Recepies for Bayesian Deep Learning

Dimitrios Milios

Seminar for The Department of Statistics

Athens University of Economics and Business

24th of March 2025

Recepies for Bayesian Deep Learning

$$\underbrace{p(\mathbf{w} \mid \mathcal{D})}_{\text{posterior}} = \frac{\overbrace{p(\mathcal{D} \mid \mathbf{w})}^{\text{likelihood}}\,\overbrace{p(\mathbf{w})}^{\text{prior}}}{\underbrace{\int_{\mathbf{w}} p(\mathcal{D} \mid \mathbf{w}) p(\mathbf{w}) d\mathbf{w}}_{\text{marginal likelihood}}}$$

- G. Franzese, D. Milios, M. Filippone, P. Michiardi: *Revisiting the Effects of Stochasticity for Hamiltonian Samplers*. ICML 2022: 6744-6778
- B. Tran, S. Rossi, D. Milios, M. Filippone: *All You Need is a Good Functional Prior for Bayesian Deep Learning*. J. Mach. Learn. Res. 23: 74:1-74:56 (2022)
- B. Tran, S. Rossi, D. Milios, P. Michiardi, E.V. Bonilla, M. Filippone: *Model Selection for Bayesian Autoencoders*. NeurIPS 2021: 1973019742

Supervised learning

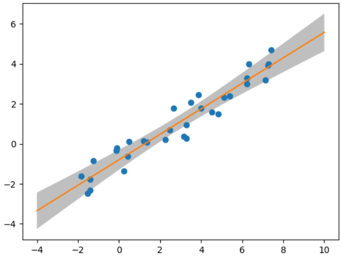
We give a machine some examples . . .

- Input space: $x \in \mathcal{X}$ (i.e. $x \in \mathcal{R}^d$)
- Output space: $y \in \mathcal{Y}$ (i.e. $y \in \mathcal{R}$)

We fit a function:

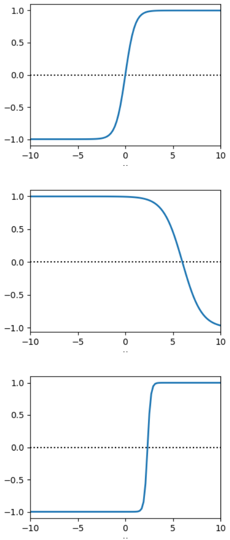
$$f : \mathcal{X} \to \mathcal{Y}$$

Use $f(x)$ to: *predict*, *extrapolate*, *interpolate*, . . .



(Non-)linear models

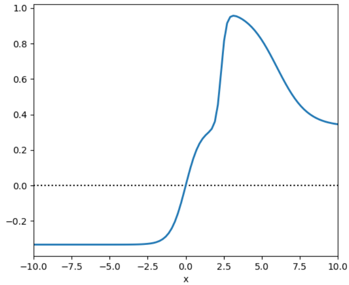
Linear Combinations of Feature Functions



w_1

w_2

w_3

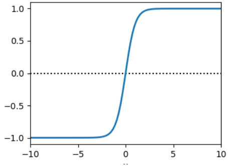


Which feature functions?

- Kernel Methods
- Neural Networks

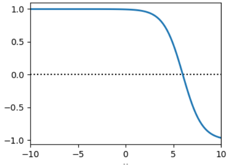
Neural Networks

$w_{1,1}$



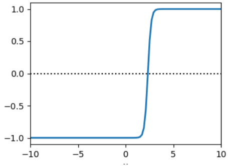
w_1

$w_{1,2}$



w_2

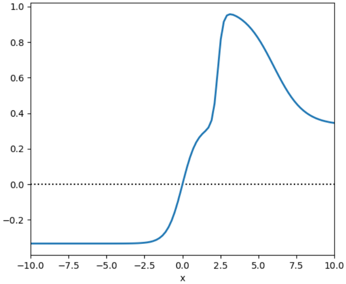
$w_{1,3}$



w_3

Parametrise $\phi(x)$:

$$f(\mathbf{x}) = \mathbf{w}_2^\top \phi(\mathbf{w}_1^\top \mathbf{x})$$



More layers:

$$f(\mathbf{x}) = \mathbf{w}_3^\top \phi(\mathbf{w}_2^\top \phi(\mathbf{w}_1^\top \mathbf{x}))$$

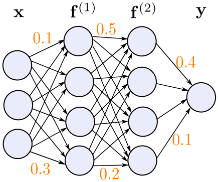
Neural Networks

- For input datapoint $\mathbf{x}_n$ with output y_n, we define log-likelihood function:

$$\log p(y_n|\mathbf{x}_n, \mathbf{w}) = -\mathrm{loss}(f(\mathbf{x}_n; \mathbf{w}), y_n)$$

