

# Bayesian Deep Learning

---

# Deep ensemble (Lakshminarayanan, Pritzel, and Blundell, 2017)

- The simplest yet powerful method.
- Train the same model  $M$  times with different random seeds and average them.

$$p(y|x) \approx \frac{1}{M} \sum_{m=1}^m p(y|x; \theta_m). \quad (1)$$

- $M = 5 \sim 10$  is sufficient for a decent performance.
- Requires  $M$  times training and  $M$  times parameters.
- Even better than most of the recent techniques.

# Deep ensemble (Lakshminarayanan, Pritzel, and Blundell, 2017)

Deep ensemble can explore diverse modes in a function space.

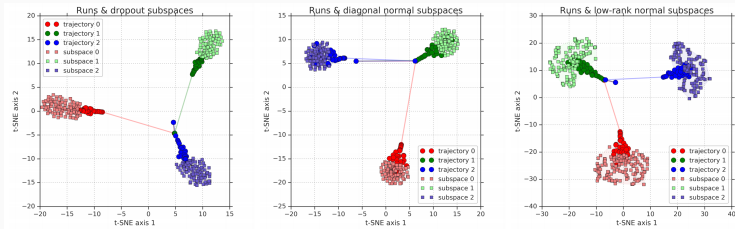


Figure 1: Function space visualizations for Deep ensembles (Fort, Hu, and Lakshminarayanan, 2019).

## More expressive approximations

Multiplicative normalizing flows ([Louizos and Welling, 2016](#)) for variational distributions.

$$q(W; \phi) = \int q(W|z)q(z; \phi)dz, \quad (2)$$

where

$$q(W|z) = \prod_{i=1}^r \prod_{j=1}^c \mathcal{N}(w_{ij}|z_i\mu_{ij}, \sigma_{ij}^2). \quad (3)$$

Here,  $q(z; \phi)$  is a normalizing flow ([Rezende and Mohamed, 2015](#)) defined as

$$z_0 \sim q(z_0), \quad z = f_K \circ \dots \circ f_1(z_0). \quad (4)$$

# Stochastic gradient descent and Bayesian inference

- A Stochastic Gradient Descent (SGD) with a constant learning rate, under some conditions, defines a Markov chain targetting the true posterior distribution, and thus can be understood as an approximate Bayesian inference scheme ([Mandt, Hoffman, and Blei, 2017](#)).
- Based on this observation, [Izmailov et al. \(2016\)](#) proposed Stochastic Weight Averaging (SWA), where a model is trained with cyclic learning rates and multiple snapshots of parameters are used to average results.

# Stochastic gradient descent and Bayesian inference

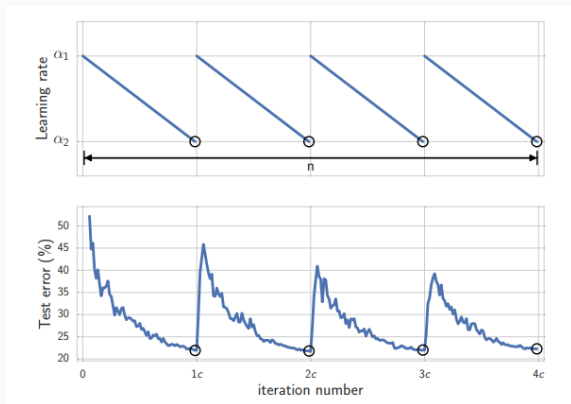


Figure 2: Cyclic learning rates and corresponding test errors (Izmailov et al., 2016).

---

**Algorithm 1** Stochastic Weight Averaging

---

**Require:**

weights  $\hat{w}$ , LR bounds  $\alpha_1, \alpha_2$ ,  
cycle length  $c$  (for constant learning rate  $c = 1$ ), number of iterations  $n$

**Ensure:**  $w_{\text{SWA}}$ 

$w \leftarrow \hat{w}$  {Initialize weights with  $\hat{w}$ }

$w_{\text{SWA}} \leftarrow w$

**for**  $i \leftarrow 1, 2, \dots, n$  **do**

$\alpha \leftarrow \alpha(i)$  {Calculate LR for the iteration}

$w \leftarrow w - \alpha \nabla \mathcal{L}_i(w)$  {Stochastic gradient update}

**if**  $\text{mod}(i, c) = 0$  **then**

$n_{\text{models}} \leftarrow i/c$  {Number of models}

$w_{\text{SWA}} \leftarrow \frac{w_{\text{SWA}} \cdot n_{\text{models}} + w}{n_{\text{models}} + 1}$  {Update average}

**end if**

**end for**

{Compute BatchNorm statistics for  $w_{\text{SWA}}$  weights}

---

# Stochastic gradient descent and Bayesian inference

SWA-Gaussian (SWAG) ([Maddox et al., 2019](#)), an improved version with Gaussian approximation.

