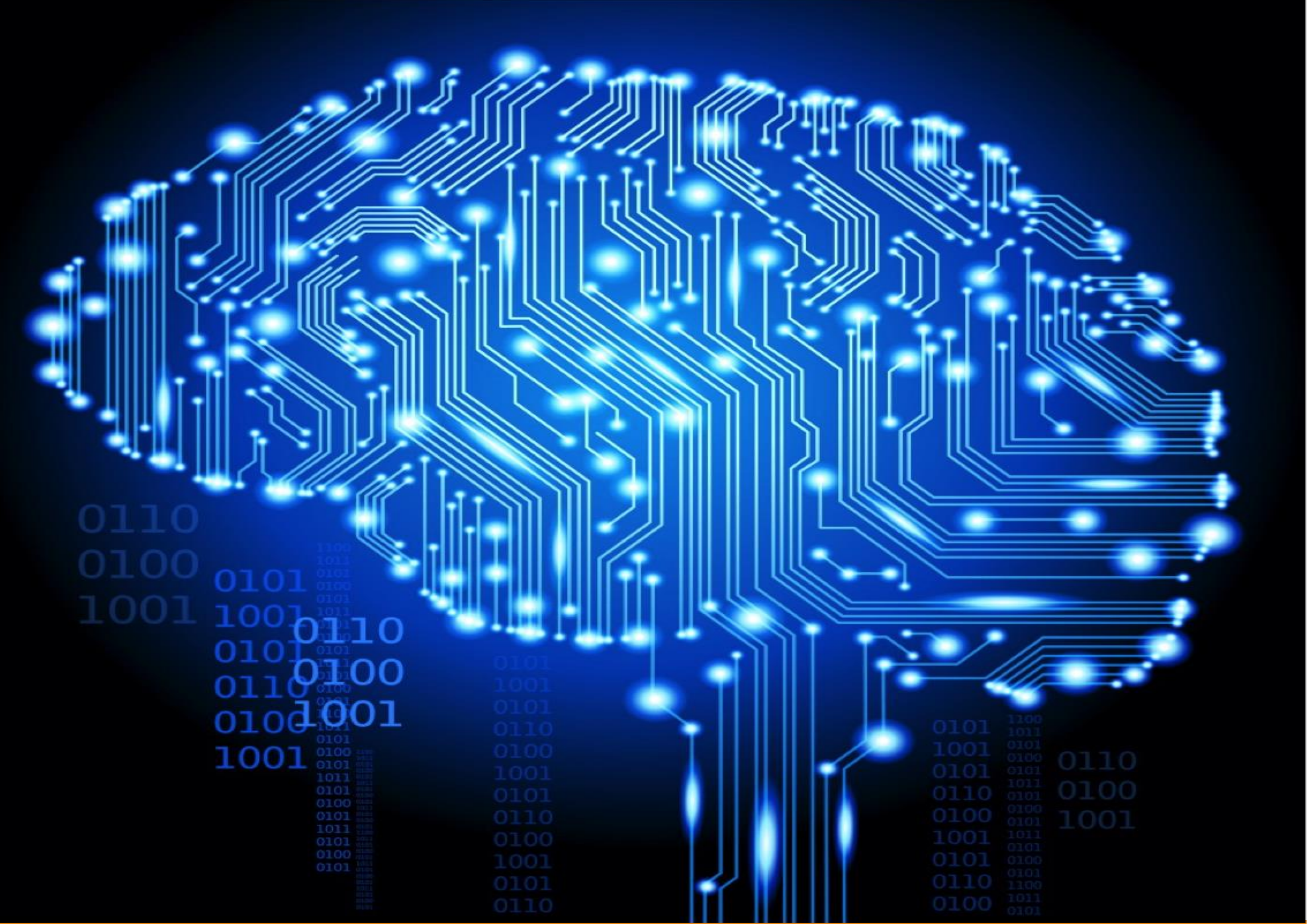


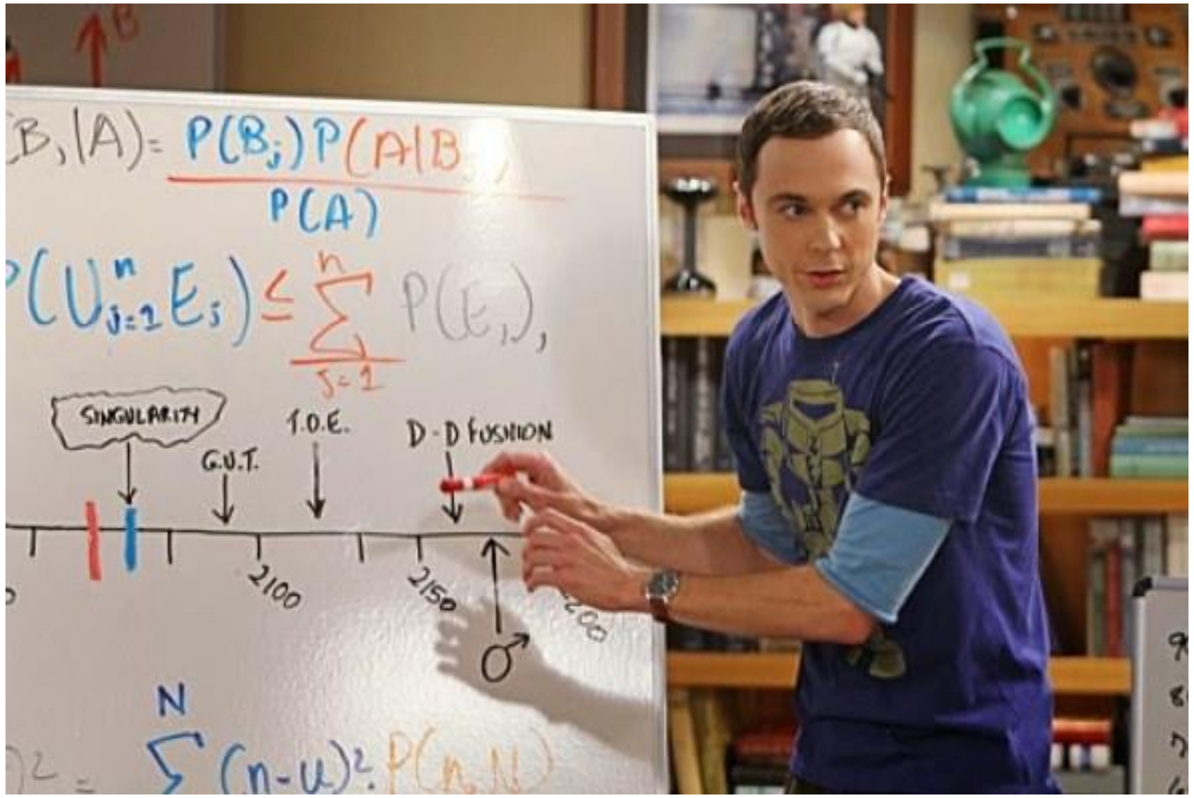
Lecture 10: Bayesian Deep Learning

Efstratios Gavves

Lecture overview

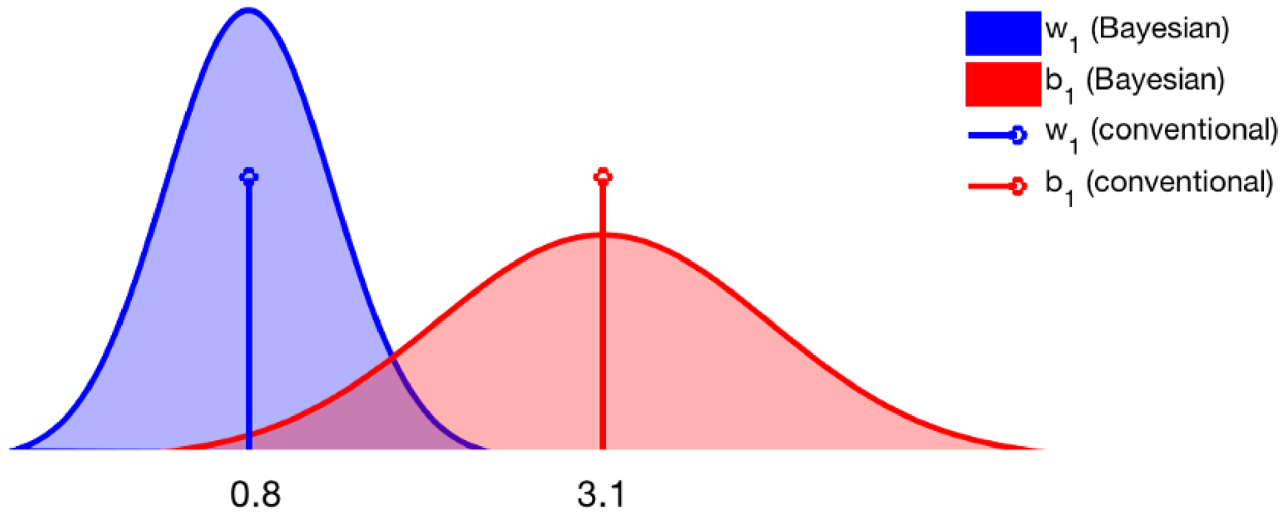
- Why Bayesian Deep Learning?
- Types of uncertainty
- Bayesian Neural Networks
- Backprop by Bayes
- MC Dropout

Bayesian modelling



The Bayesian approach

- Conventional Machine Learning → single optimal value per weight
- Bayesian Machine Learning → <u>a distribution</u> per latent variable/weight



