A Survey of Bayesian Methods for Deep Learning

Hongyu Lu HL1535@NYU.EDU
Samuel Stanton SS13641@NYU.EDU
Jiuhong Xiao JX1190@NYU.EDU
New York University
60 5th Ave
New York, NY 10011

Editor:

Abstract

Bayesian philosophy and statistics provides a formal approach to coherent inference. Historically, Bayesian methods played an important role in the construction of predictive models by allowing experts to incorporate domain knowledge and by specifying how model parameters should be updated in the presence of new observations. The advent of deep learning has posed new challenges for Bayesian statistics. Entities like the posterior on the model parameters or the marginal likelihood are rarely tractable in the context of deep learning. We survey recent works that apply principles of Bayesian inference to deep learning. Additionally, we make note of notable applications of Bayesian deep learning.

1. Introduction

Fundamentally, Bayesian inference answers the question, "Among my set of competing hypotheses, which should I use to inform my action?" The question is, of course, a difficult one. In the simplest setting, a learner has a dataset of input-label pairs $\mathcal{D} := \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ and a hypothesis class $\mathcal{F} := \{f(\cdot;\boldsymbol{\theta})|\boldsymbol{\theta} \in \mathcal{W} \subset \mathbb{R}^d\}$. Given a new test input $\mathbf{x}^*$, the learner must generate a corresponding label $\hat{y}^*$. A probabilistic approach to the problem starts by specifying a likelihood, $p(\mathcal{D}|\boldsymbol{\theta})$. One option available to the learner is to simply predict $\hat{y}^* = f(\mathbf{x}^*|\hat{\boldsymbol{\theta}})$, where $\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} p(\mathcal{D}|\boldsymbol{\theta})$. What should the learner do when there are multiple, equally-good choices for $\boldsymbol{\theta}$ that result in different predicted labels? If one choice of $\boldsymbol{\theta}$ narrowly edges out another, and they disagree, should the predictions of the loser simply be discarded? A Bayesian learner avoids these quandaries by considering the predictions of all possible $\boldsymbol{\theta}$, weighted by their agreement with the data and a prior, $p(\boldsymbol{\theta})$. A quick application of Bayes theorem results in the expressions for the posterior likelihood of $\boldsymbol{\theta}$

$$p(\boldsymbol{\theta}|\mathcal{D}) = p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})/p(\mathcal{D}) \tag{1}$$

and the Bayesian model average

$$\mathop{\mathbb{E}}_{\boldsymbol{\theta}\sim p(\cdot|\mathcal{D})}[f(\mathbf{x};\boldsymbol{\theta})] = \int f(\mathbf{x};\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})d\boldsymbol{\theta}. \tag{2}$$

While a Bayesian learning algorithm is certainly a philosophically appealing way to resolve competing explanations of the data, many deep learning models are decidedly not Bayesian.

Bayesian algorithms encounter difficulties at each of the three distinctive steps of the algorithm (prior specification, posterior inference, and model averaging). The prior must be chosen to accurately reflect the *a priori* beliefs of the learner about what $\boldsymbol{\theta}$ are likely. When $\boldsymbol{\theta}$ is the weights and biases of a neural network, it is difficult (though not impossible) to determine a reasonable choice of prior (Neal, 2012; Nalisnick et al., 2018; Pearce et al., 2019). During posterior inference with neural networks, analytic solutions like those for simple linear models with Gaussian priors are unavailable. In fact, we might not even be able to sample from the posterior directly. Finally during model averaging the integral in Eq. 2 is also typically analytically intractable, requiring further approximations. While these challenges may seem discouraging, we should not be surprised by them. It is fundamentally more difficult to attempt to consider all possible hypotheses than it is to go all-in on a single max-likelihood solution. In this work, we will give a general overview of research tackling the challenges of posterior inference and model averaging in the context of deep learning.

A Bayesian learner has three main resources: train-time compute, test-time compute, and memory. Bayesian methods for deep learning can broadly be grouped into four main categories (Table 1). Each of these categories represent different tradeoffs between compute resources, test-time latency, estimator bias and variance.

Method	Advantages	Disadvantages
MCMC	Asymptotically unbiased	High variance and high latency
VI	Low latency and low memory cost	Asymptotically biased
Ensembles	Low bias, low variance (empirically)	Computationally intensive
Dropout	Simple, computationally cheap	Uni-modal posterior approximation

Table 1: Summary of Bayesian methods for deep learning

.

MCMC (Section 2) methods attempt to sample $\boldsymbol{\theta}$ from the posterior, and use the samples to estimate the Bayesian model average (BMA). Variational methods (Section 3) frame posterior inference as an optimization problem by first positing a parametric form for the posterior $q(\boldsymbol{\theta}; \mathbf{z})$ and then optimizing the variational parameters $\mathbf{z}$. Ensemble methods (Section 4) explicitly store and optimize different copies of the network weights, bypassing explicit posterior inference in favor of cheap BMA estimates. Dropout (Section 5) also bypasses posterior inference and directly estimates the BMA by randomly dropping connections between hidden units. The optimal choice of method typically depends on the available compute resources and the requirements of the application. In Section 6 we present some recent applications of deep learning and discuss how the application domain influenced the choice of inference procedure.

2. Markov Chain Monte Carlo Methods

Monte Carlo methods are among the oldest, most extensively studied algorithms for Bayesian statistical inference in high-dimensional parameter spaces. In this section we will attempt to

give a general overview of the important ideas in the literature. For an extensive treatment of Monte Carlo methods for neural networks, the reader is encouraged to see Neal (2012). The fundamental characteristic of all Bayesian Monte Carlo methods is the approximation of Eq. 2 with a Monte Carlo sum

$$\mathop{\mathbb{E}}_{\boldsymbol{\theta}\sim p(\cdot|\mathcal{D})}[f(\mathbf{x};\boldsymbol{\theta})] \approx \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x};\boldsymbol{\theta}^{(i)}), \tag{3}$$

where $\mathcal{W} := \{\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(N)}\}$ is some sequence of parameters. Typical convergence rates for Monte Carlo approximations assume that each $\boldsymbol{\theta}^{(i)}$ is drawn independently from the distribution of interest (in this case $p(\boldsymbol{\theta}|\mathcal{D})$). Posteriors over neural network parameters are not typically known in closed-form, and in general we cannot draw independent samples from the posterior. Fortunately we can still use Eq. 3 as long as $\mathcal{W}$ is generated by a Markov chain that has $p(\cdot|\mathcal{D})$ as its invariant distribution.

A sufficient (though not necessary) condition for $p(\cdot|\mathcal{D})$ to be an invariant distribution of a Markov chain with transition probabilities $t(\boldsymbol{\theta}'|\boldsymbol{\theta})$ is called *detailed balance*. For any distribution $q(\boldsymbol{\theta})$, the condition for detailed balanced is

$$t(\boldsymbol{\theta}'|\boldsymbol{\theta})q(\boldsymbol{\theta}) = t(\boldsymbol{\theta}|\boldsymbol{\theta}')q(\boldsymbol{\theta}') \tag{4}$$

An ergodic Markov chain has a unique invariant distribution, and will converge to that distribution if the chain is run long enough. Because the elements of a sequence generated by a Markov chain are often highly correlated, in general the chain must be run for a long time to ensure that it has adequately mixed, and that the distribution of $\mathcal{W}$ has converged.

