$$

Use deterministic weights for $l = 1, \ldots, L-1$
Sample $W^L \sim q(W^L)$

KL regulariser for the last layer only

MC-Dropout

- Add dropout layers to the network
- Perform dropout during training

L2 regulariser on the variational parameters

$$L = E_q[\log p(D|\theta)] - (1 - \pi)\ell_2(\phi)$$

The MC sampling procedure is implicitly defined

Dropout rate

- In test time, run multiple forward passes with dropout

$$p(y^* \mid x^*, D) \approx \frac{1}{K} \sum_{k=1}^{K} p(y^* \mid x^*,\ \theta_k), \quad \theta_k \sim q(\theta)$$

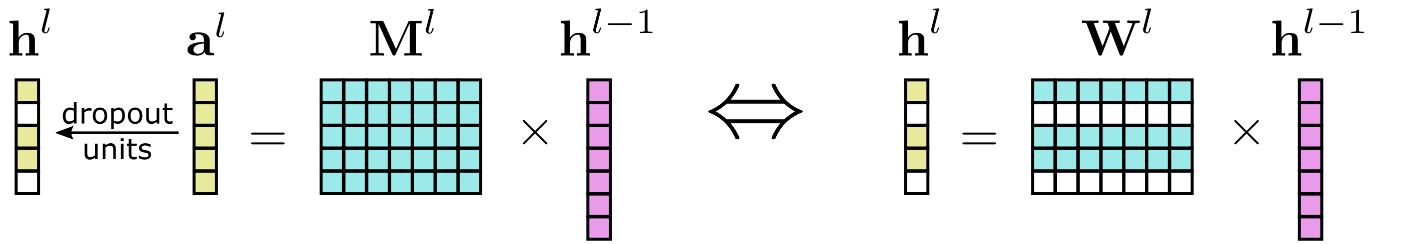
The MC sampling procedure is implicitly defined

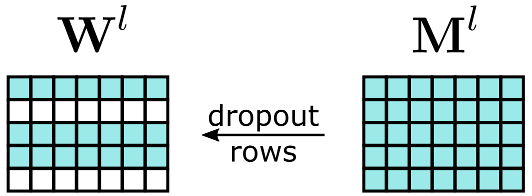
Gal and Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. ICML 2016.

MC-Dropout

- Two equivalent ways to implement MC-Dropout:

Activation Dropout with rate π $\qquad$ Dropout rows with rate π





$\Leftrightarrow$ Sample rows $\mathbf{W}_i^l$ from

$$q(\mathbf{W}_i^l) = (1-\pi)\mathcal{N}(\mathbf{M}_i^l, \eta\mathbf{I}) + \pi\mathcal{N}(\mathbf{0}, \eta\mathbf{I})$$

$$\eta \to 0$$

(Similar logic applies when including the bias terms, see lecture notes.)
(Notice that pytorch's nn.Linear layer uses formats like xW^T instead of Wx.)

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

Using other q distributions?

- What you'll do for the next part of the tutorial:
  - Implement MC-Dropout in 2 ways
  - Run the regression sample with the 2 approximation methods discussed
  - Compare with MFVI

Case study 1: Bayesian Optimisation

- Imagine you'd like to solve the following task:

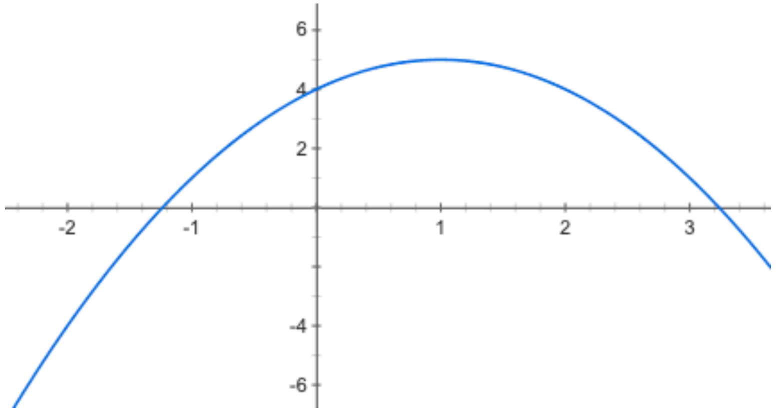
$$x^* = argmax_x \, f_0(x)$$

Case study 1: Bayesian Optimisation

- Imagine you'd like to solve the following task:

$$x^* = argmax_x \, f_0(x)$$

Known functional form of f_0:



Gradient descent, Newton's method,

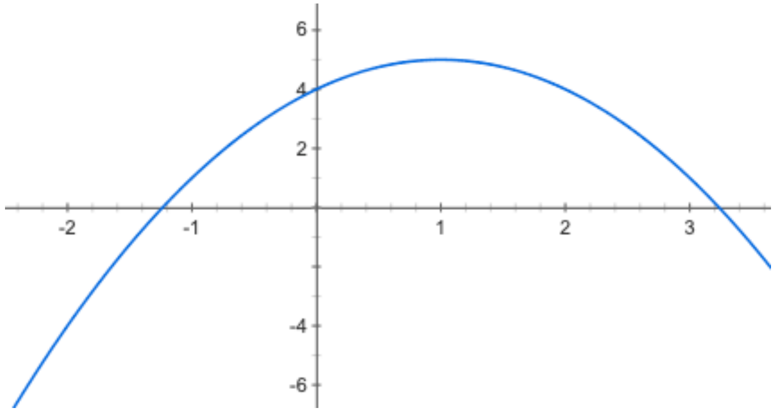
…

Case study 1: Bayesian Optimisation

- Imagine you'd like to solve the following task:

$$x^* = argmax_x \; f_0(x)$$

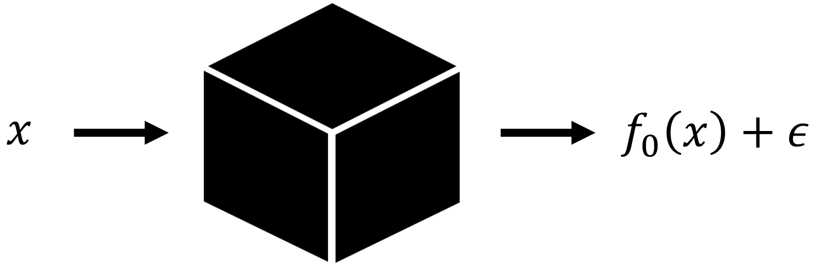
Known functional form of f_0:



Gradient descent, Newton's method,

…

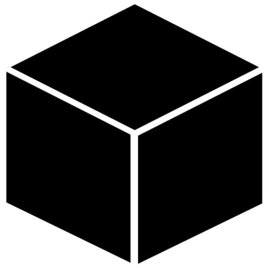
Unknown functional form of f_0:



(can only query (noisy) function values)

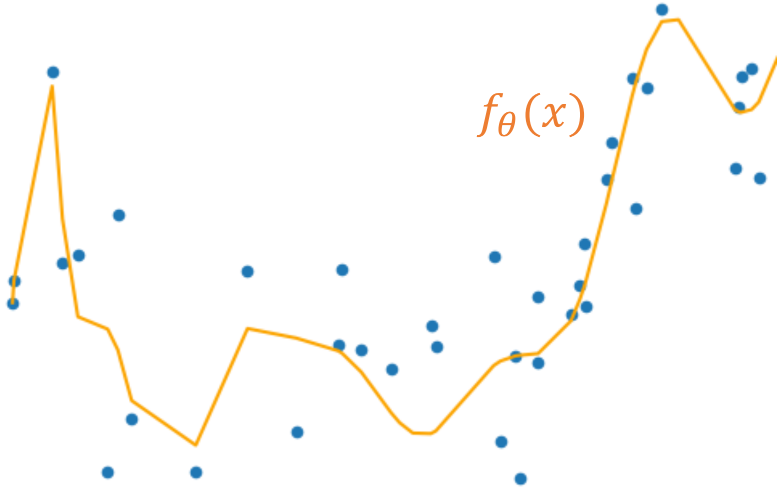
Case study 1: Bayesian Optimisation

- Idea 1: fit a surrogate function $f_\theta \approx f_0$

$x_i \longrightarrow$  $\longrightarrow y_i = f_0(x_i) + \epsilon_i$

Collect a dataset $D = \{(x_i, y_i)\}_{i=1}^N$

Fit f_θ using D $\longrightarrow$



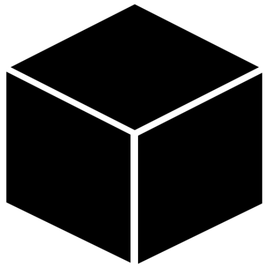
$f_\theta(x)$ has a known (parametric) form

