Bayesian Deep Learning

Roberto Halpin Gregorio (rgh224)

---

1 Introduction

The topic that will be discussed in this report is applying a Bayesian framework to deep learning models, which commonly goes by the name Bayesian deep learning. The paper that this report is primarily based on is ***Bayesian Deep Learning and a Probabilistic Perspective of Generalization*** **by Andrew Gordon Wilson and Pavel Izamilov** [1], which was recently published in 2020. The mathematical nomenclature and notations will primarily be based on the ones used in this paper, and in other similar literature. For lack of clutter, quotations and diagrams throughout the report will belong to this paper, unless otherwise specified.

The manuscript *The Case for Bayesian Deep Learning* [17] and the conference tutorial presentation *Bayesian Deep Learning and a Probabilistic Perspective of Model Construction* [16] both by Andrew Gordon Wilson will also be primary sources of reference. Additionaly, I will be using a planetary science paper, *A Bayesian neural network predicts the dissolution of compact planetary systems*, authored by Miles Cranmer et al. [3], as a real-world application of Bayesian deep learning. In addition, to these sources there are several secondary papers that the primary sources and I use to explain certain points in more detail.

2 Deep Learning Primer

To understand Bayesian Deep Learning we first must go over the basics of deep learning. Deep learning is a term describing a broad class of statistical or machine learning models. Like all statistical or machine learning models, deep learning models take in examples as input and output structured predictions. These predictions are determined by the architecture or structure of the model and its parameter values. The models are then trained using gradient information resulting from the output of the model and the loss, or risk.

We can loosely define deep learning models as neural networks with many layers. A neural network is a model which modifies the input by a weight and then performs a summation, in other words, a linear combination, which we will call the linear layer. This linear combination then becomes input to an activation function, which controls the amplitude of the output. These non-linear activation functions tend to be functions such as sigmoid, tanh, or the rectified linear unit (ReLU). The reasoning behind the non-linear activation functions is so that the composition of a linear layer and the activation function produces an expressive function. Deep learning models are essentially neural networks with many repetitions of this linear layer + activation function setup with varying weight dimensions and choice of activation function. Deep learning occurs when we scale this functional framework to a massive scale, meaning that our layer weights become quite large and we use a substantial amount of layers. Of course this is a very coarse description of deep learning models and many do not exactly follow this pattern, but it is a sufficient description to achieve the understanding needed for Bayesian deep learning.

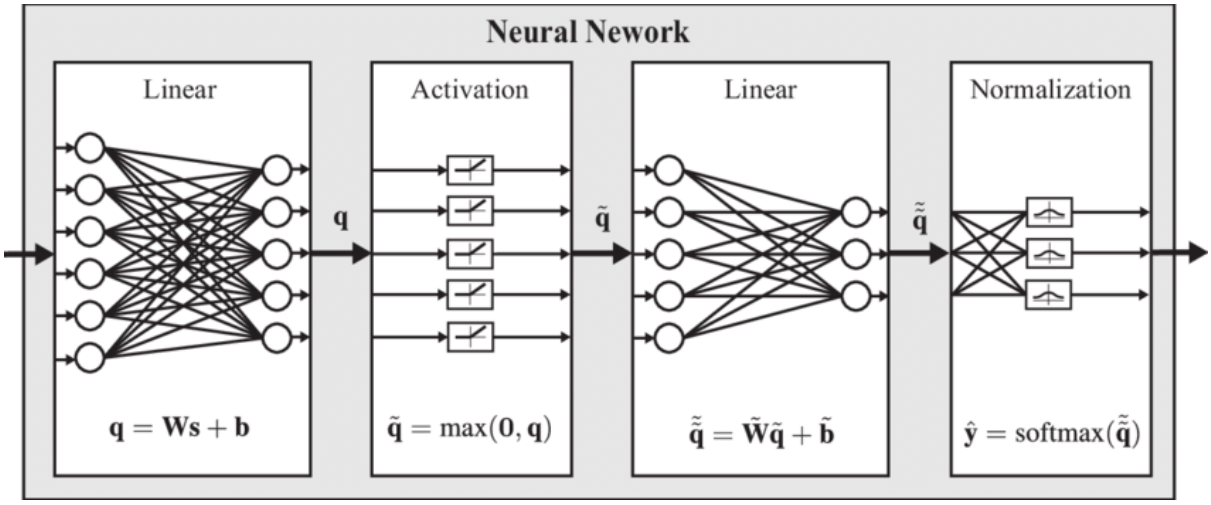


Figure 1: Two Layer Neural Network [2] (Normalization layer is also an activation function).

I will briefly go over two types of deep learning models since they will be mentioned in future sections. These models are the multi-layer preceptron (MLP) and the convolutional neural network (CNN). The MLP is exactly how we described our deep learning models, many layers of linear combinations with non-linear activation functions. Theoretical results show that MLPs are able to represent any possible mathematical function even when they only consist of a small

amount of layers. Thus, they are very expressive, but they also do not have any particular biases build into them, thus they are seen as being generalist. The second model, the CNN, is a bit more complex, so I will discuss it at a high level. It follows the same idea as the MLP, many consecutive layers, however instead of a linear layer, it uses a convolutional layer. This convolutional layer consists of filters that slide along input features and provide translation equivariant outputs. In other words, these are less expressive than MLPs due to the constraining feature of the convolutional layer. However, they have specific biases that make them very powerful in particular tasks, especially in visual imagery.

Deep learning models are very powerful and popular, but also quite mysterious. Since they are essentially a massive function composition, with many non-linearities, they are extremely non-convex, so we have no global optimal parameter settings. In addition, they are at such a scale that it is impossible to completely analyze the training of these non-convex models using today's tools. Even though they are seen as black-boxes, they are heavily used in Computer Vision, Natural Language Processing, and many other fields.