Having computed  $\theta_{\text{SWA}} = \frac{1}{T} \sum_{i=1}^T \theta_i$ ,

- SWAG-diagonal: compute empirical diagonal covariance as

$$\Sigma_{\text{diag}} = \text{diag}(\overline{\theta^2} - \theta_{\text{SWA}}^2) \text{ where } \overline{\theta^2} = \frac{1}{T} \sum_{i=1}^T \theta_i^2. \quad (5)$$

- SWAG: add additional low-rank covariances.

$$\Sigma = \frac{1}{2} \Sigma_{\text{diag}} + \frac{1}{2(K-1)} \sum_{i=1}^K (\theta_i - \theta_{\text{SWA}})(\theta_i - \theta_{\text{SWA}})^\top, \quad (6)$$

with  $K < T$ .

# Stochastic gradient descent and Bayesian inference

Then the prediction is done as

$$\begin{aligned}\theta^{(1)}, \dots, \theta^{(S)} &\stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(\theta_{\text{SWA}}, \Sigma), \\ p(y_* | x_*) &\approx \frac{1}{S} \sum_{s=1}^S p(y_* | x_*, \theta^{(s)}).\end{aligned}\tag{7}$$

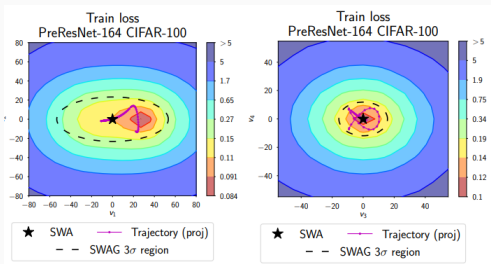


Figure 3: Weight space visualization of SWAG (Maddox et al., 2019).

## Less expensive ensembles

BatchEnsemble (Wen, Tran, and J., 2020): perturb a weight matrix  $W \in \mathbb{R}^{a \times b}$

$$\overline{W}_m = W \odot r_m s_m^\top, \quad m = 1, \dots, M, \quad (8)$$

where  $r_m \in \mathbb{R}^a$  and  $s_m \in \mathbb{R}^b$ . That is, instead of training the entire weights  $M$  times, only train the rank-1 perturbations  $r_m s_m^\top$   $M$  times.

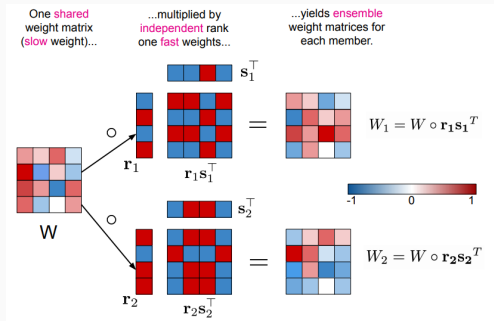


Figure 4: Diagram for BatchEnsemble (Wen, Tran, and J., 2020).

Training BatchEnsemble: unlike the typical ensemble trains a model  $M$  times (either sequentially or parallelly), BatchEnsemble divides a mini-batch into  $M$  sub-batches so that each mini-batch is processed with  $\overline{W}_m = W \odot r_m s_m^\top$ .

Rank-1 Bayesian Neural Network (BNN) ([Dusenberry et al., 2020](#)): Bayesian version of BatchEnsemble. Assume prior distributions on  $r$  and  $s$ , and optimize the Evidence Lower BOund (ELBO),

$$\begin{aligned}\mathcal{L} = & \sum_{i=1}^n \mathbb{E}_q[\log p(y_i|x_i, W, r, s)] \\ & - D_{\text{KL}}[q(r)||p(r)] - D_{\text{KL}}[q(s)||p(s)] + \log p(W).\end{aligned}$$

- The weights of deep neural networks have no physical meaning, it is really hard to posit a meaningful prior distribution.
- The posteriors of the weights are extremely high-dimensional and multi-modal so even with modern methods it is hard to approximate them.
- The relationship between the weights and function values are complicated, so it is not clear if we would get desired function value behavior.
- After all, we are interested in finding a function. Why don't we place a prior distribution on function directly?