$\Rightarrow$ find maximum using e.g., Newton's method

Case study 1: Bayesian Optimisation

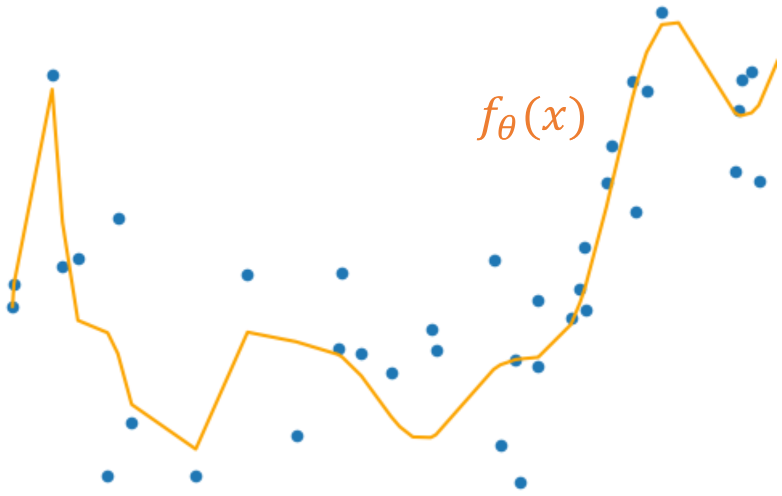
- Idea 1: fit a surrogate function $f_\theta \approx f_0$

$x_i \rightarrow$



$\rightarrow y_i = f_0(x_i) + \epsilon_i$

Fit f_θ using D



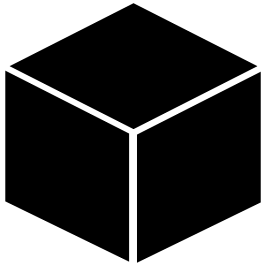
Collect a dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$

- Issues of this approach:
  - Need to collect a lot of datapoints for accurate fitting of f_θ
  - Do not consider uncertainty at unseen locations

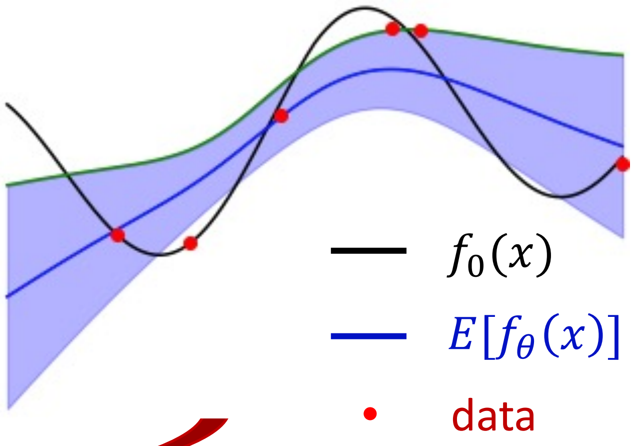
Case study 1: Bayesian Optimisation

- Idea of BO: iterate the following steps
  - fit a surrogate function f_θ with uncertainty estimates
  - Use the surrogate function to guide the dataset collection process

Update the dataset $D = D \cup \{(x_*, y_*)\}$

$x_* \longrightarrow$  $\longrightarrow y_* = f_0(x_*) + \epsilon_*$

Fit f_θ using D $\longrightarrow$ with uncertainty estimates



Propose next point to query: $x_* = argmax_x\ a(x)$ $\longleftarrow$ acquisition function

Srinivas et al. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. ICML 2010.
Snoek et al. Practical Bayesian Optimization of Machine Learning Algorithms. NeurIPS 2012.