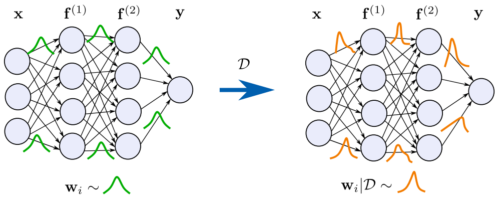
- Maximum likelihood estimation given a data set $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$:

$$\mathbf{w}^* = \mathrm{argmax}_{\mathbf{w}} \sum_{n=1}^{N} \log p(y_n|\mathbf{x}_n, \mathbf{w})$$



- Backpropagation: $\nabla_w \sum_{n=1}^{N} \log p(y_n|\mathbf{x}_n, \mathbf{w})$
- Mini-batches: Random subsets $\rightarrow$ *Unbiased* gradient estimates
- Point estimate $\rightarrow$ No *uncertainty quantification*.

Bayesian Neural Networks



Place a prior distribution $p(\mathbf{w})$ over the network's parameters $\mathbf{w}$.

- Compute posterior given a data set $\mathcal{D}$:

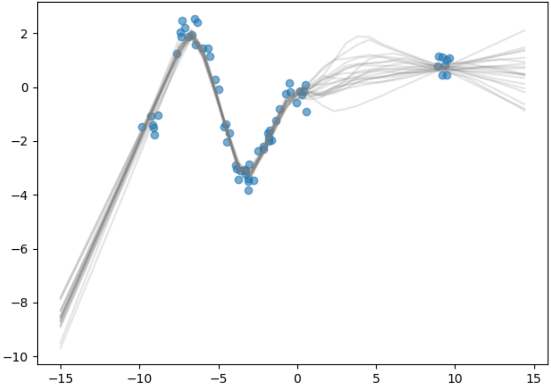
$$\underbrace{p(\mathbf{w} \mid \mathcal{D})}_{\text{posterior}} \propto \underbrace{p(\mathcal{D} \mid \mathbf{w})}_{\text{likelihood}} \underbrace{p(\mathbf{w})}_{\text{prior}}$$

- Posterior predictive:

$$p(y^* \mid \mathbf{x}^*, \mathcal{D}) = \mathbb{E}_{p(\mathbf{w}|\mathcal{D})}[p(y^* \mid \mathbf{x}^*, \mathbf{w})]$$

Samples from the Posterior Distribution

Bayesian Neural Networks



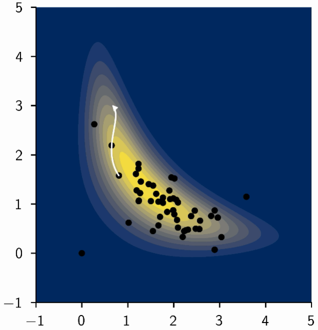
Bayesian Posterior for Neural Networks

Approximating Methods

- Assume a tractable distribution for posterior
  - Variational methods: Optimize measures related to KL-divergence
  - Laplace Approximation: Gaussian Approximation based on Hessian
- Do not capture the full extend of the posterior

Sampling Methods

- Hamiltonian Monte Carlo
- Supports arbitrary distributions
- Non-trivial treatment of stochastic gradients



Stochastic Hamiltonian Monte Carlo

Consider the SDE:

$$d\theta_t = \mathbf{r}_t dt$$
$$d\mathbf{r}_t = -\nabla_\theta U(\theta_t)dt - C\mathbf{r}_t dt + \sqrt{2C}dW_t$$

Potential energy:

$$U(\theta) = -\sum_{i=1}^{N} \log p(y_i|\theta) - \log p(\theta)$$

Friction term C

Wiener Process (a.k.a. **Brownian motion**):

- For the differential we have: $dW_t \sim \mathcal{N}(0, dt)$
- More formally: $W_t : \quad W_{t+s} - W_t \sim \mathcal{N}(0, s)$

Simulating the Hamiltonian system

Theorem

For an ergodic stochastic process described by the SDE above with stationary distribution $\rho_{ss}(\theta, \mathbf{r})$ we have:

$$\rho_{ss}(\theta, \mathbf{r}) \propto \exp(-U(\theta) - 1/2||\mathbf{r}||^2)$$

For instance: Simulate with *leapfrog integrator*:

$$\begin{cases} \theta^* = \theta_{i-1} + \frac{\eta}{2} r_{i-1} \\ r_i = r_{i-1} - \eta \nabla U(\theta^*) - \eta C r_{i-1} + \sqrt{2C\eta} w, \quad w \sim \mathcal{N}(0, I) \\ \theta_i = \theta^* + \frac{\eta}{2} r_i \end{cases}$$

Two sources of error:

- Time-discretization error
- Stochastic gradient error (i.e. due to mini-batches)

