Fast Uncertainty Estimates and Bayesian Model Averaging of DNNs

Wesley Maddox[1] **Timur Garipov**[4,5] **Pavel Izmailov**[1] **Dmitry Vetrov**[2,3] **Andrew Gordon Wilson**[1]
[1]Cornell University, [2]Higher School of Economics, [3]Samsung-HSE Laboratory,
[4]Samsung AI Center in Moscow, [5]Lomonosov Moscow State University

Abstract

Reliable uncertainty estimates for both the weights and the predictions of deep learning models have proven hard to come by due to the complexity and size of the models used. In many other areas, Bayesian methods are used to capture uncertainty estimates and incorporate prior knowledge into the modelling process; however, they often suffer from scalability issues when applied to deep learning models. We extend the recently developed stochastic weight averaging (SWA) procedure in a simple and computationally efficient manner, creating Gaussian approximations to the true posterior distribution. This procedure, termed SWA-Gaussian (SWAG), produces reliable uncertainty estimates, while maintaining accuracy in Bayesian model averaging. Code is available at `https://github.com/wjmaddox/swa_uncertainties`.

1 Introduction

Deep learning models are typically over-confident in their predictions [4] and will often silently fail in practice. Incorporating Bayesian model averaging has been shown to help in model robustness; however, Bayesian methods typically have issues scaling to large datasets, requiring approximate inference techniques. In Bayesian inference, we are interested in learning a distribution over the set of model parameters instead of simply a single model, and using that distribution to reason with uncertainty. To do so requires using Bayes' rule, $p(\theta|D) \propto p(D|\theta)p(\theta)$.

Typically, this posterior distribution cannot be computed analytically and must be used approximately. Three standard methods of approximate Bayesian inference exist: Markov Chain Monte Carlo (MCMC), variational inference (VI), and Laplace approximations. MCMC is asymptotically exact, but computationally expensive even for small models; VI and Laplace approximations induce bias by assigning a parametric form, but perform well in practice. We instead introduce another method: using the iterates of stochastic gradient descent by performing Polyak-Ruppert averaging [13, 17]. Like Laplace and VI, this method introduces a parametric structure, but directly uses the iterates instead of estimating a different set of parameters (like VI) or using curvature (like Laplace).

2 Background

In this section, we describe some prior related work on Laplace approximations for neural networks and on the asymptotic distribution of the iterations of stochastic gradient descent (SGD).

2.1 Laplace Approximations for Bayesian Neural Networks

All three standard techniques for approximate inference can be applied to Bayesian neural networks and deep learning. However, variational inference and Laplace approximations have gained the most interest, as they show some promise in scaling to deep neural networks.

Laplace approximations use a Taylor expansion around the maximum a posteriori (MAP) estimate, θ^*, to construct a Gaussian distribution around the true posterior [8]. This approximation to the posterior distribution can be written as

$$p(\theta|D) \approx p(\theta^*) \exp\{-\frac{1}{2}(\theta - \theta^*)' \mathbb{H}_{|\theta^*}(\theta - \theta^*)\}, \quad (1)$$

where $\mathbb{H}_{|\theta} = \nabla_\theta p(y|\theta) \nabla_\theta p(y|\theta)'$ is the Hessian (or curvature) of the likelihood evaluated at the MAP estimate. The Laplace approximation is motivated by the Bernstein-von Mises Theorem, which states that in the large data limit and under certain regularity conditions, posterior

distributions tend towards $N(\theta^*, \mathcal{I}^{-1})$, where $\mathcal{I}^{-1}$ is the inverse of the Fisher information matrix (the expected value of the Hessian), in total variation distance [19].

The Laplace approximation was first used for Bayesian neural networks in [7], where a diagonal approximation to the inverse of the Hessian was utilized for computational reasons. For scalability reasons, [16] propose the use of either a diagonal or block Kronecker factored (KFAC) approximation to the Hessian matrix for Laplace approximations. The diagonal approximation is easily written as $\mathbb{H} \approx diag(\mathcal{I}) = \sigma diag(\mathbb{E}(\nabla_\theta \log p(y|x,\theta)^2) + \tau,)$ where τ is the standard deviation of the assumed Gaussian prior on the weights and σ is a scaling factor.[1] For a deep neural network, the KFAC approximation writes the Hessian matrix as a Kronecker factorization between the covariance of the activations by each layer, $Q_l = a_l a_l'$, and the Hessian of the error before the activation function: $\mathbb{H}_l = {\nabla_{h_l}}^2 E$. This type of approximation has been successfully used as a pre-conditioner for standard stochastic gradient descent (SGD) [10] and for MCMC sampling with stochastic gradient Langevin dynamics [11]. This potentially trades off accuracy in the approximation for computational feasibility.