3 A Perspective on Generalization

In this section we will examine how systems that learn, in the setting of machine or statistical learning, are able to generalize to new examples that they have not previously seen.

First, let us define some key terms that will help us understand the idea of generalization. "The ability for a system to learn is determined by its support (which solutions are a priori possible) and inductive biases (which solutions are a priori likely). The evidence, or marginal likelihood, $p(\mathcal{D}|\mathcal{M}) = \int p(\mathcal{D}|\mathcal{M}, w)p(w)dw$, is the probability we would generate a dataset if we were to randomly sample from the prior over functions $p(f(x))$ induced by a prior over parameters $p(w)$." We can think of this in another way: We first choose a model/hypothesis class, and marginalize through all possible parameter settings of that model class with their respective prior probabilities. "The support is the range of datasets for which $p(\mathcal{D}|\mathcal{M}) \geq 0$." Inductive biases are "the relative prior probabilities of different datasets — the distribution of support given by $p(\mathcal{D}|\mathcal{M})$." Another way of thinking about inductive biases is given our model class

and its support over the datasets, where is the probability mass located, or distributed, it is a statement about the distribution. Overall, it is important to understand that generalization, as portrayed here, is a two-dimensional concept, it relies on both the support and the inductive biases. Additionally, deep learning models tend to generalize well because they have a large support and are able to be tweaked to develop well-calibrated inductive biases (e.g. CNN), thus they fit many datasets.

We will now discuss in more detail why exactly these deep learning models are able to fit many datasets of interest, and why a Bayesian approach may be desirable.

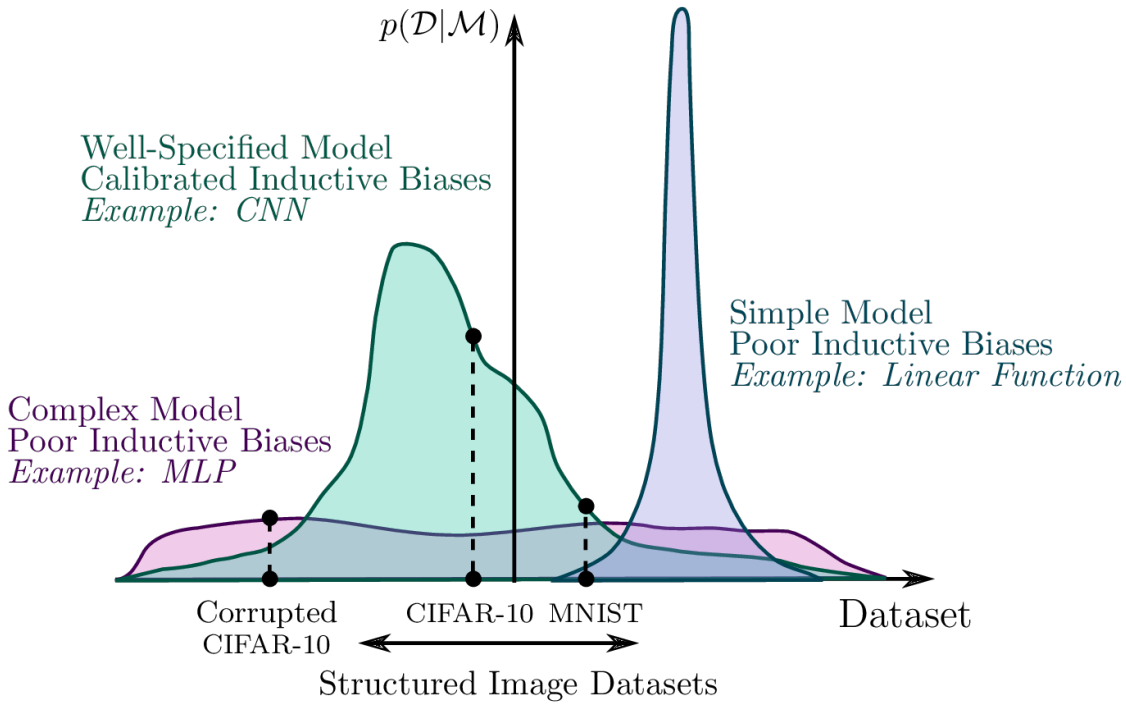


Figure 2: Evidence over image datasets

In figure 2, we have several computer vision image datasets on the x-axis and the evidence on the y-axis. MNIST is a dataset of 28 pixels by 28 pixels handwritten digits, numbers 0 through 9. CIFAR-10 is a dataset of 32 pixels by 32 pixels real-world objects, such as cats, dogs, horses, ships, etc. Corrupted CIFAR-10 is the CIFAR-10 dataset but corrupted with random noise, so it represents an unnatural image dataset, since these corrupted images do not occur in world.

Here the blue curve could represents a linear function. It has a truncated support and cannot even represent a quadratic function. Given that the marginal likelihood must marginalize over the datasets $\mathcal{D}$, the linear model assigns a large amount of mass to the datasets it does support.

The red-purple curve could represent a large fully-connected MLP. This model has very high flexibility, but has no preference for certain datasets, distributing its support too evenly. This causes it not to be particularly compelling for many image datasets.

The green curve could represent a convolutional neural network (CNN), which "represents a compelling specification of support and inductive biases for image recognition: this model has the flexibility to represent many solutions, but its structural properties provide particularly good support for many image problems."

We will now examine another figure depicting how these model classes' posteriors contract the initial prior hypothesis space.