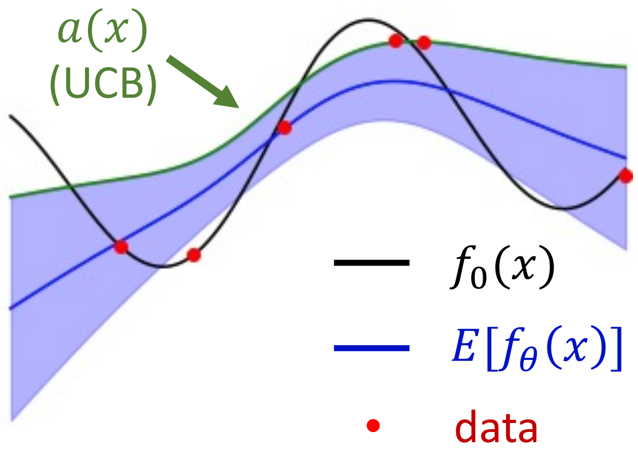
Case study 1: Bayesian Optimisation

- Upper confidence bound (UCB): a widely used acquisition function

$$a(x) = m(x) + \beta\sigma(x)$$

Mean of $f_\theta(x)$ over $\theta \sim q(\theta)$

Std of $f_\theta(x)$ over $\theta \sim q(\theta)$



Srinivas et al. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. ICML 2010.
Snoek et al. Practical Bayesian Optimization of Machine Learning Algorithms. NeurIPS 2012.

Case study 1: Bayesian Optimisation

- What you'll do for the case study part of the tutorial:
  - Implement UCB acquisition function
  - Run the BO example
  - Play around with hyper-parameters and other settings

Part III: Bayesian ConvNets

- **Classification example test**
- **Case study 2: Detecting adversarial examples**

https://bit.ly/probai2023_bnn_classification

Instructions for using this Google Colab notebook:

- Make sure you have signed in with your Google account;
- Click “File > Save a copy in Drive” to create your own copy;
- Use GPU: in “Runtime > Change runtime type”, choose “GPU” for “hardware accelerator”
- Let’s play around with the demo using your own copy!

Case study 2: Detecting adversarial examples

- Hypothesis:
  - Adversarial examples are regarded as OOD data
  - BNNs become uncertain about their prediction on OOD data
  - $\Rightarrow$ uncertainty measures can be used for detecting adversarial examples

Uncertainty measures

Total uncertainty = epistemic uncertainty + aleatoric uncertainty

Due to lack of “knowledge”
(reducible when having more data)

Due to inherent stochasticity in data
(non-reducible)

Imagine flipping a coin:

- Epistemic uncertainty: “How much do I believe the coin is fair?”
  - Model’s belief after seeing the population
  - Reduces when having more data
- Aleatoric uncertainty: “What’s the next coin flip outcome?”
  - Individual experiment outcome
  - Non-reducible

Uncertainty measures

Total uncertainty = epistemic uncertainty + aleatoric uncertainty

Due to lack of "knowledge"
(reducible when having more data)

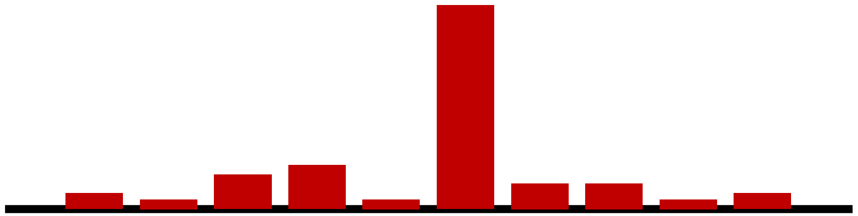
Due to inherent stochasticity in data
(non-reducible)

Computing uncertainty in classification models:

$$H[p] = -\sum_{c=1}^{C} p_c \log p_c, \qquad p = (p_1, \dots, p_C), \sum_{c=1}^{C} p_c = 1$$

High entropy: $H[p] \to \log C$

Low entropy: $H[p] \to 0$



Uncertainty measures

Total uncertainty = epistemic uncertainty + aleatoric uncertainty

Due to lack of "knowledge"
(reducible when having more data)

Due to inherent stochasticity in data
(non-reducible)

Computing uncertainty in classification models:
Recall for Bayesian predictive distribution:

$$p(y^* \mid x^*, D) = \int p(y^* | x^*, \theta) p(\theta | D) d\theta$$

$$H[y^* | \, x^*, D] = I[y^*; \theta \mid x^*, D] + E_{p(\theta \mid D)}[H[y^* \, | x^*, \theta]]$$

Total entropy of the predictive distribution

Mutual information between y^* and θ

Conditional entropy under posterior

Uncertainty measures

Total uncertainty = epistemic uncertainty + aleatoric uncertainty

Due to lack of "knowledge"
(reducible when having more data)

Due to inherent stochasticity in data
(non-reducible)

Computing uncertainty in classification models:

Recall for Bayesian predictive distribution with approximation:

$$p(y^* \mid x^*, D) \approx \frac{1}{K} \sum_{k=1}^{K} p(y^* \mid x^*, \theta_k), \quad \theta_k \sim q(\theta)$$