Mini-batches as Operator Splitting

Markov kernel at time t: $\exp(t\mathcal{L}^\dagger)$

Infinitesimal generator operator:

$$\mathcal{L} = \underbrace{-\left(\nabla_\theta^\top U(\theta)\right)\nabla_{\mathbf{r}} + \mathbf{r}^\top\nabla_\theta}_{\text{pure Hamiltonian evolution}} \underbrace{-C\mathbf{r}^\top\nabla_{\mathbf{r}} + C\nabla_{\mathbf{r}}^\top\nabla_{\mathbf{r}}}_{\text{friction and Noise}}$$

- Dataset D split into mini-batches $\mathrm{D}_1, \ldots, \mathrm{D}_K$
- Infinitesimal generators $\mathcal{L}_1, \ldots, \mathcal{L}_K$ such that:

$$\mathcal{L} = \mathcal{L}_1 + \cdots + \mathcal{L}_K$$

Hopefully, we have:

$$\exp(\eta\mathcal{L}_1)\ldots\exp(\eta\mathcal{L}_K) \simeq \exp(\eta(\mathcal{L}_1 + \cdots + \mathcal{L}_K))$$

Baker–Campbell–Hausdorff formula

$$\exp(\mathcal{A})\exp(\mathcal{B}) = \exp\left(\mathcal{A} + \mathcal{B} + \frac{1}{2}[\mathcal{A}, \mathcal{B}] + \frac{1}{12}\left([\mathcal{A}, [\mathcal{A}, \mathcal{B}]] + [\mathcal{B}, [\mathcal{B}, \mathcal{A}]]\right) + \ldots\right)$$

Mini-batches: Weak Order of Convergence

Theorem (Main result)

For mini-batches, the ergodic error has expansion:

$$e(\psi, \phi) = \mathcal{O}\left(\eta^{\min(p,2)}\right)$$

where, p is the order of the numerical integrator ψ

Ergodic average error:

$$e(\psi, \phi) = \int \phi(\theta, r)\rho_{ss}^{\psi}(\theta, r)d\theta dr - \int \phi(\theta, r)\rho_{ss}(\theta, r)d\theta dr.$$

Weak order p implies $e(\psi, \phi) = \mathcal{O}(\eta^p)$

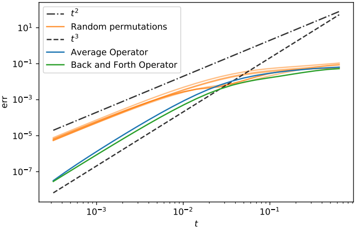
Mini-batches Convergence: Proof Sketch

$$\exp(\mathcal{A})\exp(\mathcal{B}) = \exp\left(\mathcal{A} + \mathcal{B} + \frac{1}{2}[\mathcal{A},\mathcal{B}] + \dots\right)$$

$$\exp(\mathcal{B})\exp(\mathcal{A}) = \exp\left(\mathcal{B} + \mathcal{A} + \frac{1}{2}[\mathcal{B},\mathcal{A}] + \dots\right)$$

For the *commutators* we have:

$$[\mathcal{A},\mathcal{B}] = -[\mathcal{B},\mathcal{A}] \qquad \text{as } [\mathcal{A},\mathcal{B}] = \mathcal{A}\mathcal{B} - \mathcal{B}\mathcal{A}$$

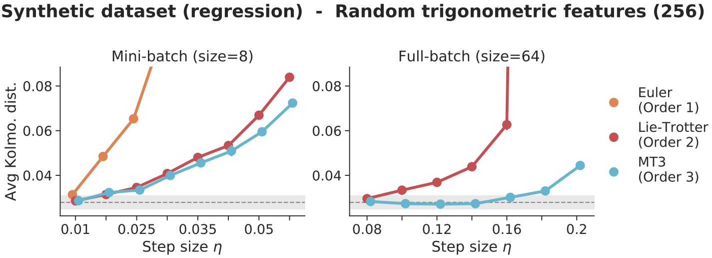


Definition (Debussche and Faou, 2012)

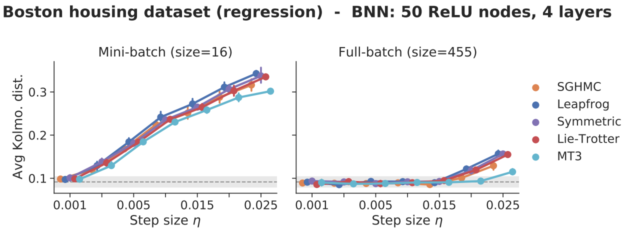
Weak order of convergence p, if for any function ϕ:

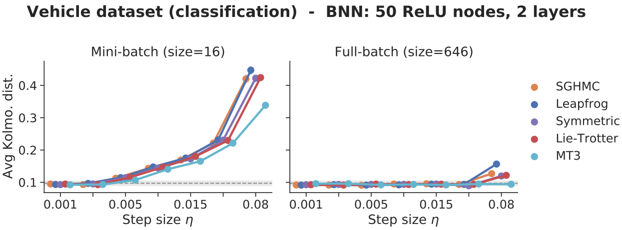
$$\left|\mathbb{E}\left[\phi(\psi(\theta_0, r_0; \eta))\right] - \mathbb{E}\left[\phi(\theta_\eta, r_\eta)\right]\right| = \mathcal{O}(\eta^{p+1})$$