2.1 The Metropolis-Hastings Algorithm

The Metropolis-Hastings (MH) algorithm (Metropolis et al., 1953; Hastings, 1970) is a foundational example of an MCMC method. It is notable for its simplicity. To construct a Markov chain with invariant distribution $q(\boldsymbol{\theta})$, we specify a *proposal* distribution $s(\boldsymbol{\theta}'|\boldsymbol{\theta})$, and use it to draw new candidates $\boldsymbol{\theta}' \sim s(\cdot|\boldsymbol{\theta}^{(i)})$. If $q(\boldsymbol{\theta}') \geq q(\boldsymbol{\theta}^{(i)})$, we set $\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}'$ w.p. 1. Otherwise, $\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}'$ w.p. $q(\boldsymbol{\theta}')/q(\boldsymbol{\theta}^{(i)})$. If we reject $\boldsymbol{\theta}'$, $\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(i)}$ and the process repeats. The proposal distribution s must be symmetric in order to satisfy Eq. 4. While the MH algorithm is simple to implement, the proposal distribution must be tuned carefully to obtain good results. Typically, you want the proposal distribution to be wide, so that Markov chain is not trapped in a region near its initialization for too long. However, often the proposal distribution must be fairly tight, otherwise most of the proposed samples are thrown away and once again the chain barely moves. Typically even when some balance is struck, the proposal distribution is still so conservative that the chain's behavior resembles a random walk, increasing the mixing time.

2.2 Hamiltonian Monte Carlo

Hamiltonian Monte Carlo (Duane et al., 1987) seeks to solve the issue of the MH algorithm's slow exploration of the posterior parameter space by reducing the random-walk behavior of naïve MH. Instead of randomly drawing candidate states from a proposal distribution, new

states are generated by following fictitious Hamiltonian dynamics. Given an unnormalized form for the posterior

$$p(\boldsymbol{\theta}|\mathcal{D}) \propto \exp(-E(\boldsymbol{\theta})),$$

where $E(\boldsymbol{\theta})$ is the "potential energy" of $\boldsymbol{\theta}$ w.r.t. the posterior, we can define auxiliary momentum variables $\mathbf{p} \in \mathbb{R}^d$ and the standard Euclidean kinetic energy function $K(\mathbf{p}) := (\mathbf{p}^\top \boldsymbol{M}^{-1}\mathbf{p})/2$, where $\boldsymbol{M}$ is the "mass" matrix of $\boldsymbol{\theta}$. Commonly we take $\boldsymbol{M}s = \boldsymbol{I}_d$ (Neal, 2012). The Hamiltonian $H(\boldsymbol{\theta}, \mathbf{p}) := E(\boldsymbol{\theta}) + K(\mathbf{p})$ measures the total energy. Note that by convention the first argument of the Hamiltonian (the generalized position in Hamiltonian mechanics) is usually written as $\mathbf{q}$, but we have chosen to maintain our notation convention for $\boldsymbol{\theta}$ for consistency. The dynamics of Hamiltonian systems are governed by Hamilton's equations,

$$\frac{d\boldsymbol{\theta}}{dt} = \nabla_{\mathbf{p}} H = \boldsymbol{M}^{-1}\mathbf{p}, \qquad \frac{d\mathbf{p}}{dt} = -\nabla_{\boldsymbol{\theta}} H = -\nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta}).$$

Hamiltonian Monte Carlo works by approximating the Hamiltonian dynamics with a finite discretization scheme. The deterministic dynamics transitions alternate with "heatbath" updates of the momentum. If the latter were not performed the chain would only explore the region of the state space with the same energy as the initialization. Finally to account for possible bias introduced by the discretization scheme, a Metropolis acceptance rule is used for candidate states $(\boldsymbol{\theta}^*, \mathbf{p}^*)$, with an acceptance propbability $\min(1, \exp\{-(H(\boldsymbol{\theta}^*, \mathbf{p}^*) - H(\boldsymbol{\theta}, \mathbf{p}))\})$. Because the discretized, deterministic transitions approximately conserve the energy of the starting state, HMC chains can generate candidate states far from the starting state with high acceptance probabilities. As a consequence, HMC chains explore the posterior density in much fewer steps than naïve Metropolis-Hastings.

2.3 Scaling Monte Carlo Methods to Large Datasets

One of the major advantages of maximum-likelihood training is that the objective often decomposes additively across the training set, and is thus very amenable to stochastic approximation via mini-batching. Stochastic Gradient Langevin Dynamics (Welling and Teh, 2011) is an MCMC algorithm that blurs the lines between optimization and posterior inference. The weight update in SGLD is written as

$$\Delta\boldsymbol{\theta}_t = \frac{\epsilon_t}{2}\left(\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}_t) + \frac{N}{n}\sum_{i=1}^{n} \log p(y_i|x_i, \boldsymbol{\theta}_t)\right) + \eta_t, \quad \eta_t \sim \mathcal{N}(0, \epsilon_t).$$

If the weight update step-size and the scale of the gradient noise are appropriately balanced, the SGLD chain converges to the posterior as its equilibrium distribution. Unfortunately SGLD exhibits random-walk-like behavior. As we saw, HMC avoids random walk behavior, but the energy function $E(\boldsymbol{\theta})$ is typically a function evaluated over the entire dataset. Chen et al. (2014) showed that a naïve implementation of HMC with noisy gradients is almost arbitrarily bad for posterior inference. The work proposes SG-HMC (Stochastic Gradient HMC), which adds a friction term to counteract the noise in the gradients. Letting $\nabla_{\boldsymbol{\theta}}\tilde{E}(\boldsymbol{\theta})$ be the minibatch stochastic approximation of $\nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta})$, SG-HMC introduces two new parameters, $\widehat{\boldsymbol{B}}$ and $\boldsymbol{C}$. $\widehat{\boldsymbol{B}}$ is a user-defined approximation of the diffusion matrix $\boldsymbol{B}$ induced by the noise in the gradient of E, and $\boldsymbol{C}$ is a user-defined friction matrix. The main idea is to choose

$\boldsymbol{C} \succeq \boldsymbol{B}$ and use a decaying step-size in the discretization of the dynamics. The weight update of SG-HMC is the same as standard HMC, and the new momentum update is given by

$$\Delta \mathbf{p}_t = \epsilon_t(-\nabla_{\boldsymbol{\theta}} \tilde{E}(\boldsymbol{\theta}) - \boldsymbol{C}\boldsymbol{M}^{-1}\mathbf{p}) + \eta_t, \quad \eta_t \sim \mathcal{N}(0, 2(\boldsymbol{C} - \widehat{\boldsymbol{B}})\epsilon_t),$$

where $\epsilon_t \to 0$ is the step-size of the dynamical discretization scheme.