Total entropy (for total uncertainty):

$$H[y^* \mid x^*, D] \approx H[\frac{1}{K} \sum_{k=1}^{K} p(y^* \mid x^*, \theta_k)]$$

Uncertainty measures

Total uncertainty = epistemic uncertainty + aleatoric uncertainty

Due to lack of "knowledge"
(reducible when having more data)

Due to inherent stochasticity in data
(non-reducible)

Computing uncertainty in classification models:

Recall for Bayesian predictive distribution with approximation:

$$p(y^* \mid x^*, D) \approx \frac{1}{K} \sum_{k=1}^{K} p(y^* \mid x^*, \theta_k), \quad \theta_k \sim q(\theta)$$

Conditional entropy (for aleatoric uncertainty):

$$E_{p(\theta \mid D)}[H[y^* \mid x^*, \theta]] \approx \frac{1}{K} \sum_{k=1}^{K} H[p(y^* \mid x^*, \theta_k)]$$

Uncertainty measures

Total uncertainty = epistemic uncertainty + aleatoric uncertainty

Due to lack of "knowledge"
(reducible when having more data)

Due to inherent stochasticity in data
(non-reducible)

Computing uncertainty in classification models:
Recall for Bayesian predictive distribution with approximation:

$$p(y^* \mid x^*, D) \approx \frac{1}{K} \sum_{k=1}^{K} p(y^* \mid x^*, \theta_k), \quad \theta_k \sim q(\theta)$$

Mutual information (for epistemic uncertainty):

$$I[y^*; \theta \mid x^*, D] \approx H[\frac{1}{K} \sum_{k=1}^{K} p(y^* \mid x^*, \theta_k)] - \frac{1}{K} \sum_{k=1}^{K} H[p(y^* \mid x^*, \theta_k)]$$

Uncertainty measures

Total uncertainty = epistemic uncertainty + aleatoric uncertainty

Due to lack of "knowledge"
(reducible when having more data)

Due to inherent stochasticity in data
(non-reducible)

Computing uncertainty in classification models:
Recall for Bayesian predictive distribution with approximation:

$$p(y^* \mid x^*, D) \approx \frac{1}{K} \sum_{k=1}^{K} p(y^* \mid x^*, \theta_k), \quad \theta_k \sim q(\theta)$$

Mutual information (for epistemic uncertainty) if you can do exact inference:

$$I[y^*; \theta \mid x^*, D] = E_{p(y^*|x^*,D)}[KL[p(\theta|D, x^*, y^*) \parallel p(\theta|D)]]$$

"What the model thinks the posterior is going to change if we add new observation at location x^*"

Case study 2: Detecting adversarial examples

- What you'll do for the case study part of the tutorial:
  - Implement the uncertainty measures
    - Total entropy, conditional entropy, and mutual info
  - Run adversarial attacks on various trained networks
  - See how diversity helps in detecting adversarial examples
    - Detection by thresholding the uncertainty measures
    - We consider best TPR with FPR $\leq$ 5%

Ensemble BNNs

- Define q distribution as mixture of mean-field Gaussian:

$$q(\theta) = \frac{1}{S}\sum_{s=1}^{S} q(\theta|s), \quad q(\theta|s) = N(\theta; \mu_s, diag(\sigma_s^2))$$

- Objective is still a valid lower-bound to $\log p(D)$:

$$L = \frac{1}{S}\sum_{s=1}^{S} ELBO[q(\theta|s)]\,, ELBO[q(\theta|s)] = E_{q(\theta|s)}[\log p(D \mid \theta)] - KL[q(\theta|s) \,||\, p(\theta)]$$

