FLATIRON INSTITUTE
Center for Computational Astrophysics

Bayesian NN

Food and ML Club

Which uncertainty?

Aleatoric

- captures noise inherent in the observations, e.g. sensor noise
- cannot be reduced even if more data were to be collected

Epistemic

- uncertainty in the model parameters
- captures our ignorance about which model generated our collected data
- can be explained away given enough data

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.
Aleatoric uncertainty on the other hand is modeled by placing a distribution over the output of the model.

Aleatoric Uncertainty

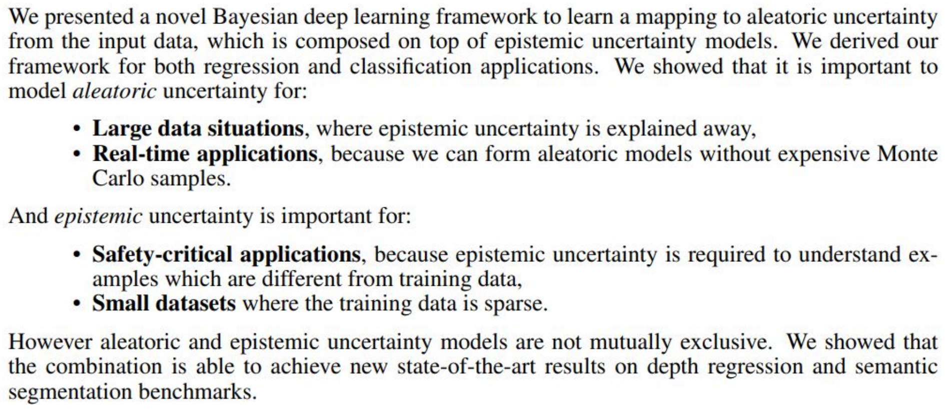
What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?

https://arxiv.org/pdf/1703.04977.pdf

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

This acts similarly to an intelligent robust regression function. This makes the model more robust to noisy data.

- Aleatoric uncertainty cannot be explained away with more data
- Aleatoric uncertainty does not increase for out-of-data examples (situations different from training set), whereas epistemic uncertainty does.
- when only one uncertainty is explicitly modeled, it attempts to compensate for the lack of the alternative uncertainty


We presented a novel Bayesian deep learning framework to learn a mapping to aleatoric uncertainty from the input data, which is composed on top of epistemic uncertainty models. We derived our framework for both regression and classification applications. We showed that it is important to model *aleatoric* uncertainty for:

- **Large data situations**, where epistemic uncertainty is explained away,
- **Real-time applications**, because we can form aleatoric models without expensive Monte Carlo samples.

And *epistemic* uncertainty is important for:

- **Safety-critical applications**, because epistemic uncertainty is required to understand examples which are different from training data,
- **Small datasets** where the training data is sparse.

However aleatoric and epistemic uncertainty models are not mutually exclusive. We showed that the combination is able to achieve new state-of-the-art results on depth regression and semantic segmentation benchmarks.

BNN model epistemic uncertainty

$$p(y|x, \mathcal{D}) = \int p(y|x, w)p(w|\mathcal{D})dw \, .$$

- marginalization
- Bayesian model averaging (BMA)
- challenging due to high dimensional posterior p(w|D)

Stochastic variational inference methods

1. Bayes by backprop $q(\mathbf{w}|\theta)$

   https://arxiv.org/pdf/1505.05424.pdf

$$\begin{aligned}\mathbf{w}^{\text{MAP}} &= \arg\max_{\mathbf{w}} \log P(\mathbf{w}|\mathcal{D}) \\ &= \arg\max_{\mathbf{w}} \log P(\mathcal{D}|\mathbf{w}) + \log P(\mathbf{w}).\end{aligned}$$