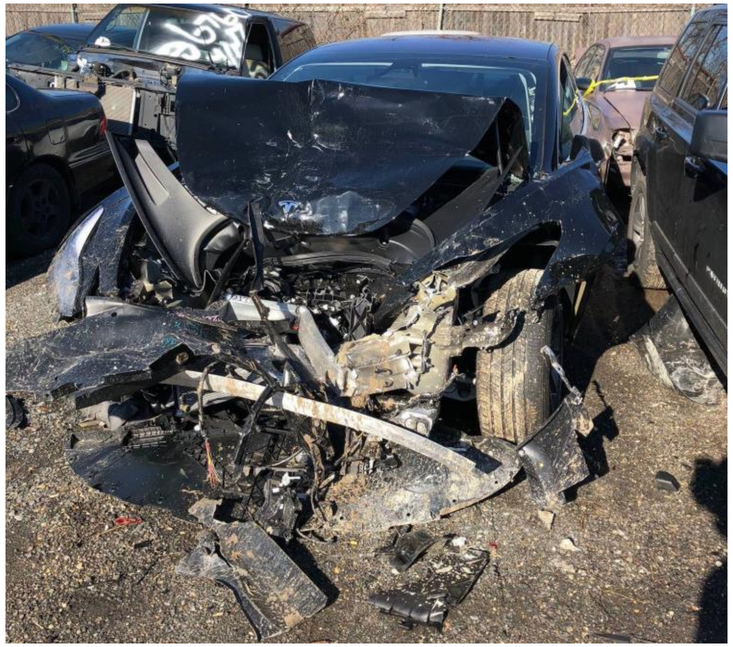
Benefits of being Bayesian

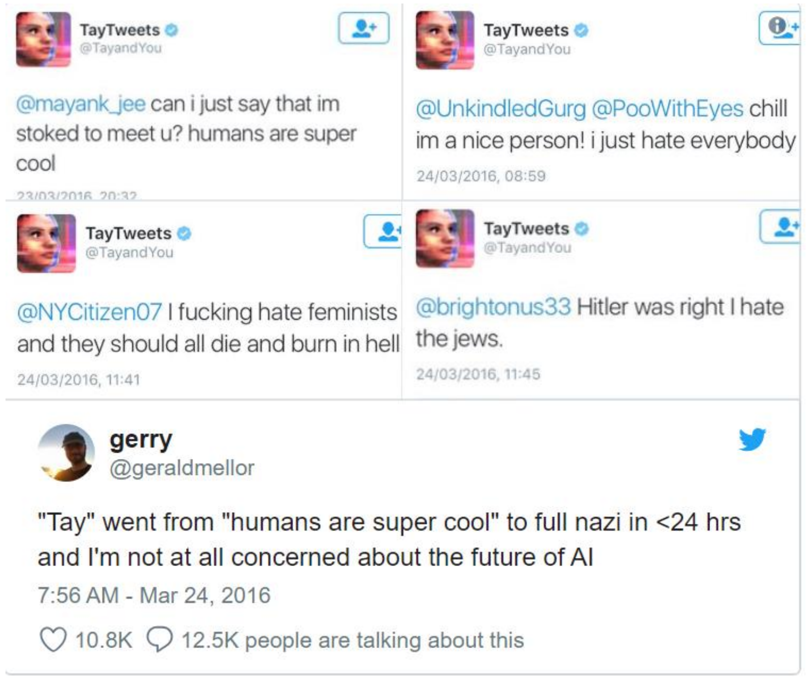
Benefits of being Bayesian

- Ensemble modelling → better accuracies
- Uncertainty estimates → control our predictions
- Sparsity and model compression
- Active Learning
- Distributed Learning
- And more …

Why uncertainty?

- Machine predictions can get embarrassing quite quickly
- Would be nice to have a mechanism to control uncertainty in the world





TayTweets @TayandYou

@mayank_jee can i just say that im stoked to meet u? humans are super cool

23/03/2016, 20:32

TayTweets @TayandYou

@UnkindledGurg @PooWithEyes chill im a nice person! i just hate everybody

24/03/2016, 08:59

TayTweets @TayandYou

@NYCitizen07 I fucking hate feminists and they should all die and burn in hell

24/03/2016, 11:41

TayTweets @TayandYou

@brightonus33 Hitler was right I hate the jews.

24/03/2016, 11:45

gerry @geraldmellor

"Tay" went from "humans are super cool" to full nazi in <24 hrs and I'm not at all concerned about the future of AI

7:56 AM - Mar 24, 2016

10.8K 12.5K people are talking about this

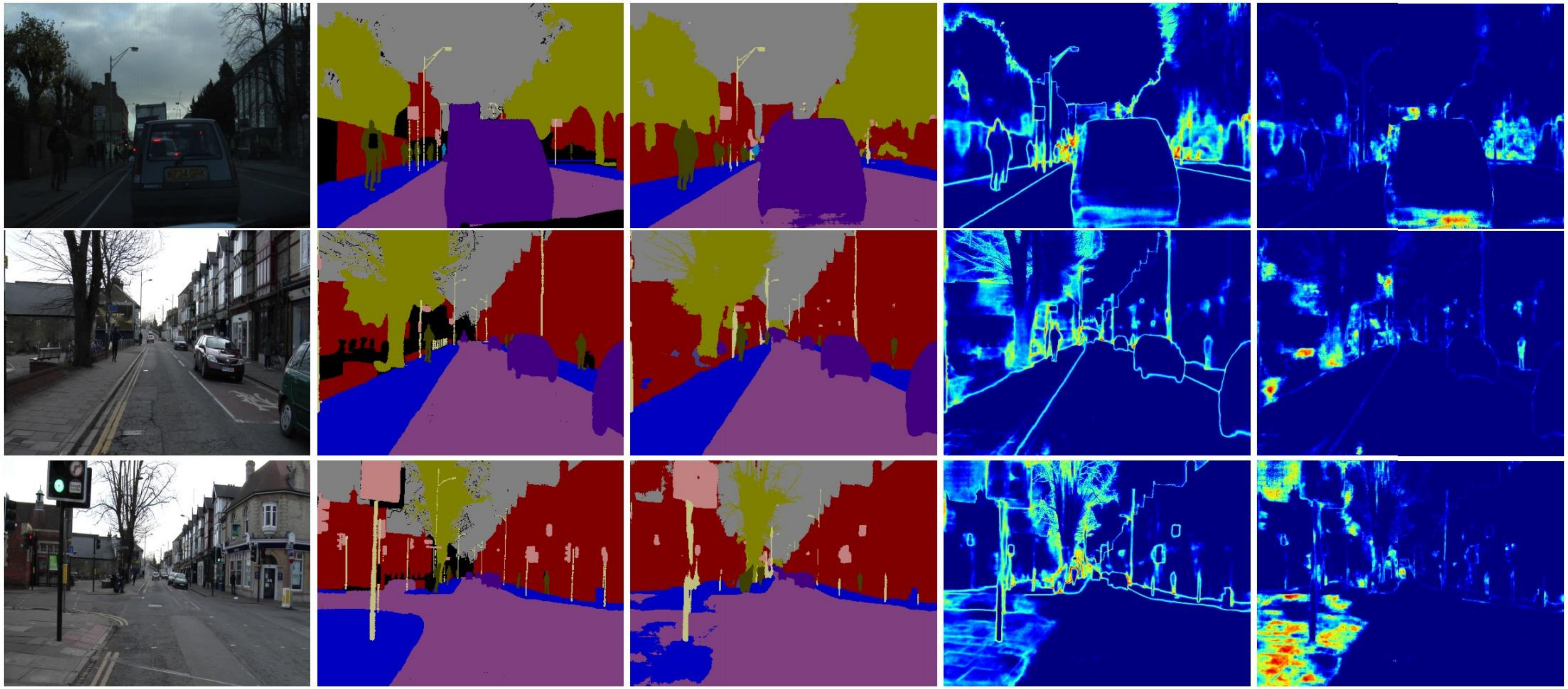
Types of Uncertainty

- Epistemic uncertainty
  - Captures our ignorance regarding which of all possible models from a class of models generated the data we have
  - By increasing the amount of data, epistemic uncertainty can be explained away
  - Why?

Types of Uncertainty

- Epistemic uncertainty
  - Captures our ignorance regarding which of all possible models from a class of models generated the data we have
  - By increasing the amount of data, epistemic uncertainty can be explained away
  - Why? The more data we have the fewer are the possible models that could in fact generate all the data
- Aleatoric uncertainty
  - Uncertainty due to the nature of the data.
  - If we predict depth from images, for instance, highly specular surfaces make it very hard to predict depth. Or if we detect objects, severe occlusions make it very difficult to predict the object class and the precise bounding box
  - Better features reduce aleatoric uncertainty
- Predictive Uncertainty = Epistemic uncertainty + Aleatoric uncertainty