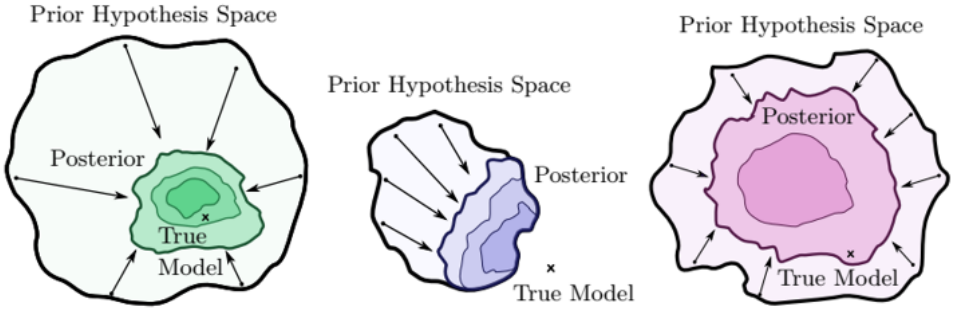


Figure 3: Collapse of probability

The convolutional neural network (CNN) representing the green diagram shows the benefit of having a large hypothesis space, combined with well-calibrated inductive biases, since the model contracts to the real solution. The blue diagram representing a linear model class is a simple model will have a posterior that contracts around an erroneous solution if it is not contained in the hypothesis space. Lastly, the red-purple model has a wide support, but does not contract around a good solution because its support is too evenly distributed, it does not have

well-calibrated inductive biases.

As we know, the marginal likelihood is used on many places such as Bayesian hypothesis testing, model comparison, and hyperparameter tuning, and in Bayes factors to select between models. Work has been done by Rasmussen & Ghahramani (2001) [15], which reasons that the marginal likelihood can favour large flexible models, as long as these models correspond to a reasonable distribution over functions. Also it is important not to confuse the *flexibility* of a model with the *complexity* of a model class. We can have deep learning models with large support, making them quite flexible, but often also have inductive biases towards simple solutions.

4 Bayesian Model Average

In the learning setting, the goal of a Bayesian approach is computing the Bayesian model average (BMA):

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

"The outputs are y (e.g., regression values, class labels, ...), indexed by inputs x (e.g. spatial locations, images, ...), the weights (or parameters) of the neural network $f(x; w)$ are w, and $\mathcal{D}$ are the data."

Instead of using a single setting of parameters, we use all possible parameter settings weighed by their posterior probabilities. Think of each setting of w as a different model, and that the Bayesian model average is an average of infinitely many models weighted by their posterior probabilities. Note that asymptotically, where we have infinite data, $p(w|\mathcal{D})$ will collapse onto a point mass.

5 Classical vs. Bayesian Approach

In the classic training setting we learn through optimization. This means we aim to find the a single model that minimizes or maximizes our objective of interest. This is typically the single setting of parameters w that maximizes the posterior distribution $p(w|\mathcal{D})$. This model is then used to predict our outputs y given an input x.

The Bayesian approach is based on the process of marginalization instead of optimization. As stated before, instead of using one singular model, we use the Bayesian model average to weigh all possible parameter settings using their posterior probabilities through integration or marginalization. This achieves our desired distribution $p(y|x, \mathcal{D})$.

We can also re-frame classical training in the lens of the Bayesian approach by showing how classical training is represented in the BMA. Classical training can be seen as using $p(w|\mathcal{D}) \approx \delta(w = \hat{w})$ where $\hat{w} = \text{argmax}_w p(w|\mathcal{D})$. This leads to the standard predictive distribution $p(y|x; \hat{w})$. Note that this $\hat{w}$ is equivalent to the maximum a posteriori probability (MAP) estimate, or w_{MAP}. If the actual posterior is not unimodal with a sharp peak, then the delta function is not a reasonable approximation. Instead we want to consider multiples modes of the posterior, which is exactly what we want to accomplish in the Bayesian setting.

Overall, the idea behind Bayesian Deep Learning can be simply stated as using the Bayesian model average (BMA) for deep learning models.

6 Bayesian Deep Learning

6.1 The case for Bayesian Deep Learning

Neural networks, or deep learning models, tend to be underspecified by the data. In other words, deep learning models have many more parameters than data. This leads to diffuse likelihoods $p(\mathcal{D}|w)$, which do not favor any one set of parameters. In addition, in many deep learning problems, there are many high performing models corresponding to different settings of parameters [4, 8]. This is exactly when marginalization will make the biggest difference for accuracy and calibration because we will have an ensemble containing many different but high performing models.

In most deep learning training setups, the loss or risk that we use tends to be the negative log posterior density $-\log p(w|\mathcal{D})$. Many studies have shown that flat regions of low loss are

associated with good generalization, visualized in figure 4. Additionally, these flat regions that generalize well tend to be diverse, which means that they correspond to many distinct high performing models [7]. To make matters even better, Huang et al. (2019) [6] describes deep learning models as having a blessing of dimensionality. This is because these flat, distinct regions of the loss that that correspond to better generalization take up much more volume in high dimensions. Thus, they will dominate in forming the predictive distribution in a Bayesian model average. These specific properties of deep learning models make using the BMA very appealing.