$$\begin{aligned}\theta^\star &= \arg\min_\theta \mathrm{KL}[q(\mathbf{w}|\theta)||P(\mathbf{w}|\mathcal{D})] \\ &= \arg\min_\theta \int q(\mathbf{w}|\theta) \log \frac{q(\mathbf{w}|\theta)}{P(\mathbf{w})P(\mathcal{D}|\mathbf{w})} \mathbf{dw} \\ &= \arg\min_\theta \mathrm{KL}\left[q(\mathbf{w}|\theta) \;||\; P(\mathbf{w})\right] - \mathbb{E}_{q(\mathbf{w}|\theta)}\left[\log P(\mathcal{D}|\mathbf{w})\right]\end{aligned}$$

$$\begin{aligned}\mathcal{F}(\mathcal{D},\theta) \approx \sum_{i=1}^{n} \log q(\mathbf{w}^{(i)}|\theta) - \log P(\mathbf{w}^{(i)}) \\ - \log P(\mathcal{D}|\mathbf{w}^{(i)})\end{aligned}$$

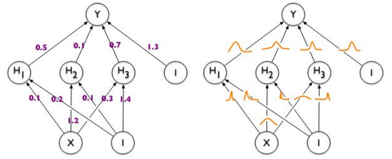


Figure 1. Left: each weight has a fixed value, as provided by classical backpropagation. Right: each weight is assigned a distribution, as provided by Bayes by Backprop.

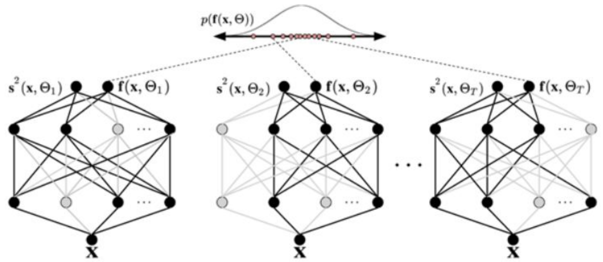
Stochastic variational inference methods

2. Monte Carlo Dropout

https://arxiv.org/pdf/1506.02142.pdf
https://arxiv.org/pdf/1506.02157.pdf (appendix)

$$\mathcal{L}(\theta, p) = -\frac{1}{N}\sum_{i=1}^{N} \log p(\mathbf{y}_i | \mathbf{f}^{\widehat{\mathbf{W}}_i}(\mathbf{x}_i)) + \frac{1-p}{2N}||\theta||^2$$

```python
class MonteCarloDropout(keras.layers.Dropout):
  def call(self, inputs):
    return super().call(inputs, training=True)
```


$p(\mathbf{f}(\mathbf{x}, \Theta))$
$\mathbf{s}^2(\mathbf{x}, \Theta_1)$
$\mathbf{f}(\mathbf{x}, \Theta_1)$
$\mathbf{s}^2(\mathbf{x}, \Theta_2)$
$\mathbf{f}(\mathbf{x}, \Theta_2)$
$\mathbf{s}^2(\mathbf{x}, \Theta_T)$
$\mathbf{f}(\mathbf{x}, \Theta_T)$
$\mathbf{x}$
$\mathbf{x}$
$\mathbf{x}$