The Convergence Bottleneck



Regression & Classification Experiments





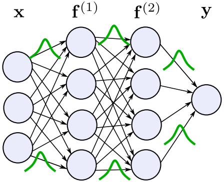
Machine Learning Overview

Sampling with Stochastic Gradient MCMC

Adjusting Neural Network Priors

Bayesian Model Selection for Autoencoders

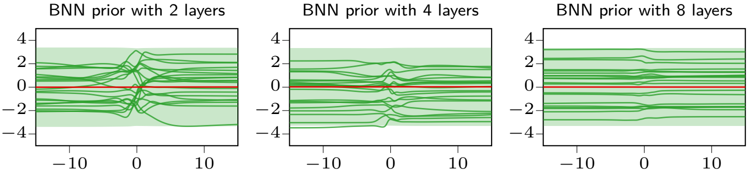
Prior for Bayesian Neural Networks



$\mathbf{w}_i \sim \mathcal{N}(0, \mathbf{I})$

- Neural networks are extremely *high-dimensional* and *unidentifiable*.
  $\longrightarrow$ Reasoning about parameters is very challenging.
- Most work has resorted to priors of convenience.
  $\longrightarrow$ Gaussian priors such as $\mathcal{N}(0, 1)$ and $\mathcal{N}(0, 1/N_{l-1})$ are the most popular priors for BNNs.

Prior for Bayesian Neural Networks



The prior $\mathcal{N}(0, 1)$ is not always problematic, but it can be for deep architectures.

- The sampled functions tend to form straight horizontal lines.
- This is a well-known pathology stemming from increasing model's depth.

Gaussian Process Priors

- Gaussian Processes (GPs) are a useful tool for choosing *sensible priors* on *functions we indent to model*.

- GP is characterized by a mean function $\mu(\cdot)$ and a covariance function $\kappa(\cdot, \cdot)$.

- A function f is distributed to a GP iff for any finite set of inputs $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{N}$, the evaluation of f is jointly Gaussian

$$f(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), \kappa(\mathbf{x}, \mathbf{x}')), \quad \text{then}$$

$$\mathbf{f}_{gp} = (f(\mathbf{x}_1), ..., f(\mathbf{x}_N))^\top \sim \mathcal{N}(\mu, \mathbf{K}),$$

  where $\mu = \mu(\mathbf{X})$ and $\mathbf{K} = \kappa(\mathbf{X}, \mathbf{X})$.

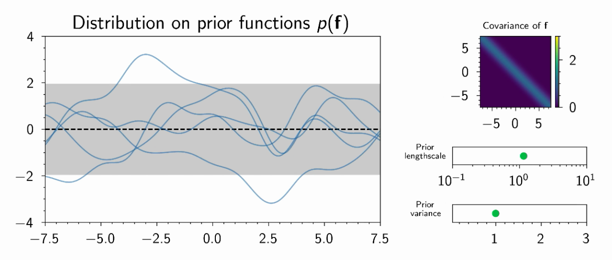
- Neal (1996) showed that BNN converge to GP in the limit of infinite network width.

Gaussian Processes Priors

- A popular covariance function is the RBF:

$$\kappa_{\alpha,l}(\mathbf{x},\mathbf{x}') = \alpha^2 \exp(-\frac{\|\mathbf{x}-\mathbf{x}'\|_2^2}{l^2})$$

- Smoothness of the prior functions: lengthscale l
- Prior marginal standard deviation: amplitude α

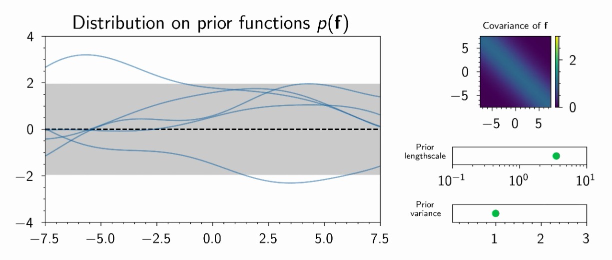


Gaussian Processes Priors

- A popular covariance function is the RBF:

$$\kappa_{\alpha,l}(\mathbf{x},\mathbf{x}') = \alpha^2 \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|_2^2}{l^2}\right)$$

- Smoothness of the prior functions: lengthscale l
- Prior marginal standard deviation: amplitude α

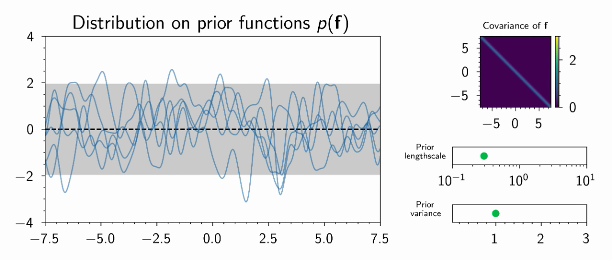


Gaussian Processes Priors

- A popular covariance function is the RBF:

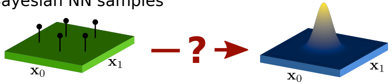
$$\kappa_{\alpha,l}(\mathbf{x},\mathbf{x}') = \alpha^2 \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|_2^2}{l^2}\right)$$

- Smoothness of the prior functions: lengthscale l
- Prior marginal standard deviation: amplitude α



Research Question

How to impose functional priors on BNNs exhibit interpretable properties, similar to GPs?