2.2 Asymptotic Normality of SGD and Averaging

Instead of relying on Laplace approximations, we propose to use the covariance of Stochastic Weight Averaging (SWA) [5], controlling the structure of the posterior through the hyper-parameters of SGD as in [9].

In stochastic weight averaging (SWA) [5], a standard decaying learning rate is used until SWA iterates are stored. Then, the learning rate is held at a high constant value, with the final iteration of each epoch being stored in a running average of the parameters, similar to thinning in MCMC. This running weight average is then used to compute predictions for deep neural networks. In the experiments performed in [5] and [2], SWA outperforms other training techniques at minimal computation cost, producing state-of-the-art results on semi-supervised learning tasks. We also note that a similar approximation to both SWA and Eq. 2 was used in [3] for producing confidence intervals from the LASSO for high-dimensional linear regression.

It has been shown that the sample average of SGD iterations with a constant learning rate will converge around a stationary point, θ', and the covariance of this distribution will be $\mathbb{H}(\theta')^{-1} S \mathbb{H}(\theta')^{-1}$, where $S = E(\nabla_\theta l(\theta) \nabla_\theta l(\theta)^T)$ [17, 13, 1, 9]. That is, the asymptotic distribution of the running average is

$$\frac{1}{T}\sum_{i=1}^{T} \theta_i \approx N(\theta', \mathbb{H}(\theta')^{-1} S \mathbb{H}(\theta')^{-1}), \qquad (2)$$

where T is the total number of steps and θ_i is each iteration. This motivates the usage of this distribution as an approximate posterior distribution. Additionally, burn-in phases do not change the asymptotic posterior, and additionally can help the rate of convergence in convex stochastic optimization [14, 1]. Finally, [9] also showed that constant step size SGD (including momentum variants) can be used as a variational-EM type algorithm by viewing the sequence of SGD iterates as a Markov chain. Directly computing this covariance would be quite expensive; however, [3] proposed using a version of the sample covariance, Σ, as an estimator of this matrix, thus justifying our usage of the sample covariance of SWA.

3 Scalable Gaussian Posterior Approximations using SWA

We propose to make a Gaussian approximation to the true posterior for Bayesian neural networks by using the SWA estimate as the posterior mean, using the same algorithm as in [5], while also estimating the covariance. Our proposed methods thus have the same expectation in weight space as SWA, but differ in their covariance, and thus in their predictions. We propose three modifications of varying time and memory costs: SWAG, which is diagonal and only requires two extra copies of the model parameters, SWAG-LR, which requires $k+1$ copies of the model parameters, and SWAG-Laplace, which requires the same storage as SWAG-LR but the computation of a diagonal Hessian computation after training.

3.1 SWA-Diagonal Gaussian (SWAG) Approximations

We first propose to additionally keep a running average of the second moment while running SWA: $\mathbb{E}(\theta^2) = \bar{\theta^2} = \frac{1}{T}\sum_{i=1}^{T} \theta_i^2$. After training, the variance can then be estimated using the standard sample variance identity: $V_{\theta_{SWA}} \approx \bar{\theta^2} - \bar{\theta}^2$. We then compute a diagonal Gaussian approximation using these estimated moments. This gives the diagonal Gaussian distribution, $N(\theta_{SWA}, V_{\theta_{SWA}})$. In our experiments, we term this method SWAG.

To generate samples from the approximate posterior, two copies of the model must be stored: the SWA estimate of the mean and the estimate of the second moment. Generating these ensembles is quite cheap as it only requires sampling independent normal distributions and rescaling

[1] We mention here that even computing these variances can be quite expensive: our PyTorch implementation took about 400 seconds on Titan X GPUs and a grid search for σ took about 1000 seconds for VGG16 on CIFAR-10.

them by the variance, while predictions from an ensemble of size K require simply K forward passes through the network architecture.

3.2 SWA - Low Rank Covariance Approximations (SWAG-LR)

To improve this approximation beyond the diagonal, a low-rank covariance approximation can be computed instead of the second moment. Noting that the sample covariance matrix can be written as the sum of outer products, $\Sigma = \frac{1}{T-1}\sum_{i=1}^{T}(\theta_i - \bar{\theta})(\theta_i - \bar{\theta})' = XX'$, where $(X)_i = \frac{1}{\sqrt{T-1}}(\theta_i - \bar{\theta}_i)$, and $\bar{\theta}_i$ is the running estimate of the parameters' mean for the ith sample. To sample and compute moments, only X must be stored, which scales as $O(Td)$, where d is the number of parameters and T is the number of samples of the sample covariance.[2] In practice, we keep a running estimate with $N = 20$, for GPU memory issues. A full-rank version of this approximation was first described in [9], but stores the full Σ. In our experiments, we term this method SWAG-LR.