SGD based BBN

Stochastic Gradient Descent as Approximate Bayesian Inference

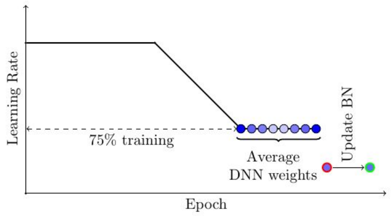
https://arxiv.org/pdf/1704.04289.pdf

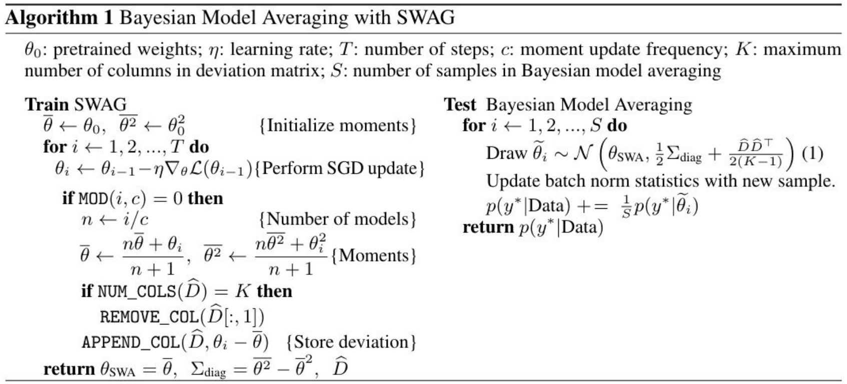
- Constant SGD first marches toward an optimum of the objective function and then bounces around its vicinity.
- *(In contrast, traditional SGD converges to the optimum by decreasing the learning rate.)*
- **Main Idea**- constant SGD is a stochastic process with a stationary distribution, one that is centered on the optimum and that has a certain covariance structure. The main idea is that we can use this stationary distribution to approximate a posterior.

SWAG

Stochastic weight averaging- Gaussian

https://arxiv.org/pdf/1902.02476.pdf





Algorithm 1 Bayesian Model Averaging with SWAG

θ_0: pretrained weights; η: learning rate; T: number of steps; c: moment update frequency; K: maximum number of columns in deviation matrix; S: number of samples in Bayesian model averaging

```
Train SWAG
  $\overline{\theta} \leftarrow \theta_0,\ \overline{\theta^2} \leftarrow \theta_0^2$   {Initialize moments}
  for $i \leftarrow 1, 2, ..., T$ do
    $\theta_i \leftarrow \theta_{i-1} - \eta \nabla_\theta \mathcal{L}(\theta_{i-1})$   {Perform SGD update}
    if $\mathtt{MOD}(i, c) = 0$ then
      $n \leftarrow i/c$   {Number of models}
      $\overline{\theta} \leftarrow \frac{n\overline{\theta} + \theta_i}{n+1},\ \overline{\theta^2} \leftarrow \frac{n\overline{\theta^2} + \theta_i^2}{n+1}$   {Moments}
      if $\mathtt{NUM\_COLS}(\widehat{D}) = K$ then
        $\mathtt{REMOVE\_COL}(\widehat{D}[:, 1])$
      $\mathtt{APPEND\_COL}(\widehat{D}, \theta_i - \overline{\theta})$   {Store deviation}
  return $\theta_{\text{SWA}} = \overline{\theta},\ \Sigma_{\text{diag}} = \overline{\theta^2} - \overline{\theta}^2,\ \widehat{D}$

Test Bayesian Model Averaging
  for $i \leftarrow 1, 2, ..., S$ do
    Draw $\widetilde{\theta}_i \sim \mathcal{N}\left(\theta_{\text{SWA}}, \frac{1}{2}\Sigma_{\text{diag}} + \frac{\widehat{D}\widehat{D}^\top}{2(K-1)}\right)$ (1)
    Update batch norm statistics with new sample.
    $p(y^*|\text{Data}) \mathrel{+}= \frac{1}{S} p(y^*|\widetilde{\theta}_i)$
  return $p(y^*|\text{Data})$
```

Deep Ensembles

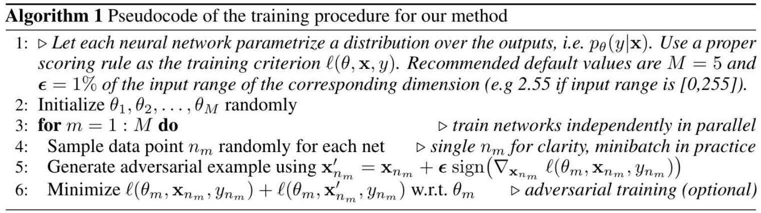
https://arxiv.org/pdf/1612.01474.pdf

https://arxiv.org/pdf/1912.02757.pdf

Proper scoring rules
https://www.cis.upenn.edu/~aaroth/courses/slides/agt17/lect23.pdf