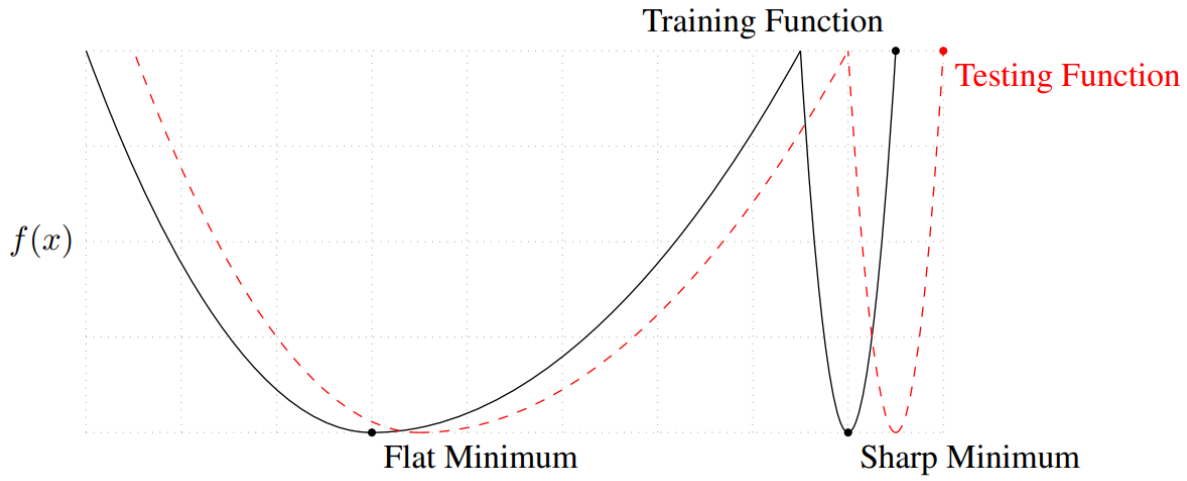


Figure 4: Generalization of solution types [9]

An additional reason why the Bayesian approach is appealing is that averaging the predictions of multiple, accurate models that disagree in some cases should lead to improved accuracy. As we discussed before in the classical vs. Bayesian section, the Bayesian approach will only provide benefits when it comes to accuracy because the classical approach is a very simplified version of the Bayesian approach. Calibration is also an important aspect when it comes to modeling. It has been noticed that modern deep learning models are often miscalibrated in the sense that their predictions are typically overconfident (Guo et al., 2017 [5]). This has been empirically shown in the several studies as well [10, 8]. This is made worse by the fact that these models are point predictions from optimization, clearly a marginalization approach should improve the calibration of these solutions.

Another strong case for Bayesian deep learning is its ability for uncertainty representation. Classical training outputs a single model without any representation of uncertainty. However, the BMA is a statement in probability. Thus, not only do we get additional explainability due to the probabilistic underpinnings, but we are able to examine the spread of the distributions, specifically the predictive distribution, $p(y|x, \mathcal{D})$.

All of these reasons give strong justifications for Bayesian deep learning. However, even if there are many potential theoretical benefits of using the Bayesian approach in deep learning, it must have strong empirical results for it to be considered over standard deep learning.

A potential roadblock for Bayesian deep learning comes up in its core property: the Bayesian model average. Theoretically the BMA is very appealing and is the leading justification behind making the case for Bayesian deep learning, but when it comes to computing it in practice, we run into a large issue, can we actually compute the Bayesian model average for these massive deep learning models?

6.2 Approximate Inference

Recall the Bayesian Model Average (BMA):

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

In deep learning, we have a very non-convex posterior landscape and a very high dimensional parameter space. This leads to a BMA that is not analytic, so we have to find a way to get around this. A straightforward solution is through Simple Monte Carlo approximation:

$$p(y|x, \mathcal{D}) \approx \frac{1}{J} \sum_j p(y|x, w_j), \quad w_j \sim q(w|\mathcal{D})$$

Here w_j are samples from a posterior $q(w|\mathcal{D})$ that approximates $p(w|\mathcal{D})$.

When using this Simple Monte Carlo approximation method, the main question at hand is what do we use for our approximate posterior, $q(w|\mathcal{D})$? The first approach is using so-called deter-

ministic methods. Here we approximate $p(w|\mathcal{D})$ with a parameterized approximate posterior $q(w|\mathcal{D}, \theta)$ that is usually *nice*, such as a Gaussian.

Some examples are Laplace (e.g., MacKay, 1995 [12]), Expectation Propogation (Minka, 2001 [14]), Variational, Standard Training. A concrete example of a Variational solution is to model $q(w|\mathcal{D}, \theta)$ as a Gaussian with θ parameterizing the mean. Then we use a Variational inference method to find the θ that minimizes the Kullback–Leibler (KL) divergence between our approximate posterior $q(w|\mathcal{D}, \theta)$ and our exact posterior $p(w|\mathcal{D})$, in other words we find $\text{argmin}_\theta \mathcal{KL}(q||p)$. It is also important to note that classical training is a variant of these deterministic methods. If we set our approximate posterior $q(w|\mathcal{D}) = \delta(w = w_{\text{MAP}})$, then this results in the exact form we discussed in the Classical vs. Bayesian Approach section. In a later section we will dive into one of these deterministic models that was introduced in our main reference [1].