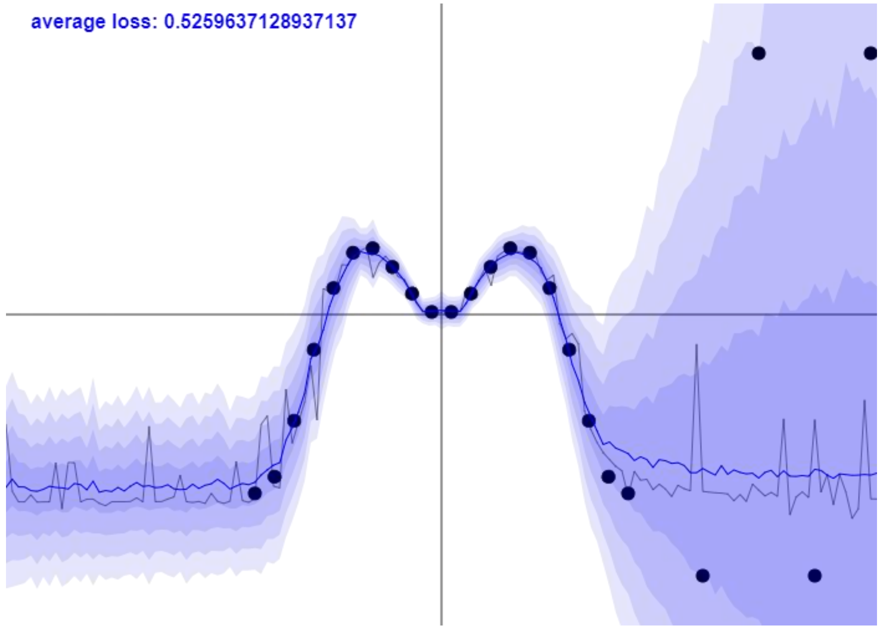
Types of Uncertainty


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

Epistemic uncertainty

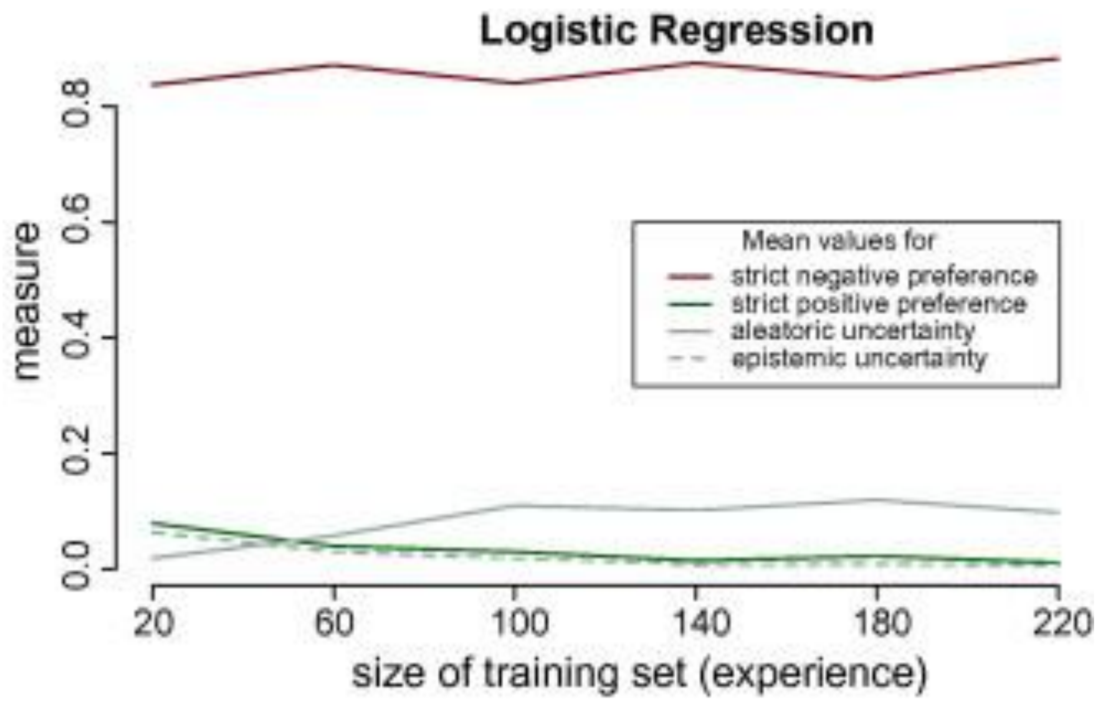
- Important to consider modelling when
  - we have safety-critical applications
  - the datasets are small

Should I give the drug or not?



Aleatoric uncertainty

- Important to consider modelling when
  - Large datasets → small epistemic uncertainty
  - Real-time → aleatoric models can be deterministic (no Monte Carlo sampling needed)



Data-dependent aleatoric uncertainty

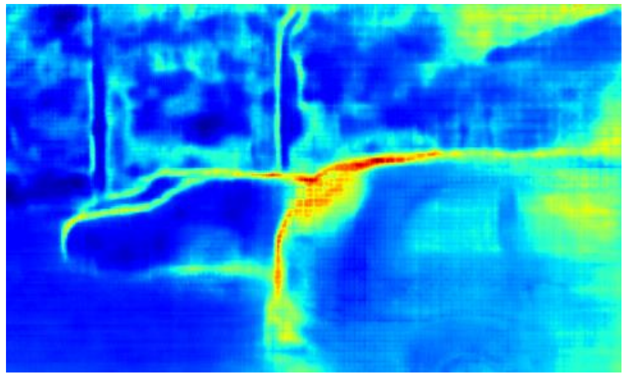
- Also called heteroscedastic aleatoric uncertainty
- The uncertainty is in the raw inputs
- Data-dependent aleatoric uncertainty can be one of the model outputs

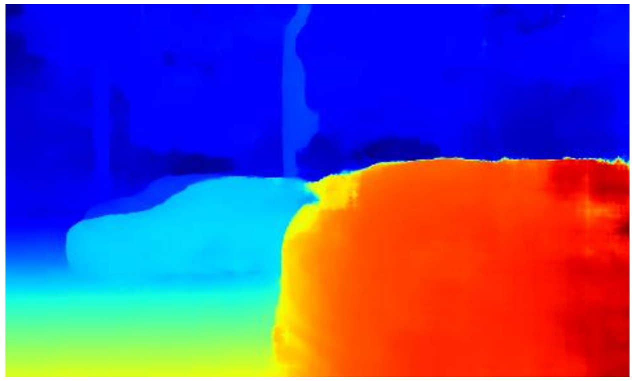
Task-dependent aleatoric uncertainty

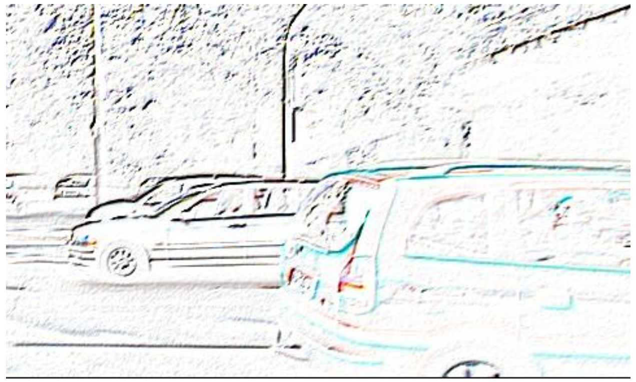
- Also called homoscedastic aleatoric uncertainty
- It is not a model output, it relates to the uncertainty that a particular task might cause
  - For instance, for the task of depth estimation predicting depth around the edges is very hard → thus uncertain
- When having multiple tasks task-dependent aleatoric uncertainty may be reduced
  - For instance?

Task-dependent aleatoric uncertainty

Input


Uncertainty


Depth Prediction


Edge prediction as second task?

Bayesian Modelling
Variational Inference

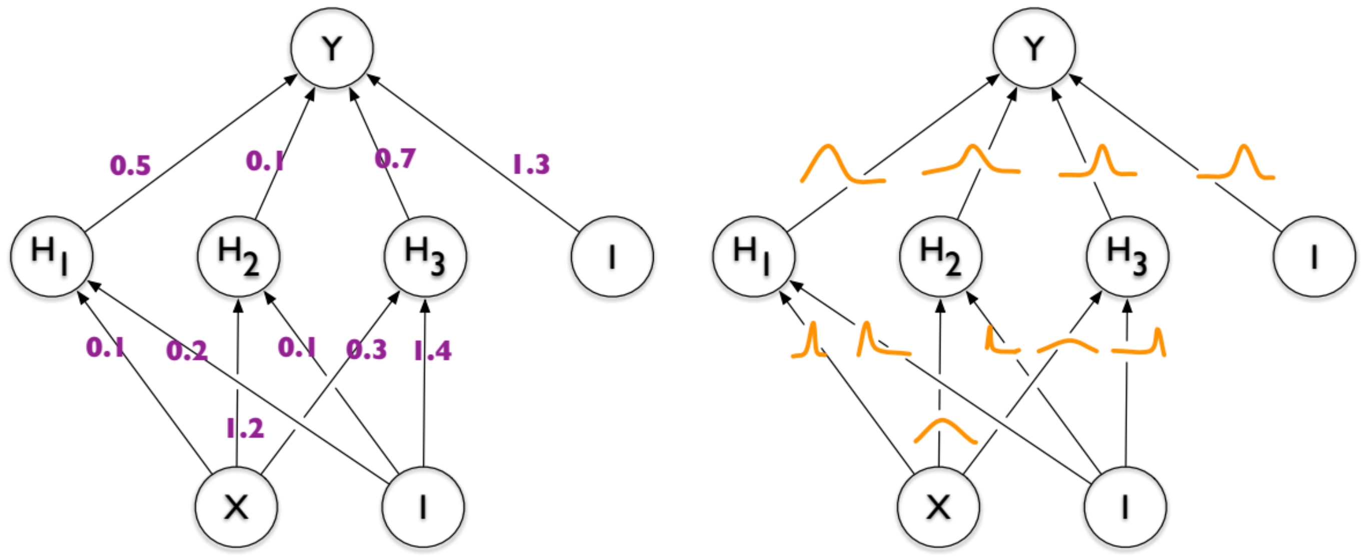


Figure 1. Left: each weight has a fixed value, as provided by classical backpropagation. Right: each weight is assigned a distribution, as provided by Bayes by Backprop.