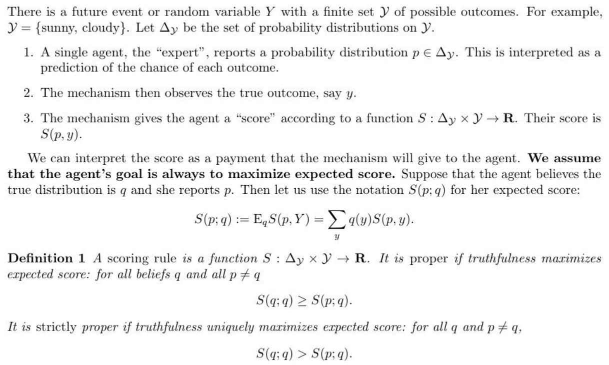
MSE is not proper scoring rule! (it is not a scoring rule)

$$-\log p_\theta(y_n|\mathbf{x}_n) = \frac{\log \sigma_\theta^2(\mathbf{x})}{2} + \frac{\left(y - \mu_\theta(\mathbf{x})\right)^2}{2\sigma_\theta^2(\mathbf{x})} + \text{constant}.$$



Algorithm 1 Pseudocode of the training procedure for our method

```
1: ▷ 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
   $\epsilon = 1\%$ of the input range of the corresponding dimension (e.g 2.55 if input range is [0,255]).
2: Initialize $\theta_1, \theta_2, \ldots, \theta_M$ randomly
3: for $m = 1 : M$ do                                          ▷ train networks independently in parallel
4:    Sample data point $n_m$ randomly for each net       ▷ single $n_m$ for clarity, minibatch in practice
5:    Generate adversarial example using $\mathbf{x}'_{n_m} = \mathbf{x}_{n_m} + \epsilon\, \text{sign}\left(\nabla_{\mathbf{x}_{n_m}} \ell(\theta_m, \mathbf{x}_{n_m}, y_{n_m})\right)$
6:    Minimize $\ell(\theta_m, \mathbf{x}_{n_m}, y_{n_m}) + \ell(\theta_m, \mathbf{x}'_{n_m}, y_{n_m})$ w.r.t. $\theta_m$     ▷ adversarial training (optional)
```



There is a future event or random variable Y with a finite set $\mathcal{Y}$ of possible outcomes. For example, $\mathcal{Y} = \{\text{sunny, cloudy}\}$. Let $\Delta_\mathcal{Y}$ be the set of probability distributions on $\mathcal{Y}$.

1. A single agent, the "expert", reports a probability distribution $p \in \Delta_\mathcal{Y}$. This is interpreted as a prediction of the chance of each outcome.
2. The mechanism then observes the true outcome, say y.
3. The mechanism gives the agent a "score" according to a function $S : \Delta_\mathcal{Y} \times \mathcal{Y} \to \mathbf{R}$. Their score is $S(p, y)$.

We can interpret the score as a payment that the mechanism will give to the agent. **We assume that the agent's goal is always to maximize expected score.** Suppose that the agent believes the true distribution is q and she reports p. Then let us use the notation $S(p;q)$ for her expected score:

$$S(p;q) := \mathrm{E}_q S(p, Y) = \sum_y q(y) S(p, y).$$

Definition 1 *A scoring rule is a function $S : \Delta_\mathcal{Y} \times \mathcal{Y} \to \mathbf{R}$. It is proper if truthfulness maximizes expected score: for all beliefs q and all $p \neq q$*

$$S(q;q) \geq S(p;q).$$

It is strictly proper if truthfulness uniquely maximizes expected score: for all q and $p \neq q$,

$$S(q;q) > S(p;q).$$

Combine ensemble networks as Mixture of Gaussians or a single Gaussian with same mean and variance

Deep Ensembles

Deep ensembles are Bayesian

https://cims.nyu.edu/~andrewgw/deepensembles/

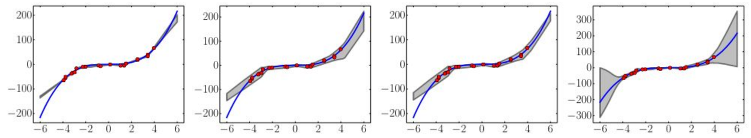
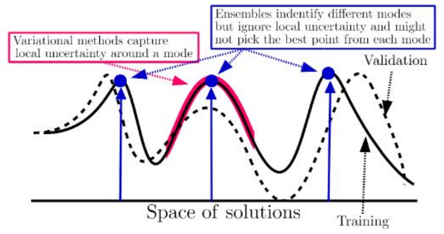