Another approach is through Markov-chain Monte-carlo (MCMC). Here we create a Markov chain of approximate samples from $p(w|\mathcal{D})$. Noting that even though these are approximate samples, asymptotically they are exact representations of $p(w|\mathcal{D})$. Some examples are Metropolis-Hastings, Hamiltonian Monte Carlo (HMC), Stochastic gradient HMC, and Stochastic gradient Langevin dynamics. In the Bayesian Deep Learning setting these MCMC algorithms are gradient-based and "generate posterior samples iteratively using the gradient of loglikelihood... Since calculating likelihood on large datasets is expensive, people use stochastic gradients in place of full gradients." [11]

The momentum based methods tend to work as follows: the general algorithmic framework of these methods maintain two discrete-time sequences for the time-step t, p_t and θ_t, and then returns the samples $\{\theta_1, \theta_2, \ldots, \theta_T\}$ as an approximation to our distribution of interest $p(\theta|\mathcal{D})$. The specifics of the framework are as follows:

$$p_{t+1} = (1 - Dh)p_t - h\tilde{\nabla}_t + \sqrt{2Dh} \cdot \xi_t$$
$$\theta_{t+1} = \theta_t + hp_{t+1}$$

"θ_t is the parameter that we wish to sample and p_t is an auxiliary variable conventionally called the 'momentum'. Here h is step size, D is a constant independent of θ and p, $\xi_t \sim \mathrm{N}(0, I_d)$ and $\tilde{\nabla}_t$ is a mini-batch approximation of the full gradient [or the full gradient of the loglikelihood]." [11] Here we are construction samples by using the information from the previous iteration, the previous gradient of the logliehlihood, some scaled noise, and a momentum quantity, which is all scaled by the step size h.

A specific MCMC based technique designed specifically for Bayesian Deep Learning is described in the following paper: Cyclical Stochastic Gradient MCMC for Bayesian Deep Learning [18]. Their method, cyclical stochastic gradient (SG)-MCMC, provides a good example of using MCMC to approximate samples from the posterior, $p(w|\mathcal{D})$. Their main contribution is replacing the traditional decreasing stepsize schedule in stochastic gradient (SG)-MCMC with a cyclically changing one, which they claim improves learning complex high dimensional and multimodal distributions.

To go into more detail, there algorithm generates samples from multiple modes of a posterior distribution by running the cyclical step size schedule for many periods. They consider each step size cycle as a way of exploring a different part of the target posterior $p(\theta|\mathcal{D})$. As discussed previously, this is particularly compelling for Bayesian deep learning because it involves rich multimodal parameter posteriors with multiple distinct, strong solutions.

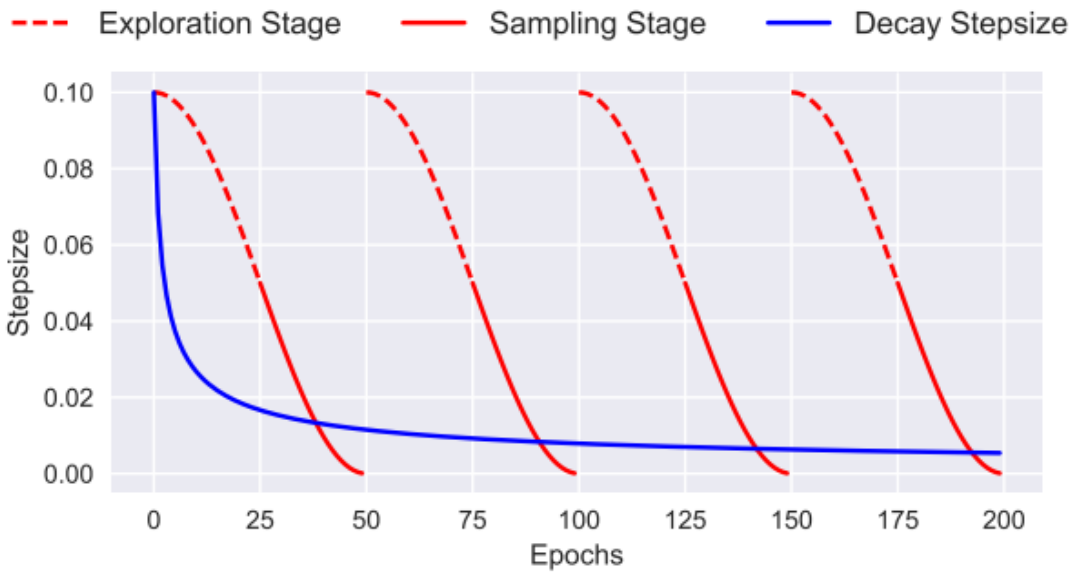


Figure 5: Cyclical stepsize [18]

We can see in the above figure 5 that there are two main periods for every cycle: exploration and sampling.

- **Exploration**: "when the stepsize is large (dashed red curves), we consider this stage as an effective burn-in mechanism, encouraging the sampler to take large moves and leave the local mode using the stochastic gradient." [18]
- **Sampling**: "when the stepsize is small (solid red curves), the sampler explores one local mode. We collect samples for local distribution estimation during this stage." [18]

The following is the pseudo-code for the algorithm:

Algorithm 1 Cyclical SG-MCMC.

Input: The initial stepsize α_0, number of cycles M, number of training iterations K and the proportion of exploration stage β.

for k = 1:K **do**

$\alpha \leftarrow \alpha_k$ according to Eq equation 1.

if $\frac{\text{mod } (k-1, \lceil K/M \rceil)}{\lceil K/M \rceil} < \beta$ **then**

% Exploration stage

$\theta \leftarrow \theta - \alpha \nabla \tilde{U}_k(\theta)$

else

% Sampling stage

Collect samples using SG-MCMC methods

Output: Samples $\{\theta_k\}$

Figure 6: SG-MCMC psuedocode [18]

Here $\tilde{U}_k(\theta)$ is equivalent to our $\tilde{\nabla}_t$ from before, a mini-batch approximation of the full gradient or the full gradient of the loglikelihood. The step-size α is chosen using a cyclical formula, its full details not discussed here. The pseudo-code follows the diagram that was previously described; alternating between the exploration and sampling stage in a cyclical pattern.