2.4 Closing Remarks

SG-HMC resolves one of the major computational bottlenecks of HMC (computing $\nabla_{\boldsymbol{\theta}} E$ on the whole dataset). However, HMC also relies on a Metropolis-Hastings acceptance rule to correct any bias resulting from the discretization of the dynamics. The Metropolis-Hastings step also naïvely requires a computation over the whole dataset. Procedures for adapting Metropolis-Hastings to stochastic approximations of the acceptance ratio is an active area of research (Bardenet et al., 2017). Recent work showing the importance of capturing multiple modes of the posterior (Snoek et al., 2019; Wilson and Izmailov, 2020) has prompted continued research in HMC (Piponi et al., 2020), but at least for now in the context of deep learning MCMC methods are largely overshadowed by variational methods, dropout, and ensembles.

3. Variational Methods

While Monte Carlo methods have nice asymptotic properties, they have a couple major drawbacks. Aside from the challenges around tuning the hyperparameters of the Markov chain, MCMC methods cannot provide a compact parametric representation of the posterior, and all of the weight samples must be stored to make test time predictions. Variational methods approach the problem of posterior inference in a very different way. All variational methods start by positing a parametric form of the posterior $q(\boldsymbol{\theta}; \mathbf{z}) \approx p(\boldsymbol{\theta}|\mathcal{D})$. The parameters $\mathbf{z}$ are optimized to minimize some measure of distance between $q(\cdot; \mathbf{z})$ and $p(\cdot|\mathcal{D})$. The KL-divergence from p to q (Eq. 5) is a very common choice of distance measure.

$$\mathrm{D}_{\mathrm{KL}}(q||p) = \int \log\left(\frac{q(\boldsymbol{\theta}; \mathbf{z})}{p(\boldsymbol{\theta}|\mathcal{D})}\right) q(\boldsymbol{\theta}; \mathbf{z}) d\boldsymbol{\theta} \tag{5}$$

$$= \mathop{\mathbb{E}}_{\boldsymbol{\theta}\sim q}\left[\log(q(\boldsymbol{\theta}; \mathbf{z})) - \log(p(\boldsymbol{\theta}, \mathcal{D})\right] + \log p(\mathcal{D}) \tag{6}$$

If we define $\ell(\mathbf{z}) := \mathop{\mathbb{E}}_{\boldsymbol{\theta}\sim q}\left[-\log(q(\boldsymbol{\theta}; \mathbf{z})) + \log(p(\boldsymbol{\theta}, \mathcal{D})\right]$ and rearrange Eq. 6 we get

$$\log p(\mathcal{D}) = \mathrm{D}_{\mathrm{KL}}(q||p) + \ell(\mathbf{z})$$

Since $\log p(\mathcal{D})$ (a quantity known as the *evidence*) is constant w.r.t. $\mathbf{z}$, maximizing $\ell(\mathbf{z})$ implies that $\mathrm{D}_{\mathrm{KL}}(q||p)$ must decrease. By definition, $\mathrm{D}_{\mathrm{KL}}(q||p) \geq 0$, so we have $\ell(\mathbf{z}) \leq \log p(\mathcal{D})$, and hence $\ell(\mathbf{z})$ is commonly called the *Evidence Lower BOund*, or ELBO. ℓ is also sometimes called the *variational free energy*, referencing the method's roots in statistical physics (Mézard et al., 1987). Stochastic variational inference (Hoffman et al., 2013) estimates $\nabla_{\mathbf{z}}\ell$ from minibatches of data, allowing variational inference to scale to large datasets.

3.1 Practical VI for Neural Networks

One of the earliest works applying VI to neural networks is found in Graves (2011), which was in turn influenced by an information-theoretic connection between weight uncertainty and minimum description length (Hinton and Van Camp, 1993). One of the simplest forms of q one could choose is the Dirac-delta distribution on $\mathbb{R}^d$, where all of the density is concentrated in a point-mass at $\mathbf{z}$. If the prior $p(\boldsymbol{\theta})$ is uniform then ℓ is equivalent the max-likelihood objective. Similarly, if the prior is a zero-centered Laplace or isotropic Gaussian distribution, ℓ is equivalent to the max-likelihood objective with L-1 or L-2 regularization, respectively. A more interesting choice of variational posterior is a diagonal Gaussian (also known as a *mean-field* variational approximation),

$$\mathbf{z} = \{\boldsymbol{\mu}, \boldsymbol{\sigma}\} \in \mathbb{R}^d \times \mathbb{R}^d_+, \quad q(\cdot; \mathbf{z}) = \mathcal{N}(\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2)).$$

Unsurprisingly the gradient the ELBO w.r.t. the variational parameters $\nabla_{\mathbf{z}}\ell$ does not have an analytic form, so Graves (2011) proposed a Monte Carlo approximation of $\nabla_{\mathbf{z}}\ell$. If $\boldsymbol{\sigma}$ is optimized the estimated gradient has a term that depends on the diagonal of the Hessian of the likelihood. Since computing Hessians for large neural networks is typically computationally expensive, the Hessian term was further approximated with the diagonal of the empirical Fisher matrix. If $\boldsymbol{\sigma}$ is not optimized, then the posterior update reduces to a gradient step on a modified max-likelihood loss with weight noise and a intermediate update on the parameters of the prior.

While the Monte Carlo estimate of $\nabla_{\mathbf{z}}\ell$ proposed in Graves (2011) was straightforward to implement, it was biased. In a follow-up work Blundell et al. (2015) applied the reparameterization trick (Opper and Archambeau, 2009; Kingma and Welling, 2013) to derive an unbiased Monte Carlo estimate of the gradient. Recalling the diagonal Gaussian form of $q(\boldsymbol{\theta}; \mathbf{z})$, $\mathbf{z} = \{\boldsymbol{\mu}, \boldsymbol{\sigma}\}$, the reparameterization trick results from the simple observation that

$$\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}) \iff \boldsymbol{\theta} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\varepsilon},\ \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}),$$

where $\odot$ indicates element-wise multiplication. For a fixed draw $\boldsymbol{\theta}^{(i)} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\varepsilon}^{(i)}$, we can rewrite ℓ as a deterministic function of $\boldsymbol{\theta}^{(i)}, \boldsymbol{\varepsilon}^{(i)}$, and $\mathbf{z}$ and stochastic estimates of the gradients of the variational parameters are readily obtained. In a slight abuse of notation, let $\boldsymbol{\sigma}^{-1} = (1/\sigma_1, \ldots, 1/\sigma_d)^\top$. Applying a change of variable $\boldsymbol{\rho} = \log(1 + \log(\boldsymbol{\sigma}))$, by the chain rule we have

$$\nabla_{\boldsymbol{\mu}}\ell = \nabla_{\boldsymbol{\theta}}\ell + \nabla_{\boldsymbol{\mu}}\ell, \quad \nabla_{\boldsymbol{\rho}}\ell = \nabla_{\boldsymbol{\theta}}\ell \odot \boldsymbol{\varepsilon}^{(i)} \odot \boldsymbol{\sigma}^{-1} + \nabla_{\boldsymbol{\rho}}\ell.$$