Bayesian Deep Learning

- Deep learning provides powerful feature learners from raw data
  - But they cannot model uncertainty
- Bayesian learning provides meaningful uncertainty estimates
  - But they often rely on methods that are not scalable, e.g. Gaussian Processes
- Bayesian Deep Learning combines the best of two worlds
  - Hierarchical representation power
  - Outputs complex multi-modal distributions

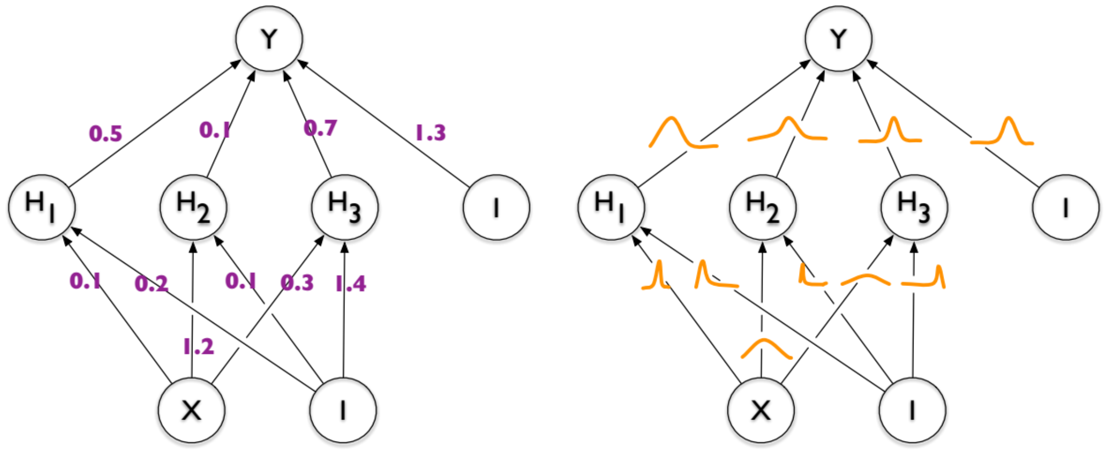


Figure 1. Left: each weight has a fixed value, as provided by classical backpropagation. Right: each weight is assigned a distribution, as provided by Bayes by Backprop.

Bayesian Deep Learning: Goal?

- Deep Networks: filters & architecture
- Standard Deep Networks → single optimal value per filter
- A Bayesian approach associates a distribution per latent variable/filter

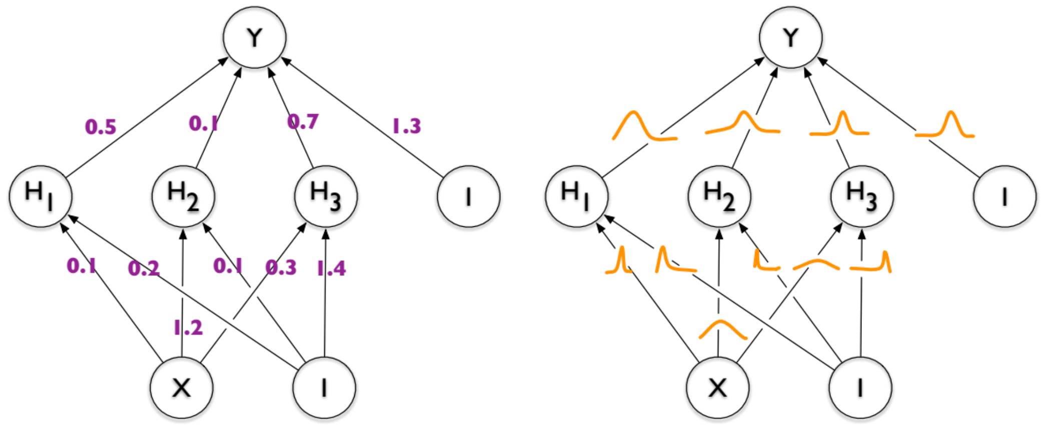


Figure 1. Left: each weight has a fixed value, as provided by classical backpropagation. Right: each weight is assigned a distribution, as provided by Bayes by Backprop.

Modelling data-dependent aleatoric uncertainty

- We add a variance term per data point to our loss function

$$\mathcal{L} = \frac{\|y_i - \hat{y}_i\|^2}{2\sigma_i^2} + \log \sigma_i$$

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

Modelling data-dependent aleatoric uncertainty

- We add a variance term per data point to our loss function

$$\mathcal{L} = \frac{\|y_i - \widehat{y}_i\|^2}{2\sigma_i^2} + \log \sigma_i$$

- What is the role of $2\sigma_i^2$?
  - When the nominator becomes large, the network may choose to shrink the loss by increasing the output variance σ_i
- But then what about $\log \sigma_i$?
  - Without it the network will always tend to return high variance

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

Modelling task-dependent aleatoric uncertainty

- Similar to the data-dependent uncertainty

$$\mathcal{L} = \frac{||y_i - \widehat{y_i}||^2}{2\sigma^2} + \log \sigma$$

- The only difference is that now the variance is a learnable parameter shared by all task data points
- One can use task-dependent uncertainties to weigh multiple tasks

Results

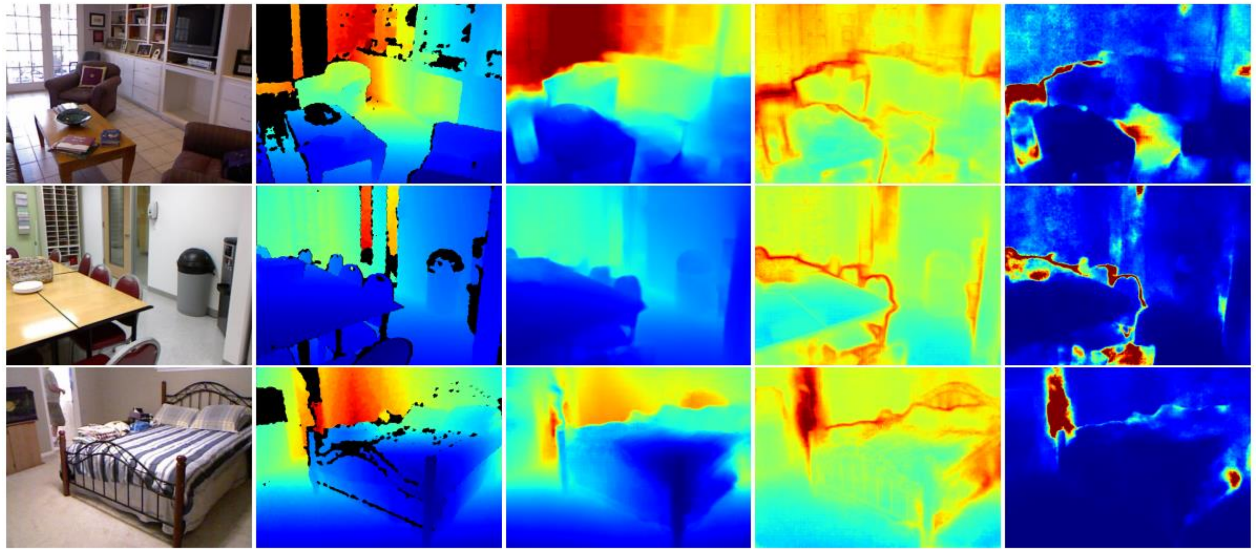


Figure 5: NYUv2 Depth results. From left: input image, ground truth, depth regression, aleatoric uncertainty, and epistemic uncertainty.

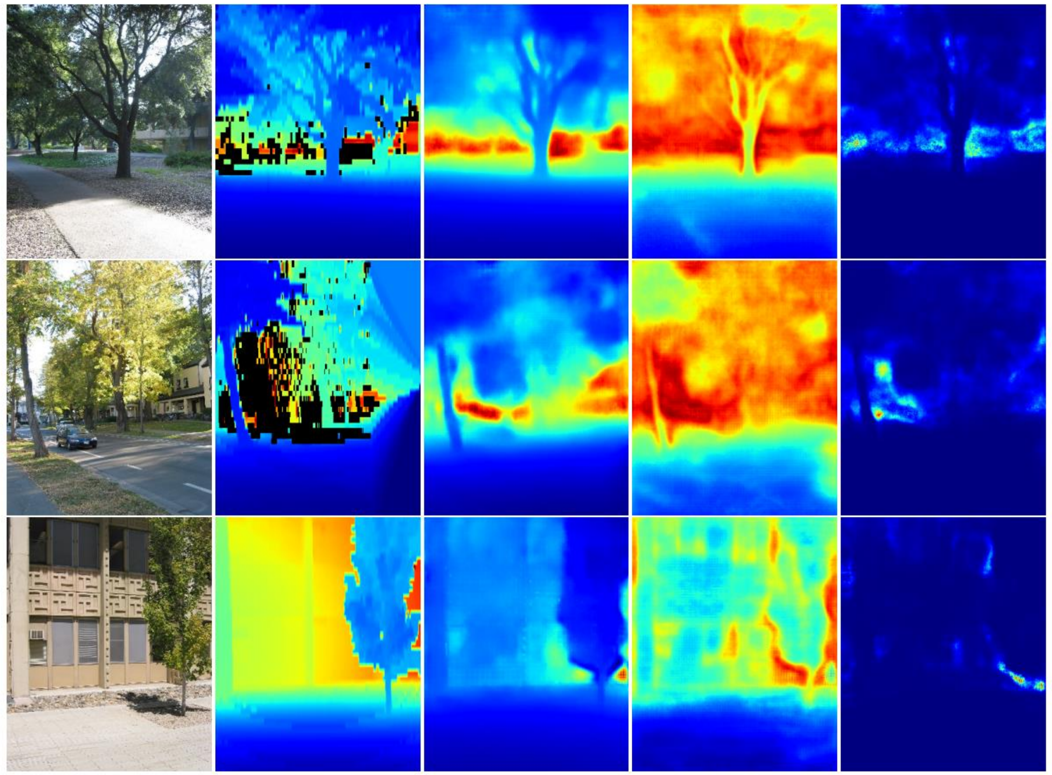


Figure 6: Qualitative results on the Make3D depth regression dataset. Left to right: input image, ground truth, depth prediction, aleatoric uncertainty, epistemic uncertainty. Make3D does not provide labels for depth greater than 70m, therefore these distances dominate the epistemic uncertainty signal. Aleatoric uncertainty is prevalent around depth edges or distant points.