We have finished going over some of the common approaches to approximate our posterior and ultimately approximate the BMA. However, a key piece to note is that when performing our approximate inference of the BMA, our main goal is to accurately compute the predictive distribution $p(y|x, \mathcal{D})$, rather than find a generally accurate representation of the posterior. In particular, we mostly care about representing the posterior in regions that will make the greatest contributions to the BMA integral. The main papers expands on this topic in good detail: "In terms of efficiently computing the predictive distribution, we do not necessarily want to place point masses at locations given by samples from the posterior. For example, functional diversity is important for a good approximation to the BMA integral, because we are summing together terms of the form $p(y|x; w)$; if two settings of the weights w_i and w_j each provide high likelihood (and consequently high posterior density), but give rise to similar functions $f(x; w_i)$,

$f(x; w_j)$, then they will be largely redundant in the model average, and the second setting of parameters will not contribute much to estimating the BMA integral for the unconditional predictive distribution." [1]

Ultimately, since the BMA is not analytic, we want to best estimate the BMA integral given computational constraints. Thus, we will always have a trade-off between the quality of the approximation and the compute and time required.

7 SWAG and Multi-SWAG

In this section I will talk about the method of computing the BMA in the main paper that the report is based on [1]. There approach, Multi-SWAG, falls under the deterministic methods discussed in the previous section.

7.1 SWAG

To discuss the overall method, Multi-SWAG, we first must go over the SWAG method [13]. SWAG stands for Stochastic Weight Averaging Gaussian, and it is claimed to be a "simple and scalable method for Bayesian deep learning." [13] The main idea behind the method is to use iterations from stochastic gradient descent (SGD) to fit a low-rank diagonal Gaussian distribution. They reason that by using this approach, they capture the geometry of the posterior in the subspace of SGD.

The algorithm involves a good amount of details. To avoid getting bogged down in details, the discussion will be a high-level overview, aiming for a solid general understanding. Stochastic Weight Averaging Gaussian applied to the BMA problem can be described as follows:

1. Using recent developed theory, SGD with a constant learning rate is approximately sampling from a Gaussian distribution.

2. Compute the first two moments of SGD trajectory.

3. Use these moments to construct a Gaussian approximation in weight space.

4. Sample from this Gaussian distribution, pass samples through predictive distribution, and form a Bayesian model average. [16]

$$p(y_*|\mathcal{D}) \approx \frac{1}{J}\sum_{j=1}^{J} p(y_*|w_j)\,, \quad w_j \sim q(w|\mathcal{D})\,, \quad q(w|\mathcal{D}) = \mathcal{N}(\bar{w}, K)$$

$$\bar{w} = \frac{1}{T}\sum_t w_t\,, \quad K = \frac{1}{2}\left(\frac{1}{T-1}\sum_t (w_t - \bar{w})(w_t - \bar{w})^{\mathrm{T}} + \frac{1}{T-1}\sum_t diag(w_i - \bar{w})^2\right)$$

Figure 7: SWAG formulation [16]

The above equations show precisely how we construct our approximate posterior $q(w|D)$ by modeling it as a Gaussian distribution using the first two moments of the SGD trajectory as the building blocks of its mean and variance.

One caveat to this method is that the initial iterations of SGD tend to be fairly poor since we are quite far from any form of convergence, which is especially true in deep learning training. Thus, instead of using the moments of the SGD trajectory from the very beginning, we first have a pre-training phase which makes our SGD moments much more representative. This can be seen as a burn-in phase, and in our case for deep learning, it takes the majority of the training time. Additionally, the step size, or learning rate, for SGD starts fairly large and then is decreased so that we can achieve convergence in a fair amount of epochs (iterations). This is illustrated in the following figure:

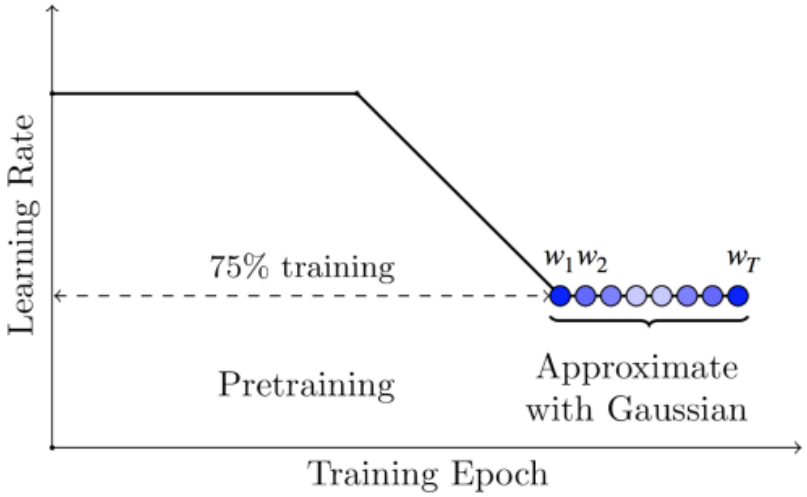


Figure 8: SWAG sampling [13]

7.2 Multi-SWAG

Multi-SWAG forms a Gaussian mixture posterior from multiple independent SWAG solutions. Each Gaussian is centered over a different basis of attraction. A basin of attraction is a "basin" or valley in our posterior landscape that typically represents a local minimum. As we discussed before there usually are multiple solutions in flat regions of the posterior and all these solutions tend to lead to better generalization. Thus, we want our posterior approximations to use multiple basins of attraction to result in more functional diversity. Due to this, Multi-SWAG gives more promise than Bayesian approaches that focus on approximating the posterior within single basin of attraction, such as plain SWAG. In essence, Multi-SWAG "incorporates multiple basins of attraction in the model average, but it additionally marginalizes within basins of attraction for a better approximation to the BMA." [1]

We now will briefly discuss the performance of Multi-SWAG compared to two other methods. These methods are a Variational Inference method similar to what we discussed before, and another popular method, Deep Ensembles [10]. Deep ensembles is a popular, recent method for producing accurate and well-calibrated predictive distributions. It is based on retraining a neural network model multiple times, taking advantage of the stochastic training nature, and then averaging the corresponding models. This creates an ensemble of deep learning models

that tends to incorporates multiple basins of attraction in the model average just like Multi-SWAG. However, a key difference is that Deep Ensembles do not marginalize within basins of attraction, whereas Multi-SWAG does.