3.2 Probabilistic Backpropagation

The work of (Hernández-Lobato and Adams, 2015) is closely related to that of (Graves, 2011) and (Blundell et al., 2015). All of these works factorized the approximate posterior as a diagonal Gaussian, but differed in their inference procedure. (Hernández-Lobato and Adams, 2015) framed the inference problem as *assumed density filtering* (Opper and Winther, 1998), rather than variational inference, and ultimately performed an approximate expectation-propagation update (Minka, 2001) to infer the parameters of the approximate posterior. During forward propagation, a Gaussian random variable is used to approximate the marginal

posterior distribution of each weight. This is done sequentially. As we propagate to the output layer, the marginal log-probability of the output is calculated instead of a prediction error. Backpropagation is done as usual and we obtain gradients, which are used to update means and variance of posterior approximation. However, the update is computed using the Bayes' rule.

3.3 Matrix-Gaussian Variational Posteriors

While a mean-field variational form of the posterior greatly simplifies analysis and lends itself to straightforward implementation schemes, it greatly restricts the expressiveness of the variational approximation, since a diagonal Gaussian precludes correlation between the different weight components of the network. Due to the highly structured architecture of neural networks, it is very likely that the true posterior encodes strong correlations between different weight components. (Louizos and Welling, 2016) specified q as a product of matrix-variate Gaussian distributions with parameters $\mathbf{z} = \{(\boldsymbol{M}_1, \boldsymbol{U}_1, \boldsymbol{V}_1), \ldots, (\boldsymbol{M}_L, \boldsymbol{U}_L, \boldsymbol{V}_L)\}$, where $\boldsymbol{M}_i \in \mathbb{R}^{n \times p}$, $\boldsymbol{U}_i \in \mathbb{R}^{n \times n}$, $\boldsymbol{V}_i \in \mathbb{R}^{p \times p}$, and L is the number of layers of the network. Note that here we are considering $\boldsymbol{\theta}$ to be a collection of random matrices $\{\boldsymbol{W}_1, \ldots, \boldsymbol{W}_L\}$, $\boldsymbol{W}_i \in \mathbb{R}^{n \times p}$ and $d = n \times p$. Specifically, the matrix Gaussian variational posterior is given by

$$q_i(\boldsymbol{W}_i; \mathbf{z}_i) = \frac{\exp\left[-\text{tr}\left(\boldsymbol{V}_i^{-1}(\boldsymbol{W}_i - \boldsymbol{M}_i)^\top \boldsymbol{U}_i^{-1}(\boldsymbol{W}_i - \boldsymbol{M}_i)\right)/2\right]}{(2\pi)^{np/2}|\boldsymbol{V}_i|^{n/2}|\boldsymbol{U}_i|^{p/2}},$$

$$q(\boldsymbol{\theta}; \mathbf{z}) = \prod_{i=1}^{L} q_i(\boldsymbol{W}_i; \mathbf{z}_i).$$

In contrast to the diagonal Gaussian variational posterior, which has independent parameters for each weight component, a matrix-variate Gaussian variational posterior treats the layers of the network as random matrices, and can encode correlation between the layer inputs and outputs.

3.4 Closing Remarks

As we have noted, conventional wisdom suggests that variational inference is useful for fast (computationally cheap), biased estimates of expectations w.r.t. the posterior, whereas MCMC yields slow, unbiased estimates (though possibly with high variance). Recent work in Hoffman and Ma (2019) has challenged this consensus, by showing that the Langevin dynamics in SGLD (Welling and Teh, 2011) follow a gradient flow corresponding to a nonparametric normalizing flow variational approach, known as *black-box* variational inference (Ranganath et al., 2014). Intriguingly, that work shows that for a small, fixed computational budget, many parallel chains of MCMC may provide *better* estimates than the BBVI approximation obtained with the same amount of computation. Given that recent gains in processing power have largely been due to massive parallelism (e.g. GPUs and TPUs), variational algorithms may struggle to keep pace with more parallelizable approaches like MCMC and ensembles, since variational inference is rooted in optimization, an inherently sequential procedure.

4. Ensemble Methods

Both MCMC methods and VI methods have some main drawbacks. The incompatibility of them makes us unable to have a balance between them. We further introduce ensemble methods which have empirical success on low bias and variance. Ensemble methods have been widely used in machine learning, especially in decision tree learning. One example is *random forest* (Breiman, 2001). It repeats this process: it randomly selects a part of training data and trains a classification or regression tree. After it gets enough trees, it will average predictions from them. This method is a variant of *bagging* methods (Breiman, 1996) in ensemble learning. Similar way can be applied to Bayesian deep learning methods: we train several neural networks whose parameters can be regarded as samples from posterior distribution, then use Eq. 3 to predict. According to sampling methods, ensemble methods can be divided into two branches: *multi-model ensembles* and *SGD trajectory ensembles*.

4.1 Simple Ensemble Method

Deep ensembles (Lakshminarayanan et al., 2017) are very fundamental multi-model ensembles. Assuming random initialization of parameters and random shuffle of data is enough to reduce correlation between base models, it simply trains several networks and takes them as samples to predict uncertainty. Note that this method is non-Bayesian because it does not apply the Bayes' rules, but still performs as well as Bayesian methods.

In detail, a proper scoring rule (Gneiting and Raftery, 2007) is introduced to give numerical scores for rewarding better calibrated prediction. To maximize likelihood, the score function is set as $S(p_{\boldsymbol{\theta}}, (\boldsymbol{x}, \boldsymbol{y})) = \log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y})$. For regression tasks, The output is two values: the predicted mean $\mu_{\boldsymbol{\theta}}(\boldsymbol{x})$ and variance $\sigma^2_{\boldsymbol{\theta}}(\boldsymbol{x})$. The target $\boldsymbol{y}$ is regarded as one sample from a Gaussian distribution with $\mu_{\boldsymbol{\theta}}(\boldsymbol{x})$ and $\sigma^2_{\boldsymbol{\theta}}(\boldsymbol{x})$, so it minimizes the negative log-likelihood score function as

$$-\log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y}) = \frac{\log \sigma^2_{\boldsymbol{\theta}}(\boldsymbol{x})}{2} + \frac{(\boldsymbol{y} - \mu_{\boldsymbol{\theta}}(\boldsymbol{x}))^2}{2\sigma^2_{\boldsymbol{\theta}}(\boldsymbol{x})} + C$$

where C is a constant. Adversarial training (Goodfellow et al., 2014) is also used to smooth the distribution to reduce overfitting.