3.3 SWA Low Rank Asymptotic Covariance Approximation (SWAG-Hessian)

We can also approximate the asymptotic covariance of Polyak-Ruppert averaging by using the low rank covariance approximation described in the previous section and using diagonal Hessian approximations. Instead of computing the full inverse Hessian, we will again use a diagonal approximation, multiplying it by the low rank sample covariance approximation described in the previous section. For sampling, this gives the approximation

$$\left(\mathbb{H}(\theta')^{-1}\Sigma\right)^{1/2} \approx \text{diag}\left(\frac{1}{\tau + \mathcal{I}_{ii}}\right) X. \qquad (3)$$

This approach should merge the trajectory dependence of the SWAG approximation with the curvature information of Laplace approximations. The trajectory dependence should remove the need for scaling of the Laplace approximation in practice, while the curvature information from the Hessian should incorporate further information about the loss geometry. In our experiments, we term this method SWAG-Hessian.

[2] It is worth noting that the resulting Gaussian is degenerate, but any Gaussian generated from the sample covariance will be degenerate unless $T > d$. To draw samples, we now only need to draw a T-dimensional independent Gaussian vector.

4 Results

4.1 Toy Distribution Problem

To illustrate the possibility of using SGD trajectories as an approximate posterior distribution, we first consider a toy mixture distribution with likelihood proportional to: $N(x|0, 0.1^2) + N(x|1, 0.15^2)^{200}$. Experimental details are in Appendix B.2. The results of the posterior approximation for Laplace, SWAG, and SWAG-Hessian are shown in Figure 1. While all three are Gaussian approximations, there are vast differences between them: SWAG seems to estimate the curvature of the distribution quite accurately, while Laplace is trapped in a single mode. Finally, SWAG-Hessian is somewhat over-confident, with too small of a variance to adequately model the true distribution.

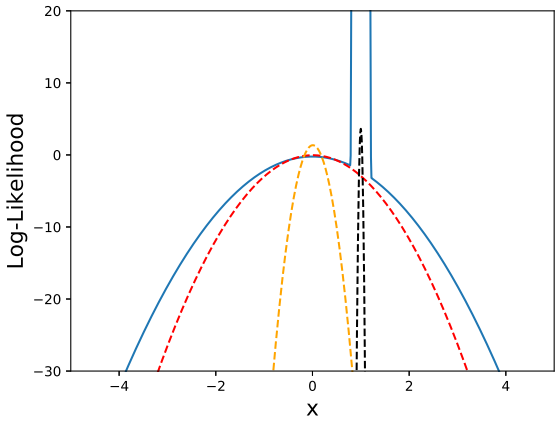
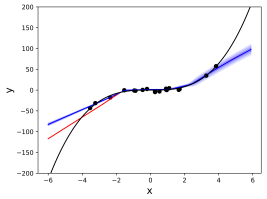


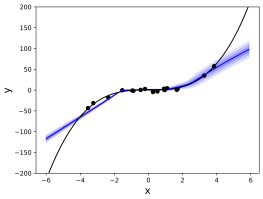
Figure 1: SWAG and Laplace approximations of a toy mixture distribution. Blue is true log-likelihood, while red is SWAG approximation, orange is SWAG-Hessian, and black is Laplace.

4.2 Toy Regression Problem

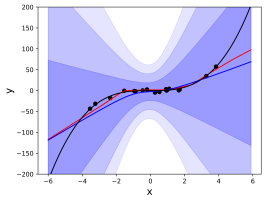
Next, we compare the SWAG methods with diagonal Laplace approximations on a toy regression problem: modelling $y \sim x^3 + N(0, 3^2)$ with a simple two-layer MLP as in [16]. Experimental details are in Appendix B.3. We find that the two covariance based posteriors are highly confident within the range of the training data, but less confident outside. We additionally see that Laplace and SWAG-Hessian are essentially confident nowhere, even after optimizing for the best scale. This suggests that SWAG and SWAG-LR are better approximations of the true posterior distribution on this toy neural network, and their posterior mean mimics the SWA point estimate.