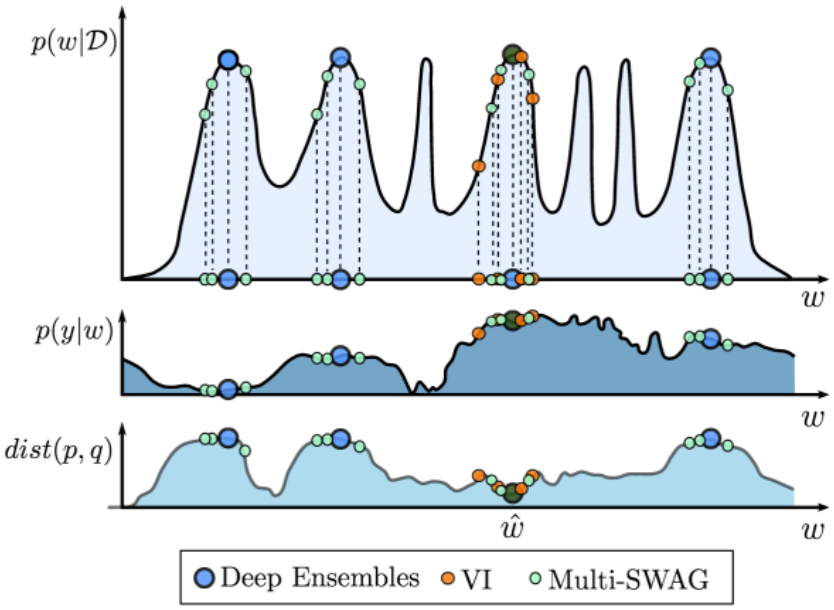


Figure 9: Approximating the BMA

The above figure 9 shows how Multi-SWAG and Deep Ensembles are able to represent different modes of our posterior, unlike the Variational method. Notice how even though many of the different modes, or basins, of the posterior have approximately the same shape and value they have drastically different likelihood, $p(y|w)$, shapes and values. Due to this, it is not always best to take the maximum value of our posterior basins as is shown in the likelihood plot but instead a marginalization within the basin, demonstration a strength of Multi-SWAG over Deep Ensembles. The last plot displays the distance between our approximation q and the true posterior p, given that we have only sampled from the dark green marker, $\hat{w}$. It shows that there is more to gain when we explore other basins, than continuing to explore the same basin due to how small the distance is near $\hat{w}$.

8 Application of these methods

So far this report has focused on describing the core principles of Bayesian Deep Learning. We have gone over how it revolves around the Bayesian Model Average, why using the Bayesian framework makes sense for deep learning, and some methods on how the BMA is computed in practice. However, we are missing a strong example of Bayesian Deep Learning in action. We can talk all about the core ideas and common approaches, but it would be interesting to see a concrete example of these techniques. Therefore, I will now briefly discuss a fascinating, recent application of Bayesian Deep Learning in planetary science.

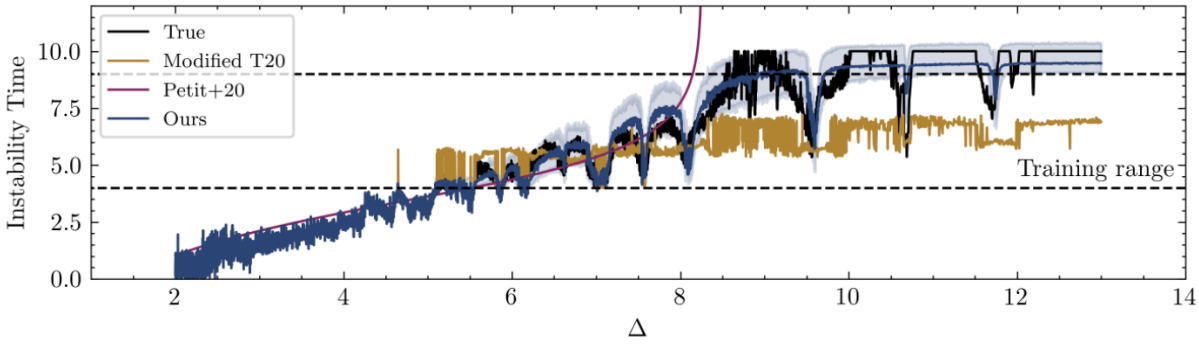
The topic at hand is the dynamical system describing planets orbiting a star, or a planetary system. The main objective is predicting how these chaotic systems will behave in the future, especially "what systems go unstable and experience a collision or ejection, and at what time will this occur?" [3] This problem has been tackled for over three centuries, but no closed-form solution exists for predicting the instability of these planetary systems.

It turns out that after many experiments with a multitude of different models, specifically designed deep learning models perform the best in this setting due to their well-calibrated inductive biases and large support. The main reason behind this observation, is that the deep learning models they use show similar algorithmic structure to the physical real-world model. However, classical training methods run into a problem, since this is a chaotic system there is a need for a distribution of instability times, not just one point. This is where Bayesian Deep Learning comes to the rescue.