In summary, deep ensembles give us a simple way to measure uncertainty with robust performance. However, there are still some concerns about it: ensemble learning tells us that the base models of bagging should have little correlation between them. Otherwise, ensembles can overfit and have a bad generalization performance. Deep ensembles use default initialization for parameters. It means that they have same prior distribution. Besides, the sampling is mixed into training, so we are not sure what happens in this black box. Therefore, despite empirical success, the correlation between deep ensembles is not promising to be negligible.

4.2 Ensemble Method with Prior Distribution

As we concerned above, the prior distribution of deep ensembles can cause overfitting. Bayesian ensembles (Pearce et al., 2018) are multi-model ensembles and sample prior dis-

tributions for each model to reduce correlation. It introduces *anchor distribution* where this method samples the mean of prior distribution. The anchor distribution is approached by maximizing a posterior distribution estimation (MAP). For the Bayes' rule of multivariate Gaussian distributions with prior distribution $p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu}_{\text{prior}}, \boldsymbol{\Sigma}_{\text{prior}})$ and likelihood distribution $p(\mathcal{D}|\boldsymbol{\theta}) \propto \mathcal{N}(\boldsymbol{\mu}_{\text{like}}, \boldsymbol{\Sigma}_{\text{like}})$,

$$\mathcal{N}(\boldsymbol{\mu}_{\text{post}}, \boldsymbol{\Sigma}_{\text{post}}) \propto \mathcal{N}(\boldsymbol{\mu}_{\text{like}}, \boldsymbol{\Sigma}_{\text{like}})\mathcal{N}(\boldsymbol{\mu}_{\text{prior}}, \boldsymbol{\Sigma}_{\text{prior}})$$

The MAP solution is

$$\boldsymbol{\theta}_{\text{MAP}} = \boldsymbol{\Sigma}_{\text{post}}\boldsymbol{\Sigma}_{\text{like}}^{-1}\boldsymbol{\mu}_{\text{like}} + \boldsymbol{\Sigma}_{\text{post}}\boldsymbol{\Sigma}_{\text{prior}}^{-1}\boldsymbol{\mu}_{\text{prior}}$$

where $\boldsymbol{\Sigma}_{\text{post}} = (\boldsymbol{\Sigma}_{\text{like}} + \boldsymbol{\Sigma}_{\text{prior}})^{-1}$. Then, it sets $\boldsymbol{\mu}_{\text{prior}} := \boldsymbol{\theta}_{\text{anchor}}$, where $\boldsymbol{\theta}_{\text{anchor}}$ is a random variable sampled from anchor distribution. The anchor distribution is proved to be a multivariate Gaussian distribution with

$$\boldsymbol{\mu}_{\text{anchor}} = \boldsymbol{\mu}_{\text{prior}} \quad \text{and} \quad \boldsymbol{\Sigma}_{\text{anchor}} = \boldsymbol{\Sigma}_{\text{prior}} + \boldsymbol{\Sigma}_{\text{prior}}\boldsymbol{\Sigma}_{\text{like}}^{-1}\boldsymbol{\Sigma}_{\text{prior}}$$

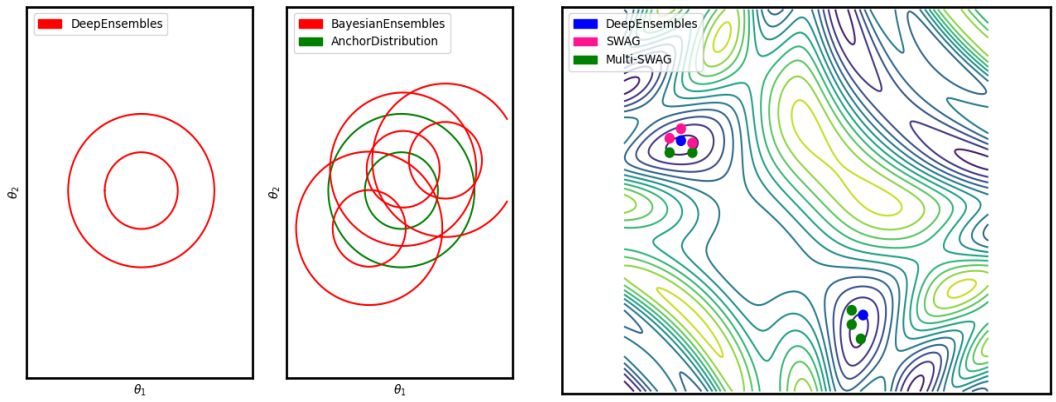
Because $\boldsymbol{\Sigma}_{\text{like}}^{-1}$ is intractable in large dataset, it approximates $\boldsymbol{\Sigma}_{\text{anchor}} = \boldsymbol{\Sigma}_{\text{prior}}$ to ignore this term.

Additional analysis shows this approximation will make the posterior distribution underestimates variance and overestimates correlation. A special case is that when all eigenvalues of scaled $\boldsymbol{\Sigma}_{\text{post}}$ equal to either 0 or 1, the approximation error will be zero. This method is expected to approach this condition to get higher accuracy. Fig. 1(a) shows a example of two-dimensional Gaussian prior distributions using deep ensembles and Bayesian ensembles. Concentric circles represent the 1σ and 2σ region of Gaussian distributions. We assume that deep ensembles and Bayesian ensembles use isotropic Gaussian distributions as prior distributions. The anchor distribution has been approximated with $\mathcal{N}(\boldsymbol{\mu}_{\text{prior}}, \boldsymbol{\Sigma}_{\text{prior}})$.

The experiments show that Bayesian ensembles have relatively better performance than deep ensembles, which proves the necessity of our concerns. Bayesian ensembles give us a more Bayesian method to make ensemble methods more robust. However, we are still worried about the approximation error. People are doing more analysis or explore other approximating methods to bypass it.

4.3 SWA Ensemble Method

In recent years, SGD trajectory ensembles have come into researchers' vision based on its easy implementation with no additional cost. This method treats the parameters of NN models as one point that lies on local optima in parameter space and ensembles these points to get more accurate prediction. Snapshot ensembles (Huang et al., 2017) uses cyclical cosine annealing learning rate (also called warm restarts (Loshchilov and Hutter, 2016)) to approach different local optima and jump out of them. Fast Geometric Ensembling (FGE) (Garipov et al., 2018) tries to find the path between local optima. When the model converges, this method uses changing learning rates with small cycle length to search in the path. Stochastic Weight Averaging (SWA) (Izmailov et al., 2018) is based on an observation: the point that



(a) The prior distribution of deep ensembles and Bayesian ensembles

(b) Results of deep ensembles, SWAG and multi-SWAG in loss landscape

Figure 1: Comparison of different ensemble methods

SGD converges to is often near the boundary of a wide flat region of optima. The center of this region generally has higher train loss but lower test loss. SWA runs exponential moving average on these local optima to get a better generalization performance.