- We aim at matching two stochastic processes $\rightarrow$ infinite-dimensional distributions.
- We don't know closed-form of the density of BNNs.
  - Minimize the KL divergence between BNN and GP priors.

$$\mathrm{KL} p_{nn} p_{gp} = -\int p_{nn}(\mathbf{f};\psi) \log p_{gp}(\mathbf{f}) d\mathbf{f} + \underbrace{\int p_{nn}(\mathbf{f};\psi) \log p_{nn}(\mathbf{f};\psi) d\mathbf{f}}_{\text{Entropy (intractable)}} .$$

Wasserstein distance

Definition

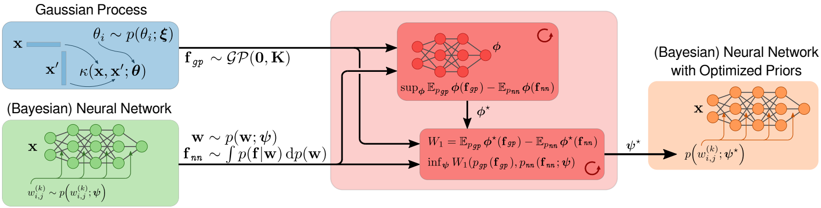
Given a measurable space Ω, the Kantorovich dual form of the 1-Wasserstein distance between two Borel's probability measures π and ν in $\mathcal{P}(\Omega)$ is

$$W_1(\pi, \nu) = \sup_{\|\phi\|_L \leq 1} \mathrm{E}_{\pi}[\phi(\mathbf{x})] - \mathrm{E}_{\nu}[\phi(\mathbf{x})]\,,$$

where ϕ is a 1-Lipschitz function.

- No need to know the closed-form of π and ν as we can estimate expectations with samples.
- The 1-Lipschitz function ϕ can be parameterized by a NN.

Proposed Method



Minimize the 1-Wasserstein distance between samples of two stochastic processes $p_{gp}(\mathbf{f} \mid \mathbf{0}, \mathbf{K})$ and $p_{nn}(\mathbf{f}; \psi)$ at a finite number of measurement points $\mathbf{X}_\mathcal{M}$ sampled from a distribution $q(\mathbf{x})$.

$$\min_{\psi} W_1(p_{gp}, p_{nn}) = \min_{\psi} \max_{\theta} \mathbb{E}_q \left[\underbrace{\mathbb{E}_{p_{gp}}[\phi_\theta(\mathbf{f}_\mathcal{M})] - \mathbb{E}_{p_{nn}}[\phi_\theta(\mathbf{f}_\mathcal{M})]}_{\mathcal{L}(\psi,\theta)} \right],$$

where ϕ_θ is the 1-Lipschitz function parameterized by a neural network with parameters θ.

Choice of the Measurement Set

We consider finite measurement sets to have a practical and well-defined optimization strategy.

- For low-dimensional problems

  $\longrightarrow$ Can use a regular grid or apply uniform sampling in the input domain.

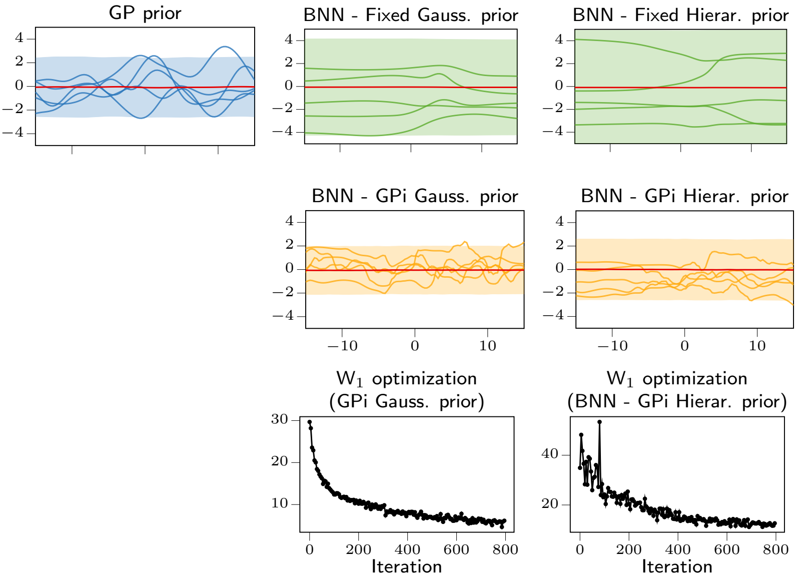
- For high-dimensional problems

  $\longrightarrow$ Can sample from the training set, possibly with augmentation.