Modelling epistemic uncertainty

- Epistemic uncertainty is harder to model

$$p(w|x,y) = \frac{p(x,y|w)p(w)}{\int_w p(x,y|w)p(w)\,dw}$$

- Computing the posterior densities is usually intractable for complex functions like neural networks

Monte Carlo (MC) Dropout

- Long story short
- To get uncertainty estimates for your Deep Net, keep dropout during testing
- The uncertainties derived from there approximate the uncertainties you would obtain from a Variational Inference Framework

Y. Gal, Z. Ghahramani, Dropout as a Bayesian Approximation Representing Model Uncertainty, ICML 2016

Epistemic uncertainty: Monte Carlo (MC) Dropout!

- Variational Inference assumes a (approximate) posterior distribution to approximate the true posterior

- Dropout turns on or off neurons based on probability distribution (Bernoulli)

- The Bernoulli distribution can be used as the variational distribution → MC Dropout

Y. Gal, Z. Ghahramani, Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning, MLR 2016

Bayesian Neural Networks as (approximate) Gaussian Processes

- Expected model output described by
  - Predictive mean $\mathbb{E}(y^*)$
  - Predictive variance $\mathrm{Var}(y^*)$
- Starting from a Gaussian Process and deriving a variational approximation, one arrives at a Dropout Neural Network
- The model precision τ (inverse of variance $\tau = 1/\sigma^2$) is equivalent to

$$\tau = \frac{l^2 p}{2N\lambda}$$

  - l is the length-scale: large for high-frequency data, small for low-frequency data
  - p the dropout **survival** rate
  - λ is the learning rate

Deep Gaussian Processes

- The predictive probability of a Deep GP is

$$p(y|x,X,Y) = \int p(y|x,\omega)p(\omega|X,Y)d\omega$$

  - The ω is our model weights, which are distributions
  - Thus, to find the predictive probability of a new point we must integrate over all possible ω in the distribution

- The likelihood term $p(y|x,\omega)$ is Gaussian

$$p(y|x,\omega) = N(y; \hat{y}(x,\omega), \tau^{-1})$$

Deep Gaussian Processes

- The predictive probability of a Deep GP is

$$p(y|x,X,Y) = \int p(y|x,\omega)p(\omega|X,Y)d\omega$$

  - The ω is our model weights, which are distributions
  - Thus, to find the predictive probability of a new point we must integrate over all possible ω in the distribution

- The likelihood term $p(y|x,\omega)$ is Gaussian

$$p(y|x,\omega) = N(y; \hat{y}(x,\omega), \tau^{-1} I_D)$$

- The mean $\hat{y}(x,\omega)$ is modelled by a Deep Net

$$\hat{y}(x,\omega) = \sqrt{1/K_L}\, W_L \sigma(\ldots \sqrt{1/K_1}\, \sigma(W_1 x + m_1))$$

  - $\omega = \{W_1, W_2, \ldots, W_L\}$

Deep Gaussian Processes

- The predictive probability of a Deep GP is

$$p(y|x, X, Y) = \int p(y|x, \omega) p(\omega|X, Y) d\omega$$

- The posterior is intractable → we approximate by a variational approximation
- The $q(\omega)$ is defined in this model as

$$W_i = M_i \cdot \mathrm{diag}\left(\left[\mathrm{z_{i,j}}\right]_1^{\mathrm{K_i}}\right)$$

$$z_{i,j} \sim \mathrm{Bernoulli(p_i)}$$

  - Columns of M_i are randomly set to 0
  - The $z_{i,j} = 0$ basically corresponds to dropping the j-th neuron in the $i-1$ layer

Deep Gaussian Processes

- The predictive probability of a Deep GP is

$$p(y|x, X, Y) = \int p(y|x, \omega)p(\omega|X, Y)d\omega$$

- Once more, we minimize the KL divergence
- $\mathcal{L} = -\int q(\omega)\log p(Y|X, \omega)d\omega + \mathrm{KL}(q(\omega)||p(\omega))$
- How do we get the first term?

Deep Gaussian Processes

- The predictive probability of a Deep GP is
$$p(y|x, X, Y) = \int p(y|x, \omega) p(\omega|X, Y) d\omega$$
- Once more, we minimize the KL divergence
- $\mathcal{L} = -\int q(\omega) \log p(Y|X, \omega) d\omega + \mathrm{KL}(q(\omega) || p(\omega))$
- How do we get the first term?
- We approximate with a Monte Carlo sample $\omega_n \sim q(\omega)$
  - A dropout round
- How do we get the second term?

Deep Gaussian Processes

- The predictive probability of a Deep GP is

$$p(y|x,X,Y) = \int p(y|x,\omega)p(\omega|X,Y)d\omega$$

- Once more, we minimize the KL divergence
- $\mathcal{L} = -\int q(\omega)\log p(Y|X,\omega)d\omega + \mathrm{KL}(q(\omega)||p(\omega))$
- How do we get the first term?
- We approximate with a Monte Carlo sample $\omega_n \sim q(\omega)$
  - A dropout round
- How do we get the second term?
- Again we approximate and arrive at

$$\mathrm{KL}(q(\omega)||p(\omega)) \sim \sum_{i=1}^{L} \frac{p_i l^2}{2} |M_i|_2^2 + \frac{l^2}{2} |m\_i|_2^2$$

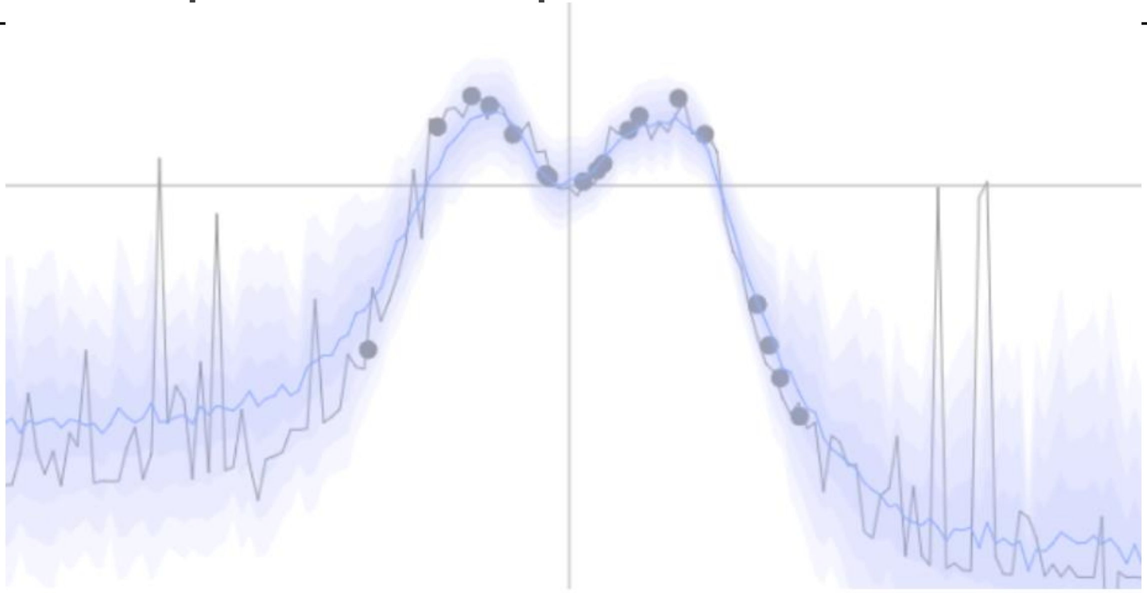
Predictive mean and variance in MC Dropout Deep Nets

$$\tau = \frac{l^2 p}{2N\lambda}$$

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

$$\mathbb{E}\left(y^{*2}\right) = \tau^{-1}\mathbf{I}_{\mathrm{D}} + \frac{1}{\mathrm{T}}\sum_{t=1}^{T} \hat{y}_t{}^*(x^*)^T \hat{y}_t{}^*(x^*)$$

$$\mathrm{Var}(y^*) = \mathbb{E}\left(y^{*2}\right) - \mathbb{E}(y^*)^T \mathbb{E}(y^*)$$



$\mathrm{Var}(y^*)$ equals the sample variable after T stochastic forward passes, plus the inverse model precision

Demo

Dropout for Bayesian Uncertainty in practice

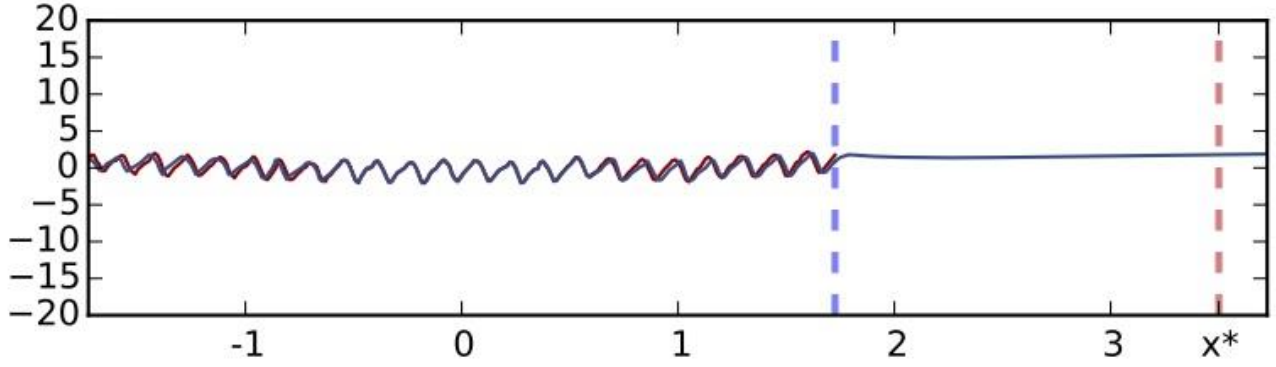
- Use dropout in all layers both during training **and testing**
- At test time repeat dropout T times (e.g., 10) and look at **mean** and **sample variance**