We continue to introduce one method called SWA-Gaussian (Maddox et al., 2019). It assumes the local optima can be treated as a sample from a multivariate Gaussian distribution and approximates this distribution over SGD iterates. A simple approximation is a Gaussian distribution with a diagonal covariance matrix, that is,

$$\tilde{\boldsymbol{\theta}} \sim \mathcal{N}(\boldsymbol{\theta}_{\text{SWA}}, \boldsymbol{\Sigma}_{\text{Diag}}) \quad \text{and} \quad \boldsymbol{\Sigma}_{\text{Diag}} = \text{diag}\left(\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{\theta}_i^2 - \boldsymbol{\theta}_{\text{SWA}}^2\right) \tag{7}$$

where N is the number of saved models over SGD iterates, $\boldsymbol{\theta}_{\text{SWA}}$ is the result of SWA and $\boldsymbol{\theta}_i$ is the ith saved models. Eq. 7 shows an element-wise square operation. A further approximation is a Gaussian distribution with a low-rank covariance plus diagonal matrix calculated by some sampled models (e.g., the last K saved models). It is rewritten as

$$\tilde{\boldsymbol{\theta}} \sim \mathcal{N}\left(\boldsymbol{\theta}_{\text{SWA}}, \frac{1}{2}(\boldsymbol{\Sigma}_{\text{Diag}} + \boldsymbol{\Sigma}_{\text{Low}})\right)$$

$$\boldsymbol{\Sigma}_{\text{Low}} = \frac{1}{K-1}\sum_{i=1}^{K}(\boldsymbol{\theta}_i - \boldsymbol{\theta}_{\text{SWA}})(\boldsymbol{\theta}_i - \boldsymbol{\theta}_{\text{SWA}})^T = \frac{1}{K-1}\sum_{i=1}^{K}\boldsymbol{D}_i\boldsymbol{D}_i^T$$

To reduce memory cost, let $\boldsymbol{D}_i \approx \boldsymbol{\theta}_i - \bar{\boldsymbol{\theta}}_i$, where $\bar{\boldsymbol{\theta}}_i$ is running estimation of the mean of parameters. For classification tasks, SWAG shows high uncertainty on test data measured by predictive entropy (Shannon, 1948) when training dataset does not cover it, which is

$$\mathcal{H}_{\text{pred}}(y|\boldsymbol{x}) = \sum_{i=1}^{N} p(y = c_i|\boldsymbol{x}) \log p(y = c_i|\boldsymbol{x}) \tag{8}$$

where N is the number of classes and c_i is the label of the ith class, and high confidence on test data that is like train data.

SWAG gives us a more efficient way to predict posterior distribution rather than training several neural networks. Further research is focused on the geometry of the subspace where SGD trajectory lies. For example, can this subspace have some complicated shape that simple Gaussian distribution cannot fit? A recent work (Wilson and Izmailov, 2020) proposes multi-SWAG, a combination of multi-model ensembles and SGD trajectory ensembles. It runs SWAG for several times to fit multi-Gaussian distributions for different attractive basins. Fig. 1(b) shows the comparison between deep ensembles, SWAG and multi-SWAG. The region with blue contour has lower loss than one with yellow contour, and different points shows the samples from ensemble methods.

4.4 Closing Remarks

For low bias and variance, ensemble methods seem to be panaceas for Bayesian deep learning. Besides, ensemble methods can be implemented parallel in training and testing, so compared to MCMC methods and variational methods with serial sampling, they can use less time to sample. However, for multi-model ensembles, we have high computational cost in training several models. For SGD trajectory ensembles, if the loss landscape has several flat regions, they may have worse generalization performance. Both of them have high memory cost to store the parameters of several models. Recent work like batch ensembles (Wen et al., 2020) continues to improve the efficiency of parallel training and testing of ensembles. Given the rapid development of parallel and distributed computing, the potential of ensemble methods will be exploited sooner or later.

5. Dropout Methods

Since its first introduction in (Srivastava et al., 2014), dropout has become a popular regularization method due to its simplicity and computational efficiency. It attempts to improve the performance of a complex deep neural network by creating a collection of sub-networks of the original network and using an alternative approach to perform model averaging.

In a nutshell, given a deep neural network, we use independent Bernoulli random variables each with probability p of being 1 and $1-p$ of being 0 to randomly discard input and/or hidden units along with their connections and create trimmed neural networks. This potentially creates 2^N sub-networks with shared weights, where N is the number of units in the given neural network. In each iteration during training, one of the sub-networks is sampled and only parameters in that sub-network are learned and updated.

5.1 Dropout as Gaussian Process

To understand the success of dropout, (Gal and Ghahramani, 2016a) provides a Bayesian interpretation of dropout and connects dropout with model uncertainty. It shows that dropout used in neural networks is equivalent to Bayesian estimation of the variational inference in the deep Gaussian process marginalized over parameters in its covariance. By

showing dropout's relation to the Gaussian process, we are able to utilize properties of Gaussian process, which provides estimations of uncertainty in the neural network and reduce overfitting.

The uncertainty estimation uses the *MC dropout* method, which applies dropout during prediction also. MC dropout can also be approximated by weight averaging. During prediction, the neural network runs multiple forward passes on each input $\boldsymbol{x}^*$ to obtain a set of outputs $\boldsymbol{y}^*$. Each forward pass uses different sets of weights determined by dropout. Then, we obtain the prediction $\boldsymbol{y}^*$ by finding the posterior mean $\mathbb{E}_{q(\boldsymbol{y}^*|\boldsymbol{x}^*)}[\boldsymbol{y}^*]$ along with its uncertainty, i.e. the variance $\mathrm{Var}_{q(\boldsymbol{y}^*|\boldsymbol{x}^*)}(\boldsymbol{y}^*)$ conditioned on the predictive distribution $q(\boldsymbol{y}^*|\boldsymbol{x}^*)$. Without changing the network architecture, we are able to extract information that was previously ignored. This is especially handy if $\boldsymbol{x}^*$ was not in the training dataset.

5.2 Dropout as Structured Shrinkage Prior

Applying dropout to a neural network can be interpreted as introducing multiplicative noise to the model, where the noise is draw independently from a Bernoulli distribution. (Nalisnick et al., 2018) shows that multiplicative noise is equivalent to a Gaussian scale mixture with the automatic relevance determination structure (ARD-GSM prior).

The ARD is a Bayesian regularization technique which applies (Gaussian) priors on the weights and optimizes them using marginal likelihood. When weights are assumed to be Gaussian random variables, then the product of weights and noise is an ARD-GSM prior. By showing the equivalence between multiplicative noise and an ARD-GSM prior through reparameterization, we are able to remove dropout's generative assumptions during inference such as in the previous interpretation in (Gal and Ghahramani, 2016a). The objective function for multiplicative noise optimization with a L-2 regularization can be seen as a lower bound for the marginal MAP objective function.

5.3 Closing Remarks

Dropout utilizes Monte Carlo sampling and variational inference methods to approximate supposedly intractable computations. It can be seen as model averaging that is different from bagging in the ensemble methods. By understanding dropout in the sense of Bayesian, we provide two rigorous explanations on the the success of dropout. Moreover, instead of using dropout as regularization, it can also be used to other tasks such as obtaining model uncertainty.