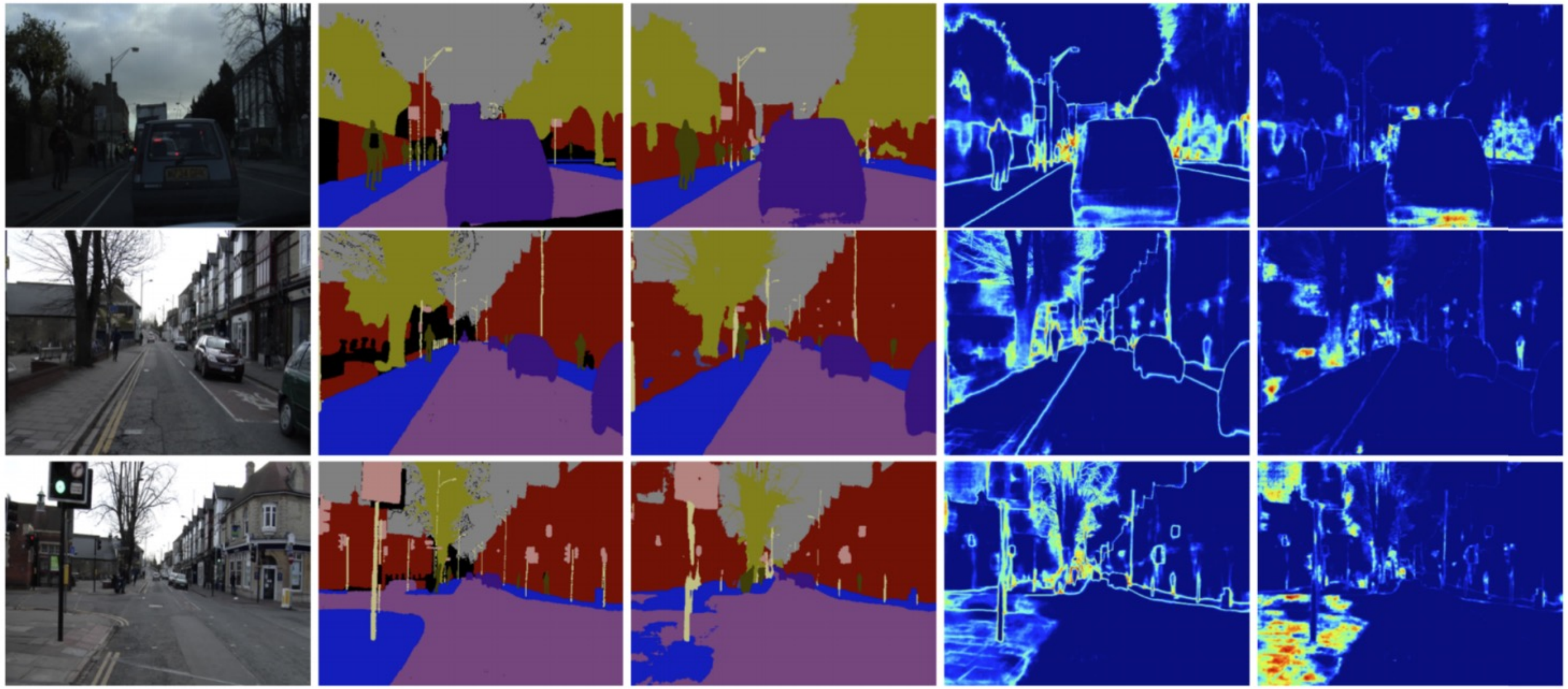
- The parameters of $q(\theta|s)$ for different s are independent
  $\Rightarrow$ train S number of MFVI-BNNs independently

Ensemble BNNs are winning solutions of NeurIPS 2021 Approximate Inference in BDL Competition

Part IV: Advances & Future Works

- **Various applications**
- **Glossary of BDL methods**
- **Future directions**

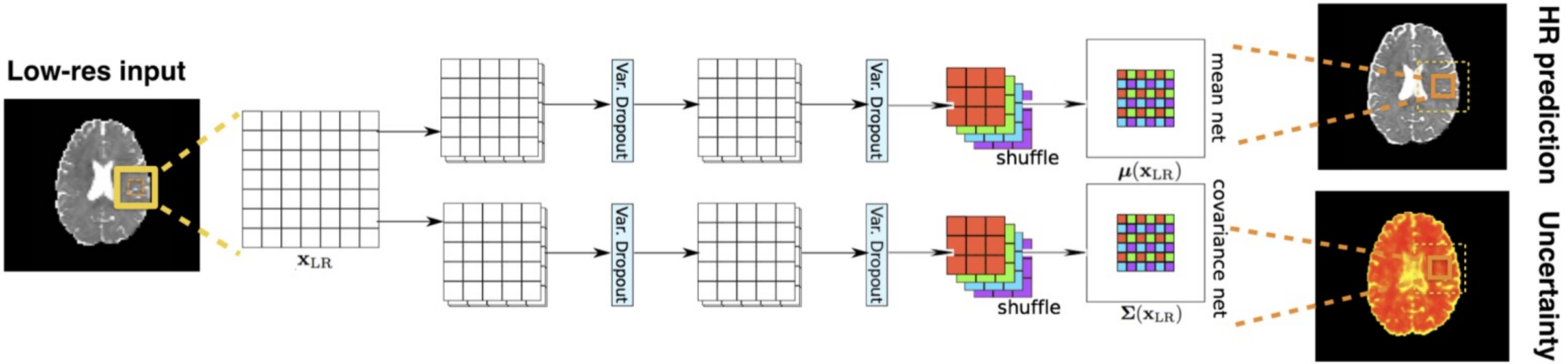
Applications of BNNs: Image Segmentation