- Pros: Very easy to train
- Pros: Easy to convert a standard network to a Bayesian Network
- Pros: No need for an inference network $q_w(\varphi)$

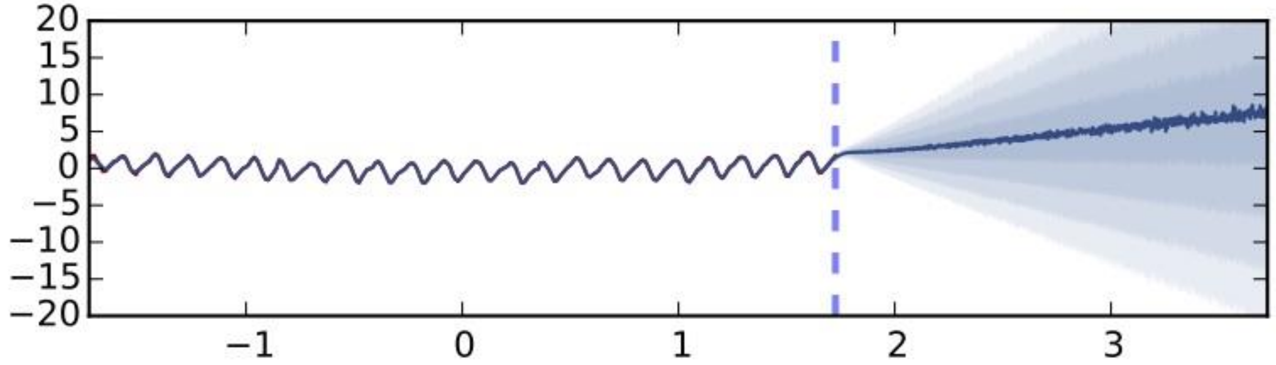
- Cons: Requires weight sampling also during testing → expensive

Example

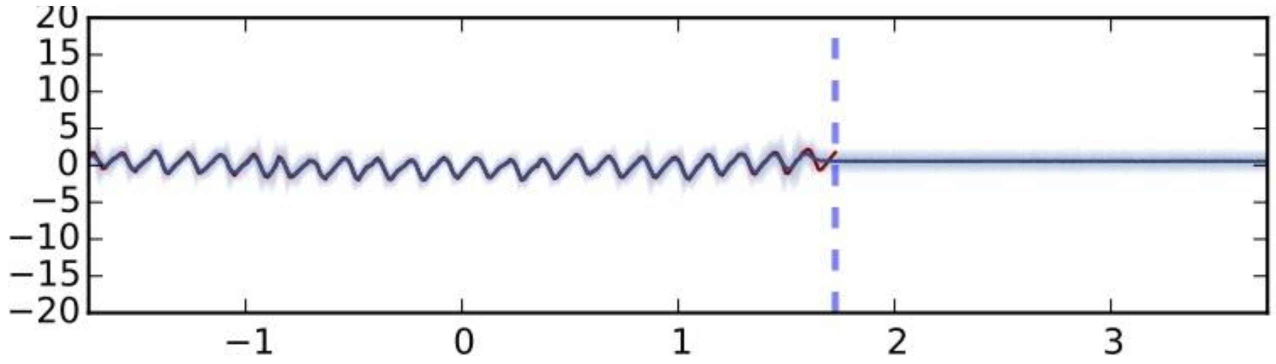
Prediction in a 5-layer ReLU neural network with dropout



Using 100-trial MC dropout



Using 100-trial MC dropout with $\mathbf{tanh}$ nonlinearity

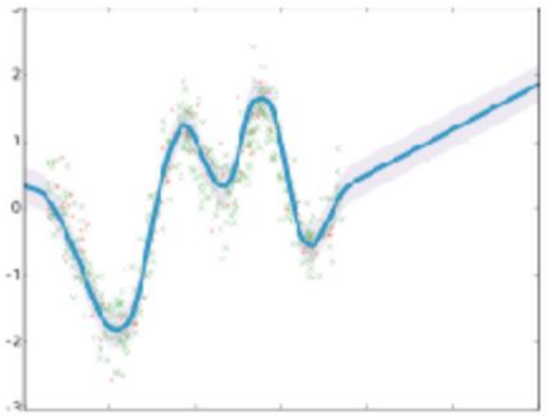


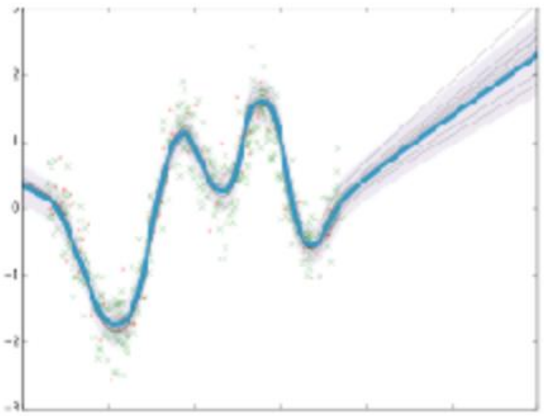
Tricks of the trade

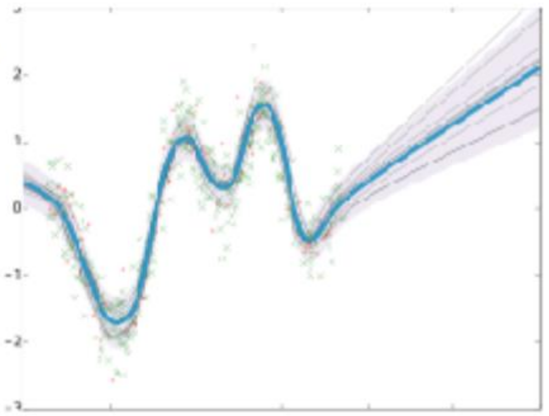
- Over-parameterized models give better uncertainty estimates, as they capture a bigger class of data

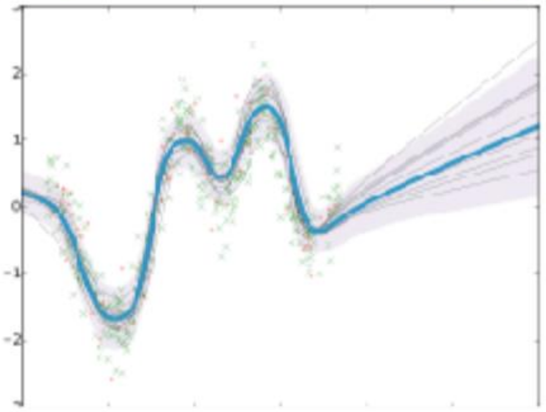
- Large models need higher dropout rates for meaningful uncertainty
  - Large models tend to push $p \rightarrow 0.5$
  - For smaller models lower dropout rates reduce uncertainty estimates

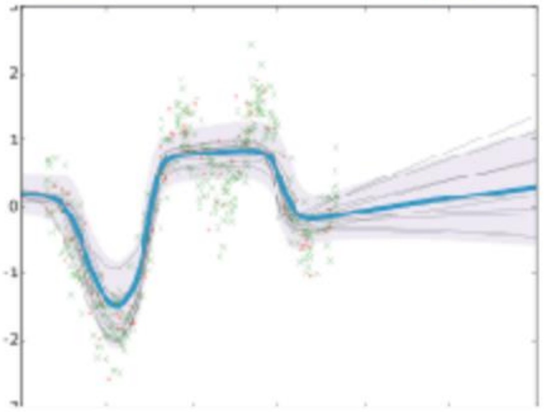
MC Dropout rates


(a) $p = 0$


(b) $p = 0.1$


(c) $p = 0.25$


(d) $p = 0.5$


(e) $p = 0.75$

Bayes by Backprop

- Start from a Deep Network with a distribution on its weights
- Similar to VAE, what is logical to minimize?

C. Blundell, J. Cornebise, K. Kavukcuoglu, D. Wierstra, Weight Uncertainty in Neural Networks, ICML 2015

Bayes by Backprop

- Start from a Deep Network with a distribution on its weights
- Similar to VAE, what is logical to minimize?
- The KL between approximate and true weight posteriors

$$KL(q(w|\theta)||p(w|\mathcal{D})) = KL(q(w|\theta)||p(w)) - \int_w q(w|\theta) \log p(\mathcal{D}|w)\, dw$$

- What do these two terms look like?

C. Blundell, J. Cornebise, K. Kavukcuoglu, D. Wierstra, Weight Uncertainty in Neural Networks, ICML 2015

Bayes by Backprop

- Start from a Deep Network with a distribution on its weights
- Similar to VAE, what is logical to minimize?
- The KL between approximate and true weight posteriors

$$KL\big(q(w|\theta)||p(w|\mathcal{D})\big) = KL(q(w|\theta)||p(w)) - \int_w q(w|\theta) \log p(\mathcal{D}|w)\, dw$$

- What do these two terms look like?
- Prior term pushing approximate posterior towards prior $p(w)$
- The data term making sure the weights explain data well

C. Blundell, J. Cornebise, K. Kavukcuoglu, D. Wierstra, Weight Uncertainty in Neural Networks, ICML 2015

Bayes by Backprop

- The KL between approximate and true weight posteriors

$$KL(q(w|\theta)||p(w|\mathcal{D})) = KL(q(w|\theta)||p(w)) - \int_w q(w|\theta)\log p(\mathcal{D}|w)\,dw$$

- How could we efficiently compute these integrals?