6. Notable Applications

After introducing several Bayesian deep learning methods, we will show some notable applications of them on computer vision, physics and medical tasks. Bayesian neural networks perform effective in these tasks because first, deterministic models cannot work with high uncertainty of complicated system, and second, some tasks need high reliability. Bayesian methods can pick out confusing samples rather than make incorrect predictions.

Researchers are trying to use Bayesian deep learning in practical application to benefit from predicted uncertainty.

6.1 Computer Vision

Uncertainty in computer vision has been studied for a long time. Early work (Szeliski, 1990) was focused on uncertainty on low-level vision. Low-level vision is troubled by noisy from sensors and unstable low-level representation. This method exploits advantages of Bayesian methods to build a more robust model. After deep convolutional neural network (LeCun et al., 2015) was proved to efficiently extract abstract information from images, constructing uncertainty on high-level vision became possible. More researchers are now working on analyzing this uncertainty and model it by Bayesian neural networks.

One existing method (Kendall and Gal, 2017) indicates that uncertainty in computer vision can be divided into *aleatoric uncertainty* and *epistemic uncertainty*, and builds a model by using dropout methods introduced in Section 5. Aleatoric uncertainty comes from noise and varying representation, which cannot be reduced even if we have large training dataset. On the other hand, epistemic uncertainty becomes nonnegligible when predicted images are very different from training data. For aleatoric uncertainty, we can further define *homoscedastic uncertainty* and *heteroscedastic uncertainty*. The distinction between them is that, homoscedastic uncertainty has the same estimated variance on all inputs, while heteroscedastic uncertainty needs to predict different variance for each input. For regression task, the loss function with heteroscedastic uncertainty can be expressed as,

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{2\hat{\sigma}_i^2}||\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i||^2 + \frac{1}{2}\log \hat{\sigma}_i^2$$

where $\hat{\boldsymbol{y}}_i, \hat{\sigma}_i$ is the prediction of a sampled model $f^{\hat{\boldsymbol{\theta}}_i}(\boldsymbol{x})$. $\hat{\boldsymbol{\theta}}_i$ is sampled from approximate posterior distribution $q(\boldsymbol{\theta})$. One further method (Kendall et al., 2018) tells that homoscedastic uncertainty can be interpreted as task-dependent uncertainty. For two-task learning, the total loss function is,

$$\mathcal{L}_{\text{total}} = \frac{1}{2\sigma_1^2}\mathcal{L}_1 + \frac{1}{2\sigma_2^2}\mathcal{L}_2 + \log \sigma_1 + \log \sigma_2$$

where $\mathcal{L}_1, \mathcal{L}_2$ are the corresponding loss functions of task 1 and task 2, and σ_1, σ_2 are estimated variances. The authors reports increasing aleatoric uncertainty on objects far from the camera and object boundaries, and increasing epistemic uncertainty on challenging objects which are never seen in training data. To some extent, epistemic uncertainty shows the weak points of training dataset, so researchers can exploit this uncertainty to improve their dataset.

6.2 Physics

Bayesian methods has been applied to extract information from physics experiment (von Toussaint, 2011) especially when observation environment is poor. Sometimes, one dynamic system is very sensitive to initial conditions, which observers cannot measure with very high

accuracy. Therefore, it is reasonable to use Bayesian methods to improve model performance.

Atmospheric retrieval is one method to build atmospheric model for a planet with some uncertainty. Bayesian methods with neural networks are naturally fitted for more accurate measurement. One method (Kendall and Gal, 2017) uses dropout methods (Section 5) to predict atmospheric parameters. These parameters $\boldsymbol{\phi}$ are represented to be jointly distributed by a multivariate Gaussian distribution with a diagonal covariance matrix calculated by Cholesky decomposition $\text{Var}(\boldsymbol{\phi}) = \boldsymbol{\Lambda}^{-1} = (\boldsymbol{L}\boldsymbol{L}^T)^{-1}$. The output consist of a lower triangular matrix $\boldsymbol{L}$ of dimension $D \times D$ and a mean vector $\boldsymbol{\mu}$ of dimension D. After training, it predicts $\boldsymbol{L}(\boldsymbol{s})$ and $\boldsymbol{\mu}(\boldsymbol{s})$ by Monte Carlo estimation. Finally, it samples parameter $\boldsymbol{\phi}$ from predicted distribution. The result shows that dropout methods provide a reasonable estimation of parameters orders of magnitude much faster than traditional methods. However, they will underestimate uncertainty by using variational approximate inference. This can be made up by other Bayesian methods like MCMC.

6.3 Medical Tasks

Bayesian methods can measure the uncertainty of predictions, which is very important for medical tasks (Matchar et al., 1990). Medical tasks, like diagnostic tests, care more about false negative and false positive samples than regular tasks. Error diagnosis by machine learning models can cause disasters. Therefore, Bayesian methods can help point out confusing samples for recheck.

An existing method (Filos et al., 2019) uses dropout methods (Section 5) for diabetic retinopathy tasks. Each sample is graded by specialist from 0 to 4, so it is regarded as multi-class classification task. Uncertainty is measured by Eq. 8. There are some points we note about this method. First, class imbalance should be stress in medical tasks because positive examples are often much less than negative examples. Second, medical tasks usually have smaller dataset, and MCMC methods seem to have a better performance than MC dropout method. But traditional MCMC methods are not scalable in large dataset. This method reports that MC dropout method can benefit from large dataset to get higher accuracy, which encourages researchers to build a larger and general dataset for it.

7. Discussion

In closing, this section will discuss notable future research directions for Bayesian deep learning.

For MCMC methods, future research will focus on how to sample with both low variance and high efficiency. For example, to get more accurate estimation for SGLD and SG-HMC under looser assumptions, recent research (Seita et al., 2018) proposes a mini-batch M-H testing method to correct the error. People also try to find some methods like (Simsekli et al., 2016; Fu et al., 2016; Hoffman and Gelman, 2014) that can efficiently explore high-dimensional parameter space and well combined with SGD.

For variational methods, most of existing variational methods have some restrictions like mean-field assumption which forces parameters to be independent. Future research continues to explore better parametric distributions with lower approximation error, better measurement (Li and Turner, 2016; Regli and Silva, 2018) than KL-divergence and more flexible assumptions.

For ensemble methods, the sampling process of ensemble methods is implicit and mixed into training. Future research will delve into the geometry of parameter space to understand how parameters change in SGD process. Efficient sampling (Maddox et al., 2019; Wilson and Izmailov, 2020) and reduction of correlation between ensemble models are also potential directions to explore.