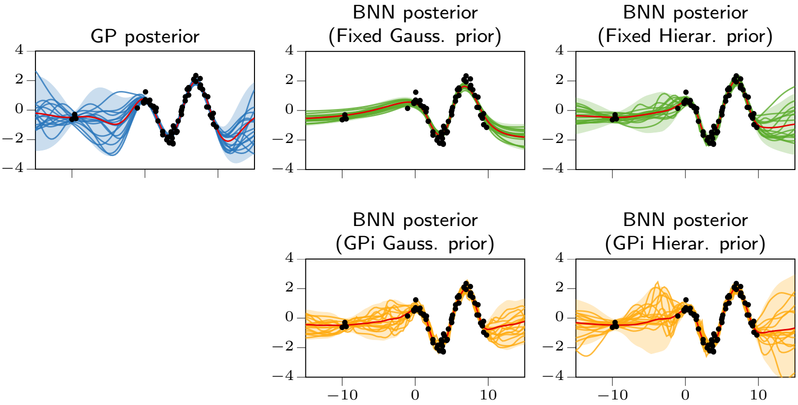
- In applications where we know the input region of the test data points

  $\longrightarrow$ Can set the distribution $q(x)$ to include it.

1D Regression Synthetic Data



1D Regression Synthetic Data

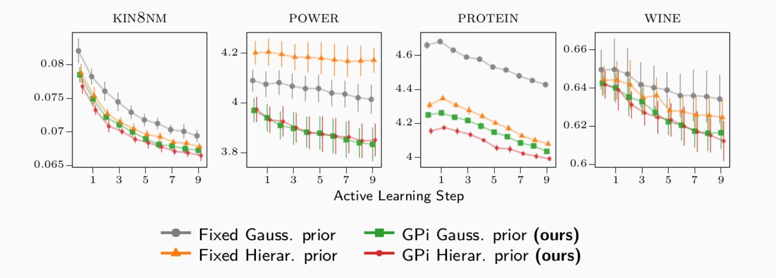


Bayesian Convolutional Neural Networks - CIFAR-10

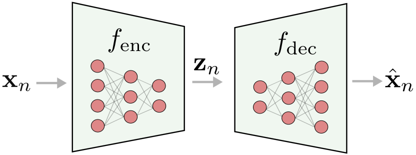
Architecture	Method	Accuracy (↑)	NLL (↓)
LENET5	Deep Ensemble	71.05%	0.8566
	Fixed Gauss. prior	75.01%	0.7410
	Fixed Gauss. prior + Temp. scale.	73.90%	0.7602
	GPi Gauss. prior (**ours**)	75.54%	0.7244
	Fixed hierarchical prior	74.68%	0.7298
	GPi hierarchical prior (**ours**)	**76.86%**	**0.6889**
PRERESNET20	Deep Ensemble	87.8%	0.3908
	Fixed Gauss. prior	85.45%	0.4915
	Fixed Gauss. prior + TS	87.71%	0.3931
	GPi Gauss. prior (**ours**)	86.41%	0.4513
	Fixed hierarchical prior	87.39%	0.4052
	GPi hierarchical prior (**ours**)	**88.31%**	**0.3796**
VGG16	Deep Ensemble	82.23%	0.8685
	Fixed Gauss. prior	81.25%	0.5826
	Fixed Gauss. prior + TS	82.49%	0.5355
	GPi Gauss. prior (**ours**)	82.94%	0.5292
	Fixed hierarchical prior	85.92%	0.4330
	GPi hierarchical prior (**ours**)	**87.11%**	**0.406**

Active Learning

- Each data set is split into 20% train, 60% pool, and 20% test sets.
- At each step, we actively collect *n* data points with the highest posterior entropy from the pool set and add them to the training set.

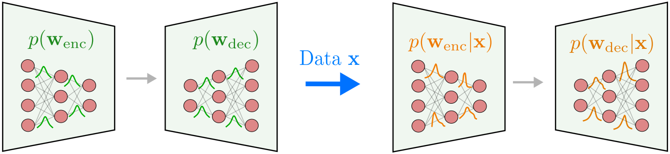


Autoencoders



- An autoencoder (AE) is a neural network used for *unsupervised learning*
- *Encoder*: transforms an unlabelled dataset, $\mathbf{x} := \{\mathbf{x}_n\}_n^N$, into latent codes, $\mathbf{z} := \{\mathbf{z}_n\}_n^N$
- *Decoder*: transforms latent codes into reconstructions, $\hat{\mathbf{x}} := \{\hat{\mathbf{x}}_n\}_n^N$
- Typical AE solution: a point estimate of the network's parameters $\mathbf{w} := \{\mathbf{w}_{\text{enc}}, \mathbf{w}_{\text{dec}}\}$