C. Blundell, J. Cornebise, K. Kavukcuoglu, D. Wierstra, Weight Uncertainty in Neural Networks, ICML 2015

Bayes by Backprop

- The KL between approximate and true weight posteriors

$$KL(q(w|\theta)||p(w|\mathcal{D})) = KL(q(w|\theta)||p(w)) - \int_w q(w|\theta) \log p(\mathcal{D}|w)\, dw$$

- How could we efficiently compute these integrals?
- Approximate with Monte Carlo Integration
- Sample a single weight value w_s from our posterior $q(w|\theta)$
  - e.g., a Gaussian
- Then, compute the MC ELBO:

C. Blundell, J. Cornebise, K. Kavukcuoglu, D. Wierstra, Weight Uncertainty in Neural Networks, ICML 2015

Bayes by Backprop

- Then, compute the MC ELBO:

$$\mathcal{L} = \log q(w_s||\theta) - \log p(w_s) - \log p(\mathcal{D}|w_s)$$

- Same for backprop
- What’s so special about $\log q(w_s||\theta) - \log p(w_s)$?

C. Blundell, J. Cornebise, K. Kavukcuoglu, D. Wierstra, Weight Uncertainty in Neural Networks, ICML 2015

Bayes by Backprop

- Then, compute the MC ELBO:

$$\mathcal{L} = \log q(w_s||\theta) - \log p(w_s) - \log p(\mathcal{D}|w_s)$$

- Same for backprop
- What’s so special about $\log q(w_s||\theta) - \log p(w_s)$?
- Monte Carlo approximation of the complexity cost as well
- Not confined to specific pdfs anymore

C. Blundell, J. Cornebise, K. Kavukcuoglu, D. Wierstra, Weight Uncertainty in Neural Networks, ICML 2015

Bayes by Backprop

- Assume a Gaussian variational posterior on the weights
- Each weight is then parameterized as

$$w = \mu + \varepsilon \cdot \sigma$$

where σ is ρ-parameterized by the softplus

$$\sigma = \log(1 + \exp(\rho))$$

- Why?

Bayes by Backprop

- Assume a Gaussian variational posterior on the weights
- Each weight is then parameterized as

$$w = \mu + \varepsilon \circ \sigma$$

where σ is ρ-parameterized by the softplus

$$\sigma = \log(1 + \exp(\rho))$$

- Why?
- With this parameterization the standard deviation is always positive
- Then we optimize the ELBO
- In the end we learn an ensemble of networks, since we can sample as many weights as we want

Bayes by Backprop - Algorithm

1. Sample $\varepsilon \sim N(0,1)$
2. Set $w = \mu + \varepsilon \cdot \log(1 + \exp(\rho))$
3. Set $\theta = \{\mu, \rho\}$
4. Let $\mathcal{L}(w, \theta) = \log q(w|\theta) - \log p(w) p(x|w)$
5. Calculate gradients

$$\nabla_\mu = \frac{\partial \mathcal{L}}{\partial w}\frac{\partial w}{\partial \mu} + \frac{\partial \mathcal{L}}{\partial \mu}$$
$$\nabla_\rho = \frac{\partial \mathcal{L}}{\partial w}\frac{\varepsilon}{1 + \exp(-\rho)} + \frac{\partial \mathcal{L}}{\partial \rho}$$

7. Last, update the variational parameters

$$\mu_{t+1} = \mu_t - \eta_t \nabla_\mu$$
$$\rho_{t+1} = \rho_t - \eta_t \nabla_\rho$$

Bayes by Backprop: Results

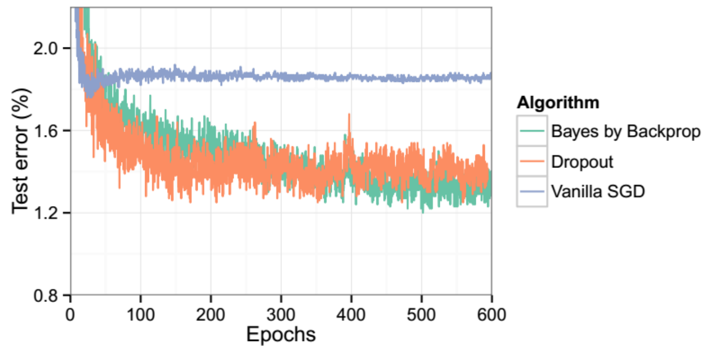


Figure 2. Test error on MNIST as training progresses.

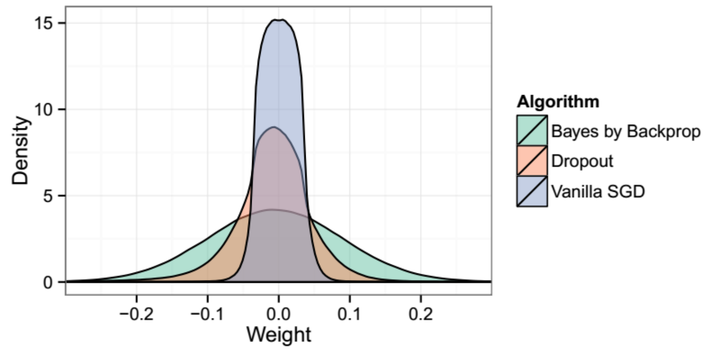


Figure 3. Histogram of the trained weights of the neural network, for Dropout, plain SGD, and samples from Bayes by Backprop.

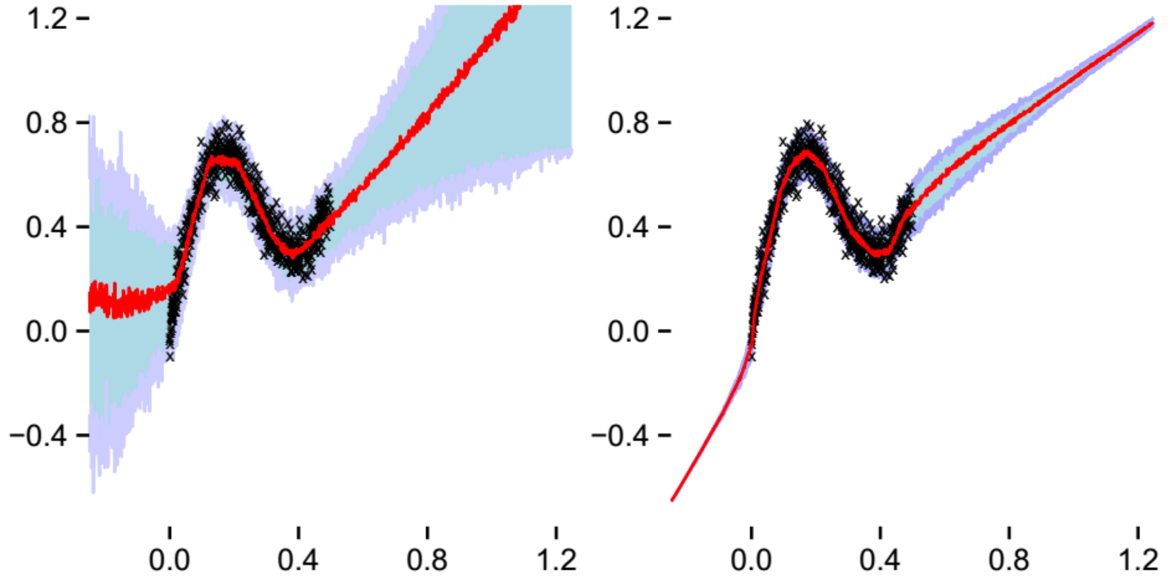


Figure 5. Regression of noisy data with interquatile ranges. Black crosses are training samples. Red lines are median predictions. Blue/purple region is interquartile range. Left: Bayes by Backprop neural network, Right: standard neural network.

Criticism for Bayes by Backprop and MC dropout

- They both assume rather simple posterior families with Mean-Field approximation
- Basically, each parameter/weight is assumed independent from all others
  - Clearly unrealistic in the context of neural networks, where everything is connected to everything
- So, they do not really capture dependencies between weights
- Also, for MC dropout in the limit of many samples the posterior may not concentrate asymptotically
  - So the σ does not reduce by the number of samples, which is what you expect (lower uncertainty) with more samples=

Criticism for Bayes by Backprop and MC dropout

- When visualizing uncertainty in simple 1D problems, Backprop by Bayes is usually *too* certain.
- Also, it is often beaten by other methods like Multiplicative Normalizing Flows or Matrix variate Gaussian approaches (check C. Louizos and M. Welling)

Bayesian Neural Network Compression

- Revisit connection between minimum description length and variational inference
- Minimum Description Length: best model uses the minimum number of bits to communicate the model complexity $\mathcal{L}^{\mathrm{C}}$ and the model error $\mathcal{L}^{\mathrm{E}}$

$$\mathcal{L}(\varphi) = \underbrace{\mathbb{E}_{q_w(\varphi)}[\log p(D|w)]}_{\mathcal{L}^{\mathrm{E}}} + \underbrace{\mathbb{E}_{q_w(\varphi)}[\log p(w)] + H(q_w(\varphi))}_{\mathcal{L}^{\mathrm{C}}}$$

- Use sparsity-inducing priors for groups of weights → prune weights that are not necessary for the model

C. Louizos, K. Ullrich, M. Welling, Bayesian Compression for Deep Learning, NIPS 2017

Bayesian Neural Network Compression

- Define the prior over weights

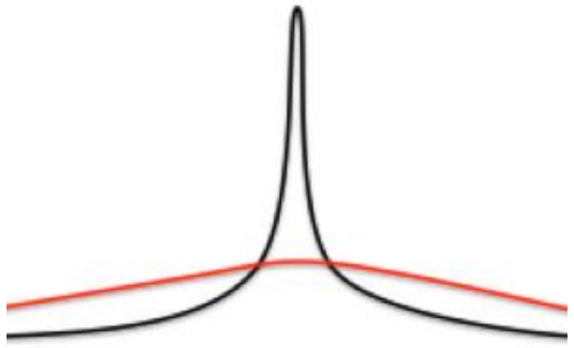
$$z \sim p(z)$$
$$w \sim N(w; 0, z^2)$$

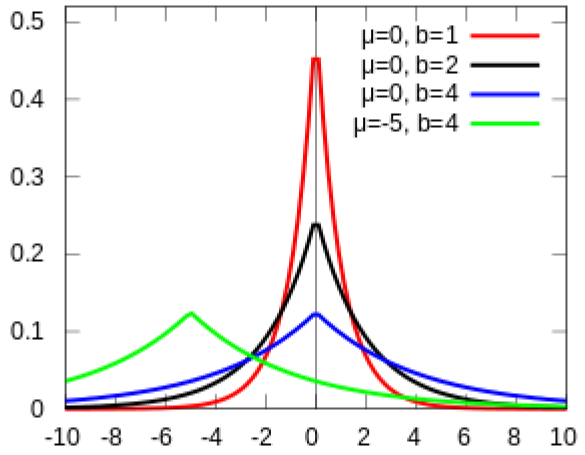
- The scales of the weight prior have a prior themselves