For dropout methods, fixed dropout rate cuts off the dependence between dropout and data, and if this rate is not well fine-tuned, it will cause high approximation error. Future research is focused on variants of dropout. For instance, in concrete dropout (Gal et al., 2017), researchers adapt dropout rate to approximate posterior distribution. In variational dropout (Kingma et al., 2015), researchers show that Gaussian dropout, a continuous dropout distribution, has a better approximate performance than Bernoulli dropout. People can also benefit from the progress of variational methods to improve dropout variational inference (Li and Gal, 2017). On the other hand, although dropout was initially introduced for a standard neural network, many works have shown that it can also be applied to convolutional neural networks (Wu and Gu, 2015; Park and Kwak, 2016; Hou and Wang, 2019) and recurrent neural networks (Gal and Ghahramani, 2016b; Krueger et al., 2016).

References

Rémi Bardenet, Arnaud Doucet, and Chris Holmes. On markov chain monte carlo methods for tall data. *The Journal of Machine Learning Research*, 18(1):1515–1557, 2017.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic gradient hamiltonian monte carlo. In *International conference on machine learning*, pages 1683–1691, 2014.

Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid monte carlo. *Physics letters B*, 195(2):216–222, 1987.

Angelos Filos, Sebastian Farquhar, Aidan N Gomez, Tim GJ Rudner, Zachary Kenton, Lewis Smith, Milad Alizadeh, Arnoud de Kroon, and Yarin Gal. A systematic comparison of bayesian deep learning robustness in diabetic retinopathy tasks. *arXiv preprint arXiv:1912.10481*, 2019.

Tianfan Fu, Luo Luo, and Zhihua Zhang. Quasi-newton hamiltonian monte carlo. In *UAI*, 2016.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016a.

Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027, 2016b.

Yarin Gal, Jiri Hron, and Alex Kendall. Concrete dropout. In *Advances in neural information processing systems*, pages 3581–3590, 2017.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. In *Advances in Neural Information Processing Systems*, pages 8789–8798, 2018.

Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Alex Graves. Practical variational inference for neural networks. In *Advances in neural information processing systems*, pages 2348–2356, 2011.

W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57, 1970.

José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869, 2015.

Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13, 1993.

Matthew D Hoffman and Andrew Gelman. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, 15(1): 1593–1623, 2014.

Matthew D Hoffman and Yian Ma. Langevin dynamics as nonparametric variational inference. In *2nd Symposium on Advances in Approximate Bayesian Inference*, 2019.

Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

Saihui Hou and Zilei Wang. Weighted channel dropout for regularization of deep convolutional neural network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8425–8432, 2019.

Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*, 2017.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584, 2017.

Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in neural information processing systems*, pages 2575–2583, 2015.

David Krueger, Tegan Maharaj, János Kramár, Mohammad Pezeshki, Nicolas Ballas, Nan Rosemary Ke, Anirudh Goyal, Yoshua Bengio, Aaron Courville, and Chris Pal. Zoneout: Regularizing rnns by randomly preserving hidden activations. *arXiv preprint arXiv:1606.01305*, 2016.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pages 6402–6413, 2017.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Yingzhen Li and Yarin Gal. Dropout inference in bayesian neural networks with alpha-divergences. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2052–2061. JMLR. org, 2017.

Yingzhen Li and Richard E Turner. Rényi divergence variational inference. In *Advances in Neural Information Processing Systems*, pages 1073–1081, 2016.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Christos Louizos and Max Welling. Structured and efficient variational deep learning with matrix gaussian posteriors. In *International Conference on Machine Learning*, pages 1708–1716, 2016.

Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. In *Advances in Neural Information Processing Systems*, pages 13132–13143, 2019.

David B Matchar, David L Simel, John F Geweke, and John R Feussner. A bayesian method for evaluating medical test operating characteristics when some patients' conditions fail to be diagnosed by the reference standard. *Medical Decision Making*, 10(2):102–111, 1990.

Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

Marc Mézard, Giorgio Parisi, and Miguel Virasoro. *Spin glass theory and beyond: An Introduction to the Replica Method and Its Applications*, volume 9. World Scientific Publishing Company, 1987.

Thomas Peter Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, Massachusetts Institute of Technology, 2001.

Eric Nalisnick, José Miguel Hernández-Lobato, and Padhraic Smyth. Dropout as a structured shrinkage prior. *arXiv preprint arXiv:1810.04045*, 2018.

Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

Manfred Opper and Cédric Archambeau. The variational gaussian approximation revisited. *Neural computation*, 21(3):786–792, 2009.

Manfred Opper and Ole Winther. A bayesian approach to on-line learning. *On-line learning in neural networks*, pages 363–378, 1998.

Sungheon Park and Nojun Kwak. Analysis on the dropout effect in convolutional neural networks. In *Asian conference on computer vision*, pages 189–204. Springer, 2016.

Tim Pearce, Mohamed Zaki, Alexandra Brintrup, Nicolas Anastassacos, and Andy Neely. Uncertainty in neural networks: Bayesian ensembling. *arXiv preprint arXiv:1810.05546*, 2018.

Tim Pearce, Mohamed Zaki, Alexandra Brintrup, and Andy Neely. Expressive priors in bayesian neural networks: Kernel combinations and periodic functions. *arXiv preprint arXiv:1905.06076*, 2019.

Dan Piponi, Matthew D Hoffman, and Pavel Sountsov. Hamiltonian monte carlo swindles. *arXiv preprint arXiv:2001.05033*, 2020.

Rajesh Ranganath, Sean Gerrish, and David M. Blei. Black box variational inference. In *International Conference on Artificial Intelligence and Statistics*, 2014.

Jean-Baptiste Regli and Ricardo Silva. Alpha-beta divergence for variational inference. *arXiv preprint arXiv:1805.01045*, 2018.

Daniel Seita, Xinlei Pan, Haoyu Chen, and John Canny. An efficient minibatch acceptance test for metropolis-hastings. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 5359–5363. International Joint

Conferences on Artificial Intelligence Organization, 7 2018. doi: 10.24963/ijcai.2018/753. URL https://doi.org/10.24963/ijcai.2018/753.

Claude E Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.

Umut Simsekli, Roland Badeau, Taylan Cemgil, and Gaël Richard. Stochastic quasi-newton langevin monte carlo. In *Journal of Machine Learning Research*, volume 48, 2016.

Jasper Snoek, Yaniv Ovadia, Emily Fertig, Balaji Lakshminarayanan, Sebastian Nowozin, D Sculley, Joshua Dillon, Jie Ren, and Zachary Nado. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, pages 13969–13980, 2019.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Richard Szeliski. Bayesian modeling of uncertainty in low-level vision. *International Journal of Computer Vision*, 5(3):271–301, 1990.

Udo von Toussaint. Bayesian inference in physics. *Rev. Mod. Phys.*, 83:943–999, Sep 2011. doi: 10.1103/RevModPhys.83.943. URL https://link.aps.org/doi/10.1103/RevModPhys.83.943.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688, 2011.

Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. *arXiv preprint arXiv:2002.06715*, 2020.

Andrew Gordon Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. *arXiv preprint arXiv:2002.08791*, 2020.

Haibing Wu and Xiaodong Gu. Towards dropout training for convolutional neural networks. *Neural Networks*, 71:1–10, 2015.