Figure 1: Results on a toy regression task: x-axis denotes x. On the y-axis, the blue line is the *ground truth* curve, the red dots are observed noisy training data points and the gray lines correspond to the predicted mean along with three standard deviations. Left most plot corresponds to empirical variance of 5 networks trained using MSE, second plot shows the effect of training using NLL using a single net, third plot shows the additional effect of adversarial training, and final plot shows the effect of using an ensemble of 5 networks respectively.

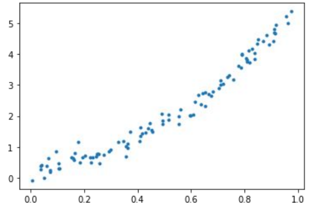


Ensembles indentify different modes but ignore local uncertainty and might not pick the best point from each mode
Variational methods capture local uncertainty around a mode
Validation
Space of solutions
Training

Figure 1: Cartoon illustration of the hypothesis. x-axis indicates parameter values and y-axis plots the negative loss $-L(\boldsymbol{\theta}, \{\boldsymbol{x}_n, y_n\}_{n=1}^{N})$ on train and validation data.

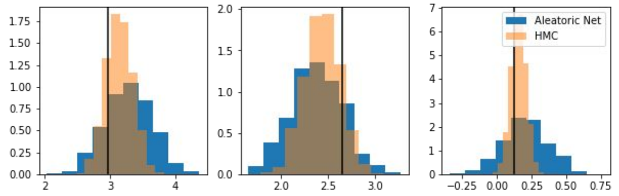
BNN is not Bayesian inference

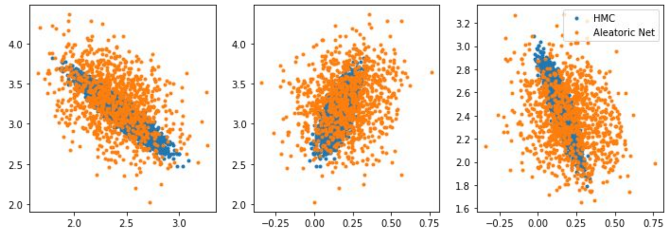
$$p(\theta|x) = \frac{p(x|\theta)}{p(x)} p(\theta)$$



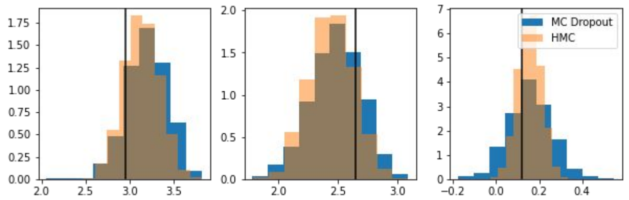
```
mu =  a * x**3 + b*x + c
dist = tfd.Normal(mu, scale=scale)
```

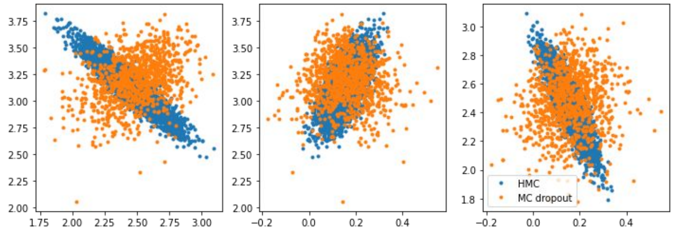
Aleatoric network





Epistemic BNN (MC dropout)