## Definition 1 (Implicit processes [\(Ma, Li, and Hernández-Lobato, 2019\)](#))

An implicit process is a stochastic process  $f(x)$  where any finite collection  $(f(x_1), \dots, f(x_n))$  has a joint distribution implicitly defined as

$$z \sim p(z), \quad f(x_i) = g_\theta(x_i, z) \text{ for } i = 1, \dots, n. \quad (9)$$

We don't know the distribution yet we can generate samples.

Instead of assuming a prior distribution on the parameters of neural networks, assume a prior stochastic process on the neural network outputs, and find an approximate posterior stochastic process of functions.

- Variational implicit processes ([Ma, Li, and Hernández-Lobato, 2019](#)): priors as an implicit process with learnable parameters, posteriors as Gaussian processes.
- Functional variational Bayesian neural networks ([Sun et al., 2019](#)): priors as Gaussian processes (or implicit processes), posteriors as implicit processes.

Both of them require non-trivial methods to optimize ELBOs.

# Priors on functions

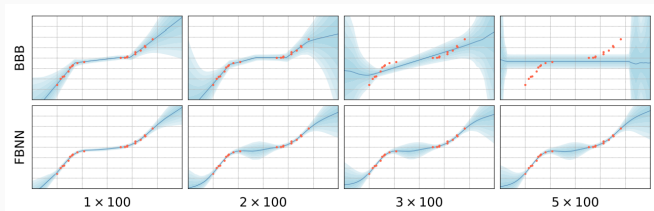


Figure 5: Bayes-by-backprop vs. Functional Bayesian neural net ([Dusenberry et al., 2020](#)).

# Neural processes [\(Garnelo et al., 2018\)](#)

- Meta-learning version of implicit processes.
- Neural-network analogy of Gaussian processes.
- A data-driven way of learning stochastic processes.

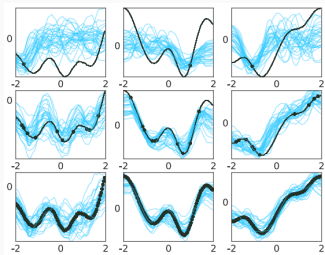


Figure 6: Neural process regressions [\(Garnelo et al., 2018\)](#).

Gaussian process regression: given a set  $(X, Y)$  and kernel  $K$ , the predictive distribution is computed as

$$p(y_*|x_*, X, Y) = \mathcal{N}(y_* | K_{*X}(K_{XX} + \sigma_y^2)^{-1}Y, \\ \sigma_y^2 + K_{**} - K_{*X}(K_{XX} + \sigma_y^2)^{-1}K_{X*}).$$

Neural Processes (NPs): put  $(X, Y)$  into a neural network to predict

$$\begin{aligned} \mu_z, \sigma_z^2 &= f_{\text{enc}}(X, Y), \quad z \sim \mathcal{N}(z | \mu_z, \sigma_z^2) \\ \mu_*, \sigma_*^2 &= f_{\text{dec}}(z, x_*), \quad p(y_*|x_*, X, Y) = \mathcal{N}(y_* | \mu_*, \sigma_*^2). \end{aligned} \tag{10}$$

The network  $f_{\text{enc}}(X, Y)$  takes a **set**  $(X, Y)$  as input!

$$f_{\text{enc}}(X, Y) = \frac{1}{n} \sum_{i=1}^n g_{\text{enc}}(x_i, y_i). \quad (11)$$

Unlike Gaussian Process (GP) or other stochastic processes, a NP assumes nothing but the neural network structure of  $(f_{\text{enc}}, f_{\text{dec}})$ , and learns proper uncertainties from **data**. Hence, training requires many examples (meta-training sets).