They claim that "Bayesian deep learning allows us to marginalize the parameters of our neural networks, giving us the ability to account for uncertainty due to extrapolation outside the training set." [3] Specifically they use the Multi-SWAG method to fit 30 different modes, or basins of attraction, of the parameter space.

They utilize a three planet planetary system train and test datasets for their experiments. The experiments show that their Multi-SWAG model is two orders of magnitude more accurate than

existing analytic estimators. Additionally, the model gives a computational speedup over N-body integrations of up to 10^5 times, and all the estimates are over distributions. This means that they are able to calculate error bars, which represent the uncertainty in the model's predictions.



Our model evaluated on a 5-planet dataset. Error bars show the 68% confidence interval, and the center line shows the median estimate. The x-axis shows increasing separation between planets.

Figure 10: Instability predictions [3]

This figure 10 shows the Bayesian Deep Learning method (ours), performs much better than the alternative methods, Modifed T20, a classical machine learning approach, and Petit+20, an analytical method, in matching the true values.

They claim that this is quite an important development because "this enables a broad range of applications: using stability constraints to rule out unphysical configurations and constrain orbital parameters, and for developing efficient terrestrial planet formation models." [3] Overall, this shows a real-world application of Bayesian Deep Learning, and the authors benefit greatly from its properties that we previously discussed.

9 Downsides of Bayesian Deep Learning

It is important to also note that Bayesian deep learning has some clear downsides, and cannot be considered strictly *better* than classical deep learning. For one, there is an obvious increase in the computational cost. The classical approach only optimizes for one model, while the

Bayesian approach marginalizes over many, thus it is clear that the Bayesian approach will have higher computational cost. This is made worse by the fact that deep learning models already take a long time to train, so it may not be feasible in certain environments. Another issue with Bayesian deep learning is found at its core: the computational intractability of the Bayesian model average. We can produce all the theory we want about the strength of the Bayesian model average in regards to deep learning model classes, but the reality is that we almost always have to approximate the BMA when working with deep learning. The quality and cost of these approximations decide how well our predictive distribution will be, which can be a big question mark. Lastly, the Bayesian framework introduces a substantial amount of moving parts to deep learning. As just mentioned, we have to decide what is the best approximate inference method for our problem at hand, and the Bayesian framework introduces many additional hyperparameters (parameters chosen by the user) that are introduced in the Bayesian setting. These hyperparameters are not learned, can be quite difficult to set, and may greatly vary based on the problem at hand.

I wanted to close by outlining the most notable downsides of Bayesian deep learning to give some perspective on why this approach may not be as popular in the overall deep learning community; classical deep learning is still the go-to method when it comes to working with deep networks. However, even though there are several downsides concerning Bayesian deep learning, ongoing work is constantly being done to alleviate these downsides, and as we have shown earlier, Bayesian deep learning has its wealth of positives.

References

[1] **Andrew Gordon Wilson and Pavel Izmailov**. ***Bayesian Deep Learning and a Probabilistic Perspective of Generalization***. **2020**. arXiv: 2002.08791 [cs.LG].

[2] Brendan Colvert, Mohamad Alsalman, and Eva Kanso. "Classifying vortex wakes using neural networks". In: *Bioinspiration & Biomimetics* 13 (Sept. 2017). DOI: 10.1088/1748-3190/aaa787.

[3] Miles Cranmer et al. *A Bayesian neural network predicts the dissolution of compact planetary systems*. 2021. arXiv: 2101.04117 [astro-ph.EP].

[4] Timur Garipov et al. *Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs*. 2018. arXiv: 1802.10026 [stat.ML].

[5] Chuan Guo et al. *On Calibration of Modern Neural Networks*. 2017. arXiv: 1706.04599 [cs.LG].

[6] W. Ronny Huang et al. *Understanding Generalization through Visualizations*. 2020. arXiv: 1906.03291 [cs.LG].

[7] Pavel Izmailov et al. *Averaging Weights Leads to Wider Optima and Better Generalization*. 2019. arXiv: 1803.05407 [cs.LG].

[8] Pavel Izmailov et al. *Subspace Inference for Bayesian Deep Learning*. 2019. arXiv: 1907.07504 [cs.LG].

[9] Nitish Shirish Keskar et al. *On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima*. 2017. arXiv: 1609.04836 [cs.LG].

[10] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. *Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles*. 2017. arXiv: 1612.01474 [stat.ML].

[11] Zhize Li et al. *Stochastic Gradient Hamiltonian Monte Carlo with Variance Reduction for Bayesian Inference*. 2019. arXiv: 1803.11159 [cs.LG].

[12] D. Mackay. "Probable networks and plausible predictions - a review of practical Bayesian methods for supervised neural networks". In: *Network: Computation In Neural Systems* 6 (1995), pp. 469–505.

[13] Wesley Maddox et al. *A Simple Baseline for Bayesian Uncertainty in Deep Learning*. 2019. arXiv: 1902.02476 [cs.LG].

[14] Thomas P. Minka. *Expectation Propagation for approximate Bayesian inference*. 2013. arXiv: 1301.2294 [cs.AI].

[15] Carl Rasmussen and Zoubin Ghahramani. "Occam's Razor". In: *Advances in Neural Information Processing Systems* (Feb. 2001).

[16] Andrew Gordon Wilson. *Bayesian Deep Learning and a Probabilistic Perspective of Model Construction*. International Conference on Machine Learning Tutorial. 2020.

[17] Andrew Gordon Wilson. *The Case for Bayesian Deep Learning*. 2020. arXiv: 2001.10995 [cs.LG].

[18] Ruqi Zhang et al. *Cyclical Stochastic Gradient MCMC for Bayesian Deep Learning*. 2020. arXiv: 1902.03932 [cs.LG].