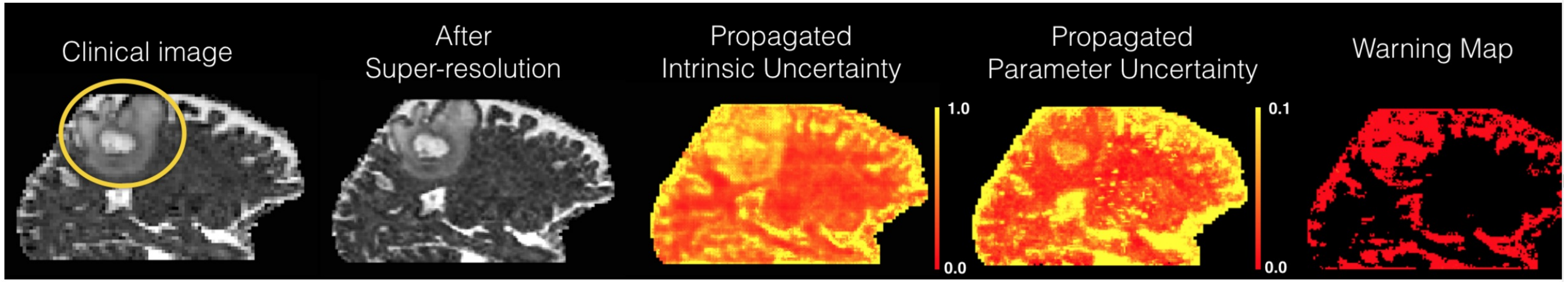


(a) Input Image (b) Ground Truth (c) Semantic Segmentation (d) Aleatoric Uncertainty (e) Epistemic Uncertainty

Kendall and Gal. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? NeurIPS 2017