Bayesian Autoencoders



- A Bayesian neural network for unsupervised learning
- Place a prior $p(\mathbf{w})$ over the network's parameters $\mathbf{w} := \{\mathbf{w}_{\text{enc}}, \mathbf{w}_{\text{dec}}\}$
- The target is exactly the input, $\mathbf{y}_n = \mathbf{x}_n$
- Compute posterior given a dataset $\{\mathbf{x}, \mathbf{y}\}$:

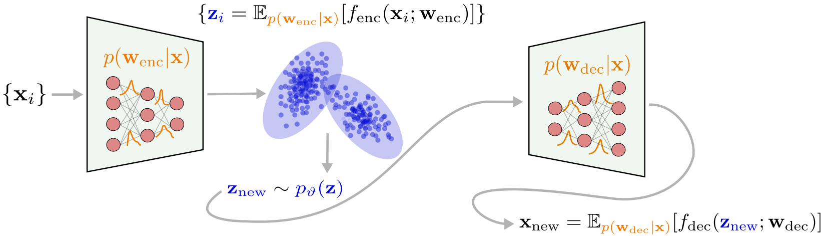
$$\underbrace{p(\mathbf{w} \mid \mathbf{y})}_{\text{posterior}} \propto \underbrace{p(\mathbf{y} \mid f(\mathbf{x}; \mathbf{w}))}_{\text{likelihood}} \underbrace{p(\mathbf{w})}_{\text{prior}}$$

✗ Generative modeling

✗ Difficulty of choosing a sensible prior

Generative Modeling for Bayesian Autoencoders

Use a Dirichlet process mixture model (Blei and Jordan, 2006) for density estimation in latent space



Functional Priors for Bayesian Autoencoders

- Difficulty of choosing a sensible prior
  - Assume a prior distribution, $p_\psi(\mathbf{w})$, on the parameters
  - ψ is the prior hyper-parameters to be chosen
  - This prior induces a non-trivial effect on the output (functional) prior

$$p_\psi(\hat{\mathbf{x}}) = \int f(\mathbf{x}; \mathbf{w}) p_\psi(\mathbf{w}) d\mathbf{w},$$

where $\hat{\mathbf{x}} = f(\mathbf{x}; \mathbf{w})$

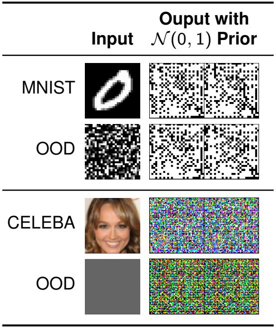


Figure: Realizations sampled from the $\mathcal{N}(0,1)$ prior given an input image. OOD stands for out-of-distribution.

Model Selection for Bayesian Autoencoders

- Difficulty of choosing a sensible prior

$\rightarrow$ Estimating prior hyper-parameters, ψ, based on the *empirical Bayes* approach

- Marginal likelihood

$$p_{\psi}(\mathbf{x}) = \int p(\mathbf{x} \,|\, \hat{\mathbf{x}}) p_{\psi}(\hat{\mathbf{x}}) d\hat{\mathbf{x}},$$

  where $p(\mathbf{x} \,|\, \hat{\mathbf{x}})$ is the likelihood, and $\hat{\mathbf{x}} = f(\mathbf{x}; \mathbf{w})$

- Equivalence between maximum likelihood estimation and KL-divergence minimization

$$\arg\max_{\psi} \int \pi(\mathbf{x}) \log p_{\psi}(\mathbf{x}) d\mathbf{x} = \arg\min_{\psi} \mathrm{KL}[\pi(\mathbf{x}) || p_{\psi}(\mathbf{x})],$$

  where $\pi(\mathbf{x})$ is the data-generating distribution

- Matching these two distributions is non-trivial!

Model Selection for Bayesian Autoencoders

We propose to use the distributional sliced 2-Wasserstein distance (Nguyen et al., 2020)

$$\psi^{\star} = \arg\min_{\psi} \left[DSW_2(p_{\psi}(\mathbf{x}), \pi(\mathbf{x})) \right]$$

- DSW distance addresses two major constraints
  - Computational scalability thanks to using random projection
  - Curse of dimensionality

- The objective is *fully sampled-based* and can be optimized with gradient descent algorithms

$\longrightarrow$ Not necessary to know the closed-form of either $p_{\psi}(\mathbf{x})$ or $\pi(\mathbf{x})$

$\longrightarrow$ Only requirement is that we can draw samples from these two distributions