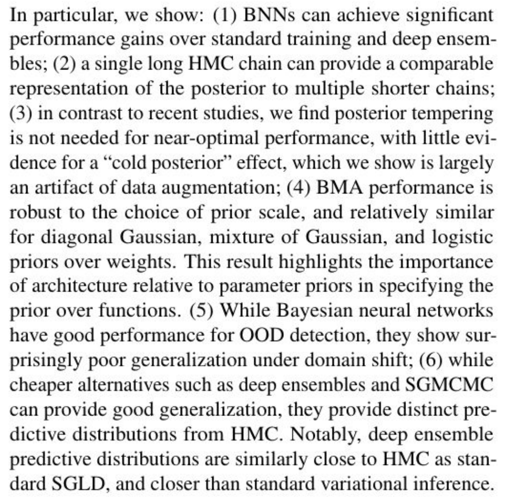


What Are Bayesian Neural Network Posteriors Really Like?

https://arxiv.org/pdf/2104.14421.pdf

NeurIPS challenge

https://proceedings.mlr.press/v176/wilson22a/wilson22a.pdf



In particular, we show: (1) BNNs can achieve significant performance gains over standard training and deep ensembles; (2) a single long HMC chain can provide a comparable representation of the posterior to multiple shorter chains; (3) in contrast to recent studies, we find posterior tempering is not needed for near-optimal performance, with little evidence for a "cold posterior" effect, which we show is largely an artifact of data augmentation; (4) BMA performance is robust to the choice of prior scale, and relatively similar for diagonal Gaussian, mixture of Gaussian, and logistic priors over weights. This result highlights the importance of architecture relative to parameter priors in specifying the prior over functions. (5) While Bayesian neural networks have good performance for OOD detection, they show surprisingly poor generalization under domain shift; (6) while cheaper alternatives such as deep ensembles and SGMCMC can provide good generalization, they provide distinct predictive distributions from HMC. Notably, deep ensemble predictive distributions are similarly close to HMC as standard SGLD, and closer than standard variational inference.

In this section we discuss the important takeaways and observations we made from the results of the competition.

Most notably, all the teams that finished in the top-5 used some form of ensembling to construct a multimodal approximation to the posterior. Despite the fact that in this competition we measure the fidelity of approximate inference and not the generalization performance, multiple participants found ensembling to lead to significant improvements. Furthermore, team `moellenh` reported that standard deep ensembles provide a very strong baseline for matching the HMC predictive distributions. The same observation has been made in Wilson and Izmailov (2020) and Izmailov et al. (2021b) and discussed in detail in the blogpost *Deep Ensembles as Approximate Bayesian Inference* (`https://cims.nyu.edu/~andrewgw/deepensembles/`). These results highlight that we should stop treating standard deep ensembles as a "non-Bayesian" alternative to procedures such as unimodal variational inference. Indeed, multi-modal approximations to the posterior should become a new standard in Bayesian deep learning, and the multi-modality may even be more important than the quality of approximation within each of the modes.

Another observation shared by multiple participants is that the quality of the posterior approximation achieved by their methods is highly dependent on the hyper-parameters. We believe that high-quality HMC references that we developed for this competition can be used to tune Bayesian deep learning methods to achieve high quality approximate inference in practice.

Infinite width limits of Bayesian neural networks are sometimes also used as a proxy for "ground truth" inference, because under certain conditions in regression these limits converge to a closed-form Gaussian process predictive distribution (e.g., Foong et al., 2020; He et al., 2020). However, these limits give rise to a different model than the parametric Bayesian neural network analogues, thus making it unclear how closely we would expect high quality inference within the parametric Bayesian neural network to match the exact inference in the infinite limit network. On the UCI benchmark (the only benchmark here that permits closed form inference for the infinite limit) we found that the infinite limit ranked last in its similarity to HMC. Comparing these metrics further is an interesting direction for future work.

Finally, we found that all the submitted methods struggled to match the predictions of HMC on corrupted data: on the corrupted CIFAR-10-C data the best agreement achieved by any team was 78.7% compared to 91.6% on the clean CIFAR-10 test set. We believe that there is still a large room for improvement for approximate inference methods and our competition provides a unique benchmark to measure the progress in this area.

In conclusion, we believe the competition will provide a foundation for innovation and continued benchmarking of approximate Bayesian inference procedures in deep learning. We received over 300 submissions from 12 active teams, each developing a unique approach to the competition. The top 3 submissions all developed novel ideas providing state-of-the-art