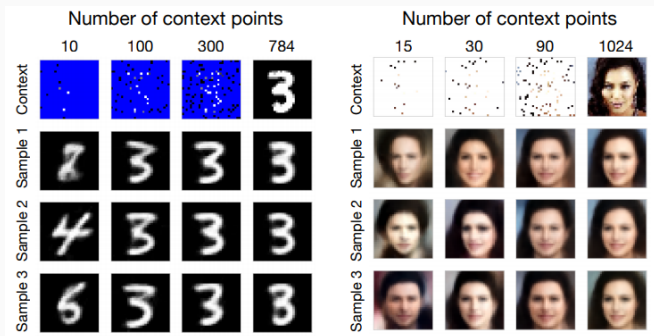
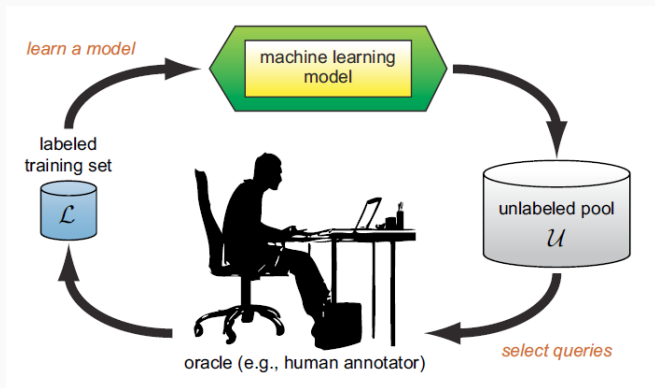


Figure 7: Image completion with NP (Garnelo et al., 2018).

# Deep Bayesian active learning



Active learning pipeline:

1. An initial labeled dataset  $\mathcal{L} = \{x_i, y_i\}_{i=1}^n$  and an unlabeled pool  $\mathcal{U} = \{x_j\}_{j=1}^m$  are given.
2. Optimize an initial model  $f(x)$  (e.g., deep neural networks) with  $\mathcal{L}$ .
3. Sort the samples in  $\mathcal{U}$  with **some criterion**  $a(x, f)$ .
4. Query the top  $K$  samples to human annotators and put them into .
5. Retrain the model and repeat the acquisition process.

Which samples should be queried? - the most uncertain samples are most informative. Focusing on the  $k$ -way classification problem where  $y \in \{1, \dots, k\}$ , here are some examples of the acquisition criterion  $a(x, f)$ .

- Predictive variance:

$$\text{Var}(y|x, \mathcal{L}) = \mathbb{E}_{p(\theta|\mathcal{L})}[\text{Var}(y|x, \theta)] + \text{Var}_{p(\theta|\mathcal{L})}(\mathbb{E}_{p(y|x, \theta)}[y|x, \theta]). \quad (12)$$

- Variation ratio ([Freeman, 1965](#)): given  $S$  models (e.g.,  $f(\cdot; \theta^{(1)}), \dots, f(\cdot; \theta^{(1)})$  with  $\theta^{(1)}, \dots, \theta^{(S)} \stackrel{\text{i.i.d.}}{\sim} p(\theta|\mathcal{L})$ , the portion of the models that do not agree with the majority vote. For instance, if 5 models predict classes as  $[1, 2, 1, 1, 4]$ , the majority vote would be the class 1 and the variation ratio is  $2/5$ .

- Maximum entropy: measures the predictive entropy:

$$H[y|x, \mathcal{L}] = - \sum_{j=1}^k p(y = j|x, \mathcal{L}) \log p(y = j|x, \mathcal{L}). \quad (13)$$

- Bayesian Active Learning by Disagreement (BALD) ([Houlsby et al., 2011](#)): measures the mutual information between the parameter  $\theta$  and the label  $y$  corresponding to an input  $x$ :

$$I[y, \theta|x, \mathcal{L}] = H[y|x, \mathcal{L}] - \mathbb{E}_{p(\theta|\mathcal{L})}[H[y|x, \theta]]. \quad (14)$$

- How to estimate the uncertainty (especially the model uncertainty)? - Bayesian deep learning
- MC dropout or variational inference are suboptimal, as they underestimate the posterior variances.
- Deep ensemble might be a good solution but requires heavy computation.
- Snapshot ensemble (deep ensemble from a single training run with cyclical learning rate) is more effective for active learning ([Jung, Kim, and Lee, 2023](#)).

- Consider the following experiments aimed at studying the working memory of individuals, with a focus on determining the capacity for remembering digits.
- How many digits do you need to present to the participants to obtain the best results?
- If you show too few numbers, participants may easily succeed, and if you show too many, they may mostly fail, thereby limiting the informative value of the experiments.