To sample from $p_{\psi}(\mathbf{x})$
$\rightarrow$ Sample $\mathbf{w}$ from prior $p_{\psi}(\mathbf{w})$
$\rightarrow$ Compute the output $\hat{\mathbf{x}} = f(\mathbf{x}; \mathbf{w})$
$\rightarrow$ Sample from likelihood $p(\mathbf{x} \mid \hat{\mathbf{x}})$

Inductive Bias of the Optimized Priors

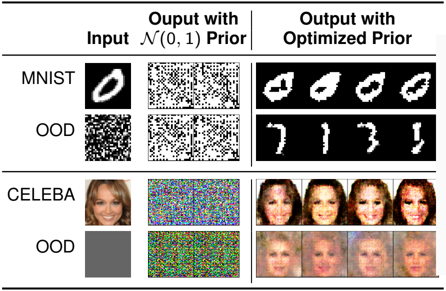


Figure: Realizations sampled from different priors given an input image. OOD stands for out-of-distribution.

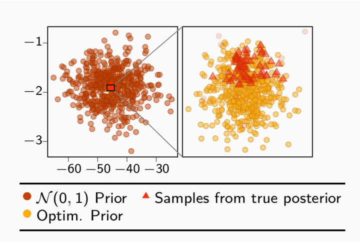
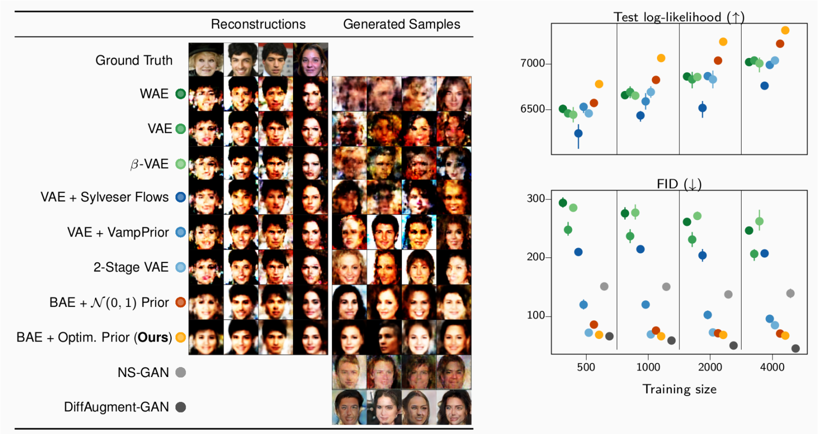


Figure: Visualization in 2D of samples from priors and posteriors of BAE parameters.

The hypothesis space of the optimized prior is reduced to regions close to the true posterior

Experiments on CelebA Dataset



Concluding Remarks

We have challenged traditional recipes for the Bayesian Deep Learning

Inference

- Approximating methods (i.e. variational) are limiting
- Traditional sampling is not scalable
- Sampling via Hamiltonian SDEs has appealing properties
  - admits stochastic gradients
  - enjoys Weak order 2 convergence

Prior

- The effect of priors has been barely studied
- We have induced Gaussian Process priors to BNNs
- Functional priors may significantly improve uncertainty quantification

Marginal likelihood for Bayesian Autoencoders

- We have defined a proxy process for type-II maximum likelihood
- Highly competitive to popular generative models

References

- Abdulle, A., Vilmart, G., and Zygalakis, K. C. High order numerical approximation of the invariant measure of ergodic sdes. SIAM Journal on Numerical Analysis, 52(4): 16001622, 2014.

- Chen, T., Fox, E., and Guestrin, C. Stochastic gradient Hamiltonian Monte Carlo. In International Conference on Machine Learning, volume 32 of Proceedings of Machine Learning Research, pp. 16831691. PMLR, 2014.

- Hoffman, M. D. and Gelman, A. The no-u-turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. Journal of Machine Learning Research, 15(1): 15931623, 2014.

- Debussche, A. and Faou, E. Weak backward error analysis for sdes. SIAM Journal on Numerical Analysis, 50(3): 17351752, 2012.

1ex
- Milstein, G. and Tretyakov, M. V. Quasi-symplectic methods for Langevin-type equations. IMA journal of Numerical Analysis, 23(4):593626, 2003.

References

- R. M. Neal. Bayesian Learning for Neural Networks (Lecture Notes in Statistics). Springer, 1st edition, Aug. 1996.
- D. Duvenaud, O. Rippel, R. Adams, and Z. Ghahramani. Avoiding Pathologies in Very Deep Networks. In Proceedings of the 17th International Conference on Artificial Intelligence and Statistics, volume 33 of Proceedings of Machine Learning Research, pages 202210. PMLR, 2014.
- D. Flam-Shepherd, J. Requeima, and D. Duvenaud. Mapping Gaussian Process Priors to Bayesian Neural Networks. In NeurIPS workshop on Bayesian Deep Learning, 2017.
- B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In Advances in Neural Information Processing Systems, volume 30, pages 64026413. Curran Associates, Inc., 2017.
- D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In International Conference on Learning Representations, 2014.