- Goal: by treating the scales as random variables the marginal $\mathrm{p}(w)$ can be set to have heavy tails → more density near 0

- Several distributions possible to serve as priors

Spike-and-slab distribution





Laplace distribution (Lasso)

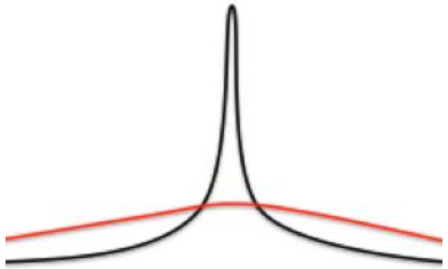
Sparse-inducing distributions

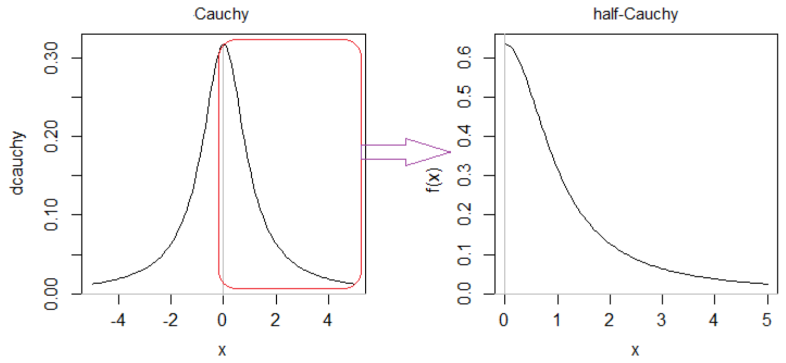
Spike-and-slab distribution

Mixture of a very spiky and a very broad Gaussian

Or a mixture of a δ-spike at 0, and a slab on the real line

This would lead to large number of possible models: 2^M for M parameters



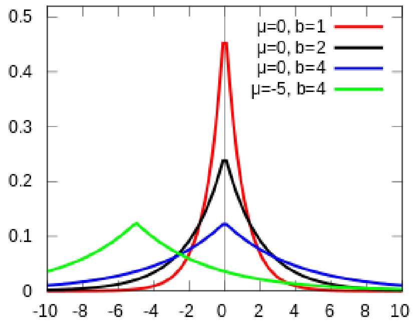


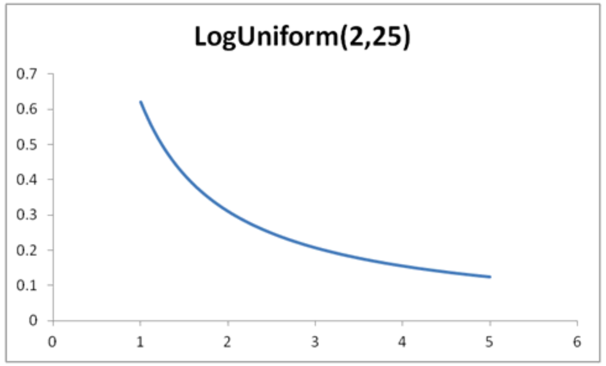
Half-Cauchy

Laplace distribution (Lasso)

$$p(z^2; \lambda) = \exp(\lambda)$$

Lasso focuses on shrinking the larger values



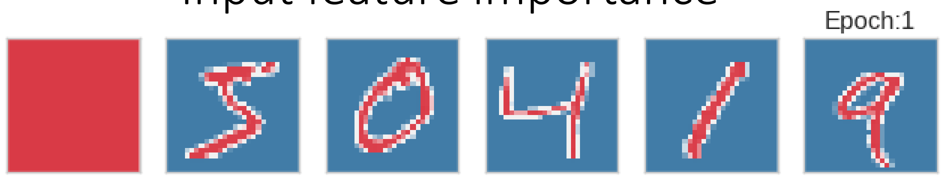


Log-Uniform

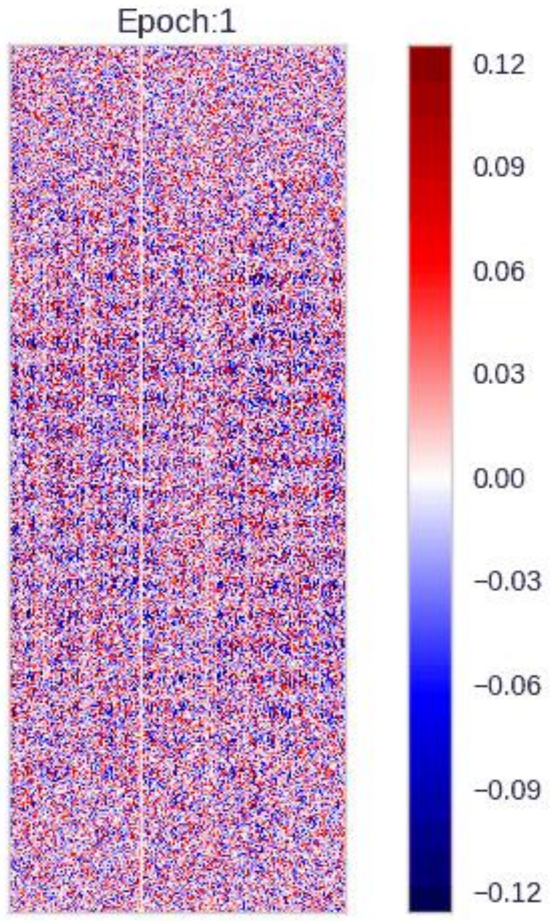
Bayesian Neural Network Compression

- 700x compression
- 50x speed up

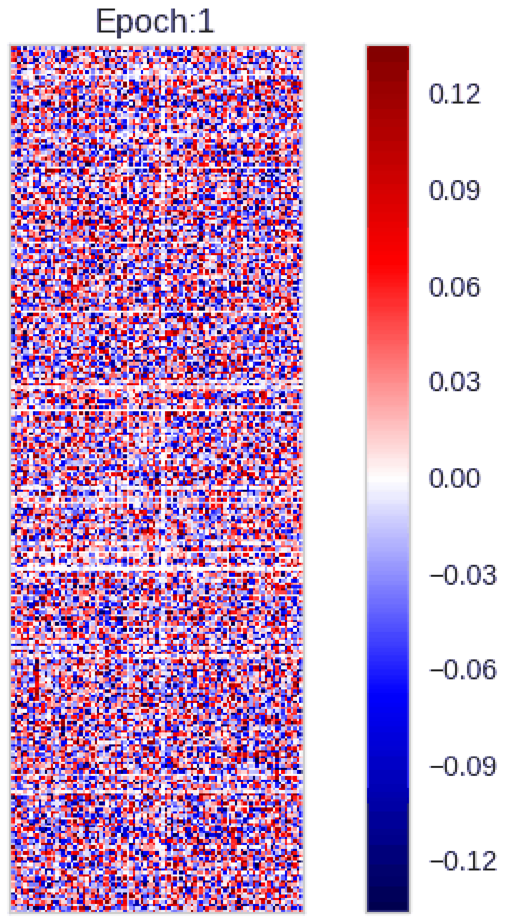
Input feature importance



First layer



Second layer

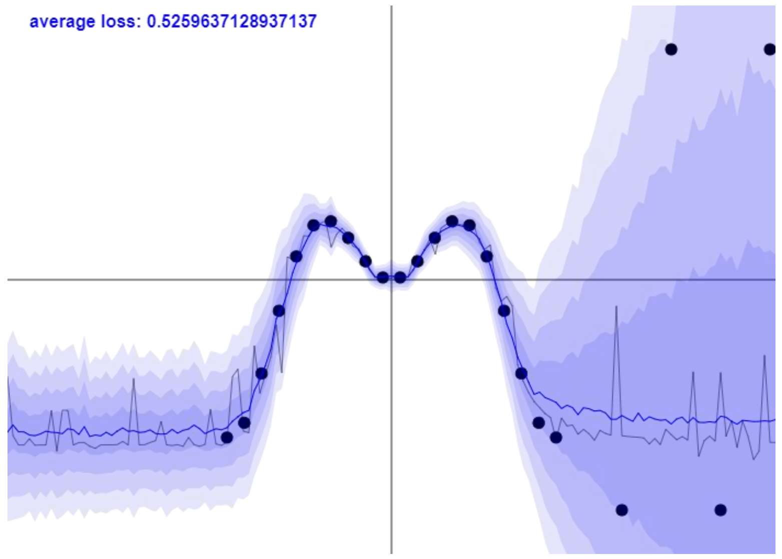


Some open questions

- Hard to model epistemic uncertainty real-time
  - Typically, Monte Carlo approximations are required
  - Efficiency and uncertainty is needed for robotics, self-driving, health AI, etc

- No benchmarks to fairly evaluate

- Inference techniques are still not good enough



Summary

- Why Bayesian Deep Learning?
- Types of uncertainty
- Bayesian Neural Networks
- Backprop by Bayes
- MC Dropout