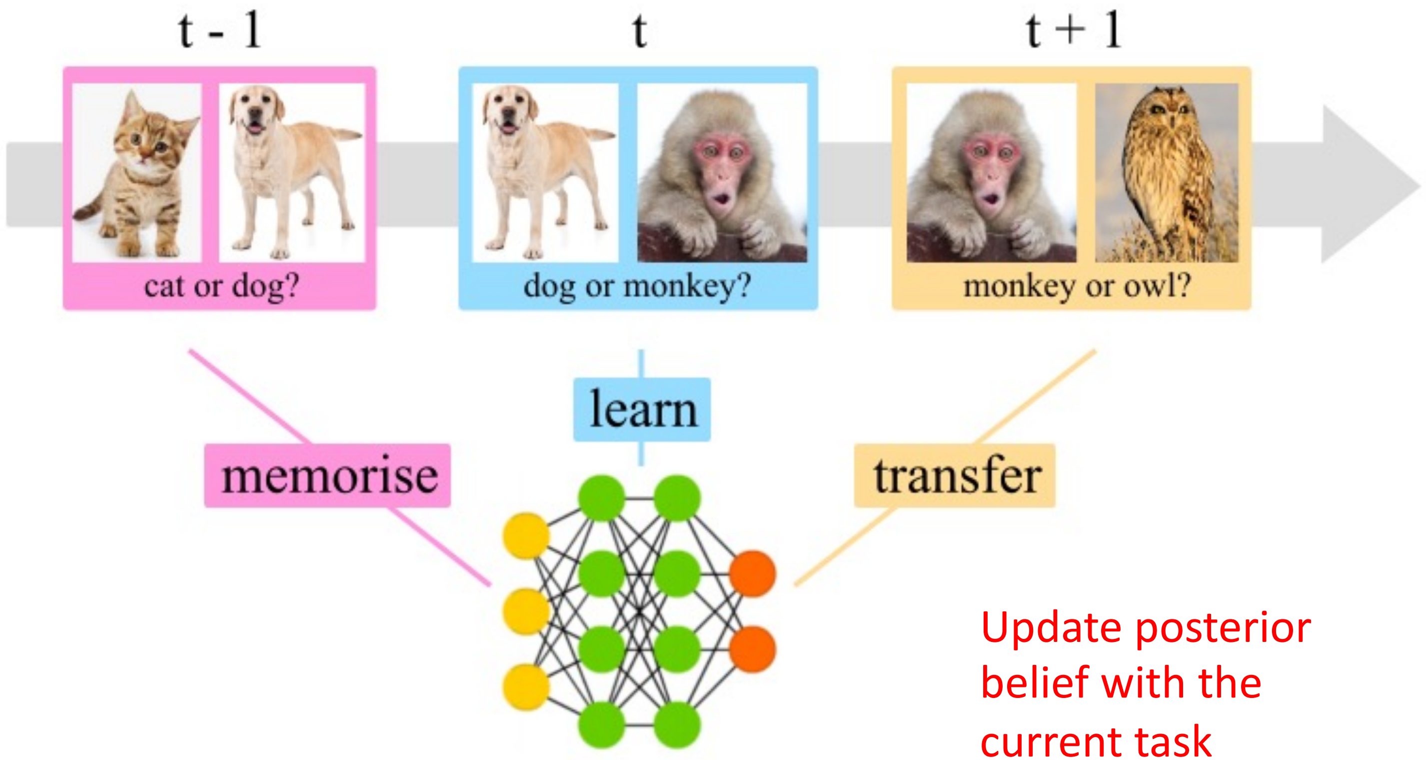
Applications of BNNs: Super Resolution





Tanno et al. Uncertainty Quantification in Deep Learning for Safer Neuroimage Enhancement. Neuroimage 2020

Applications of BNNs: Continual Learning



Update posterior belief with the current task

Posterior from the previous tasks as prior for the current task

$$L^t_{VCL}(q_t(\theta)) = E_{q_{t(\theta)}}[\log p(D_t \mid \theta)] - KL[\boxed{q_t(\theta)} \,||\, \boxed{q_{t-1}(\theta)}]$$

Nguyen et al. Variational Continual Learning. ICLR 2018

Pan et al. Continual Deep Learning by Functional Regularisation of Memorable Past. NeurIPS 2020

A Glossary of BDL methods

- Methods based on approximate inference
  - Variational inference with different q distribution design
  - Laplace method
  - Moment matching (EP, message passing)
- Methods based on sampling
  - SG-MCMC
  - Particle-based inference
- Function-space inference: $q(W) \to q(f)$
  - With helps from Gaussian Processes (GPs), Neural Tangent Kernel (NTK), deep kernel learning
- Ensemble methods
  - Deep ensemble, efficient ensemble methods

Wilson and Izmailov. Bayesian Deep Learning and a Probabilistic Perspective of Generalization. NeurIPS 2020

Applications of BNNs in Transformers?

Best method still unclear (under research), but a few observations:

- Transformers need big amount of data to train and get decent accuracy
  - So methods like simple MFVI (which tends to underfit) work less well

- Multi-head attention module: different probabilistic perspectives
  - Naïve application of BNN methods to weights (MC-dropout)
  - Probabilistic attention module: understanding attention matrix as random variable
  - Gaussian process perspective

Chen and **Li**. Calibrating Transformers via Sparse Gaussian Processes. ICLR 2023

Future directions

- Understanding BNN behaviour:
  - How would $q(f)$ behave given a particular form of $q(W)$?
  - Is weight-space objective appropriate for MFVI?
  - We don’t understand very well the optimisation properties of VI-BNN
- Scaling up BNNs in the era of “foundation models”:
  - How can we make the approximate posterior more efficient in both time and space complexities?
- Priors for BNNs
  - How to think about priors in function space?
  - Priors for Transformer-based networks?
- Applications
  - Improve for applications that require good uncertainty estimates

Ritter et al. Sparse Uncertainty Representation in Deep Learning with Inducing Weights. NeurIPS 2021
Fortuin. Priors in Bayesian deep learning: A review. International Statistical Review, 2022.

Thank You!

Questions? Ask NOW or email:
yingzhen.li@imperial.ac.uk

Example answers of the tutorial demos:
Regression: https://bit.ly/probai2023_bnn_regression_answer
Classification: https://bit.ly/probai2023_bnn_classification_answer