# Bayesian experimental design

In Bayesian Experimental Design (BED), you have a model  $p(y|\theta, \xi)$  where  $y$  is an outcome,  $\xi$  is the experimental design, and  $\theta$  is the parameter you want to infer. In our previous example,

- $y \in \{0, 1\}$  is the experiment outcome, indicating whether a participant succeeded in remembering a sequence of digits.
- $\xi$  is the experimental design, the number of digits you show to a participant.
- We have our internal model  $p(y|\theta, \xi)$  of the working memory, for instance,

$$p(y = 1|\theta, \xi) = \text{Ber}(y = 1|\text{sigmoid}(\theta - \xi)). \quad (15)$$

Given a prior  $p(\theta)$ , our goal is to infer the posterior  $p(\theta|y, \xi)$ .

The BED pipeline:

1. Choose a design  $\xi \in \Xi$ .
2. Perform experiment with  $\xi$  to obtain an outcome  $y$ .
3. Compute the posterior  $p(\theta|\xi, y)$ .
4. Compute the expected utility  $\mathbb{E}_{p(\theta|\xi, y)}[U(\theta, \xi, y)]$ .














A commonly used utility function: Information Gain (IG).

$$\begin{aligned}\text{IG}(\xi, y) &= H[p(\theta)] - H[p(\theta|\xi, y)] \\ &= \mathbb{E}_{p(\theta|\xi, y)}[\log p(\theta|\xi, y)] - \mathbb{E}_{p(\theta)}[\log p(\theta)].\end{aligned}\tag{16}$$

The Expected Information Gain (EIG) is then computed as,

$$\begin{aligned}\text{EIG}(\xi) &:= \mathbb{E}_{p(y|\xi)}[\text{IG}(\xi, y)] \\ &= \mathbb{E}_{p(\theta)p(y|\xi, \theta)}[\log p(\theta|\xi, y) - \log p(\theta)] \\ &= \mathbb{E}_{p(\theta)p(y|\xi, \theta)}[\log p(y|\theta, \xi) - \log p(y|\xi)] \\ &= \mathbb{E}_{p(\theta)}[H[p(y|\xi)] - H[p(y|\xi, \theta)]] .\end{aligned}\tag{17}$$

# References i

-  Dusenberry, M. W. et al. (2020). "Efficient and scalable Bayesian neural nets with rank-1 factors". In: *Proceedings of International Conference on Machine Learning*.
-  Fort, S., H. Hu, and B. Lakshminarayanan (2019). "Deep ensembles: a loss landscape perspective". In: *arXiv:1912.02757*.
-  Freeman, L. C. (1965). *Elementary applied statistics: for students in behavioral science*. John Wiley and Sons.
-  Garnelo, M. et al. (2018). "Neural processes". In: *ICML Workshop on Theoretical Foundations and Applications of Deep Generative Models*.
-  Houlisby, N. et al. (2011). "Bayesian active learning for classification and preference learning". In: *arXiv preprint arXiv:1112.5745*.
-  Izmailov, P. et al. (2016). "Averaging weights leads to wider optima and better generalization". In: *Proceedings of Uncertainty in Artificial Intelligence*.
-  Jung, S., S. Kim, and J. Lee (2023). "A simple yet powerful deep active learning with snapshot ensembles". In: *Proceedings of International Conference on Learning Representations*.
-  Lakshminarayanan, B., A. Pritzel, and C. Blundell (2017). "Simple and scalable predictive uncertainty estimation using deep ensembles". In: *Advances in Neural Information Processing Systems*.
-  Louizos, C. and M. Welling (2016). "Multiplicative normalizing flows for variational Bayesian neural networks". In: *Proceedings of International Conference on Machine Learning*.
-  Ma, C., Y. Li, and J. M. Hernández-Lobato (2019). "Variational implicit processes". In: *Proceedings of International Conference on Machine Learning*.
-  Maddox, W. J. et al. (2019). "A simple baseline for Bayesian uncertainty in deep learning". In: *Advances in Neural Information Processing Systems*.
-  Mandt, S., M. D. Hoffman, and D. M. Blei (2017). "Stochastic gradient descent as approximate Bayesian inference". In: *Journal of Machine Learning Research* 18.1.
-  Rezende, D. J. and S. Mohamed (2015). "Variational inference with normalizing flows". In: *Proceedings of International Conference on Machine Learning*.



Sun, S. et al. (2019). "Fucntional variational Bayesian neural networks". In: *Proceedings of International Conference on Learning Representations*.



Wen, Y., D. Tran, and Ba. J. (2020). "BatchEnsemble: an alternative approach to efficient ensemble and lifelong learning". In: *Proceedings of International Conference on Learning Representations*.