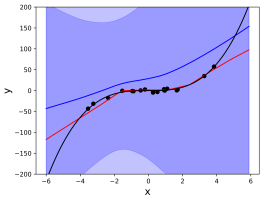
(a) Diagonal SWA approximation.



(b) Low-rank, sample covariance SWA approximation.



(c) Diagonal of Hessian approximation. Scaling was tuned with a validation set.



(d) SWAG-Hessian approximation. Scaling was tuned with a validation set.

Figure 2: Posterior means and uncertainty estimates for a toy regression problem. Black points are testing data, black line is true function, red line is SWA point estimate, while blue line is average prediction from 50 samples of the approximate posterior, and blue shading are successive standard deviations from the mean posterior.

4.3 Uncertainty Estimation with DNNs

All further experiments used the VGG16 architecture [18] trained for 300 epochs as in [5]. Further experimental details are in the Appendices.

We next show the calibration gap of the SWAG-based Bayesian model averages on CIFAR100 in Figure 3. Well-calibrated models of a probability distribution should be accurate on average $p\%$ of the time over repeated draws if their output probability is $p\%$. Here, we see that all three of the SWAG approximation models are significantly better calibrated than both SWA and diagonal Laplace approximations. In fact, the two models with covariance structure (SWAG-LR and SWAG-Hessian) are quite well-calibrated, reaching the expected calibration line. We verify this by computing the expected calibration error [12] in Table A.3.

4.4 Out-of-Sample Uncertainty

The Bayesian approach should additionally provide improvements in uncertainty when testing on data that is outside of the data distribution. We describe this experiment in further detail in Appendix B.5. Clearly, out-of-sample images should be recognized as more uncertain, and thus

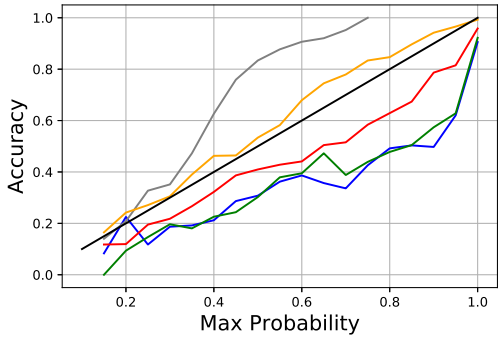


Figure 3: Calibration plots for VGG16 on CIFAR100. Black is the true accuracy/maximum probability line. Blue is SWA, green is diagonal Laplace, red is SWAG, yellow is SWAG-LR, and grey is SWAG-Hessian.

have higher entropy than in-distribution test images. Thus, models that accurately capture uncertainty should be able to distinguish between in and out-of-sample images, as measured by entropy of predictions. Looking at the accuracy of a threshold formed by entropy, as shown in Figure A.1 and the AUC of this classifier, which is given in Table A.2, we see that the SWAG models (but not SWAG-Hessian) can distinguish better between the in and out of sample distributions better overall than Laplace and SWA. This suggests that uncertainty is better modeled with these posterior approximations.

4.5 Bayesian Model Averaging on DNNs

We test our fast posterior approximation with deep neural networks on CIFAR-10 and CIFAR-100 datasets. For models trained on CIFAR-10, we additionally test on the recently developed CIFAR-10.1 set [15] to assess generalization. Further experimental details are in Appendix B.4.

The accuracy of Bayesian model averaging when compared to both SWA and SGD point estimates are shown in Table A.1. We note that with 30 samples the SWAG approximations improve the ensemble accuracy by a small but statistically significant amount against standard SWA, and require no extra hyper-parameter tuning. Interestingly, they do not outperform the empirical distribution of constant learning rate SGD, but perform quite similarly to the empirical distribution. Based on this performance, this suggests that the Gaussian approximation is close to the empirical distribution of SGD. Of course, at test time, using the empirical distribution requires considerably more storage than generating the models on the fly, as in both SWAG methods. On the CIFAR10.1 dataset, we also see that all of the SWA models outperform their

SGD counterparts, suggesting that the SGD models are in a sharper optima, and that both SWA and SWAG provide better generalization capabilities.

In Figure A.2, we test the accuracy of the model in comparison to the number of samples from the approximation distribution to test how many samples are necessary for improvements using this type of ensembling. This is critical for evaluating the test time computational efficiency of the proposed methods.

5 Conclusion

We have proposed SWAG, a Gaussian posterior approximation for DNNs using the covariance of the SGD iterates. SWAG produces much better model calibration and out-of-sample uncertainty estimation than Laplace posterior approximations. The proposed approximations are also considerably computationally cheaper, and require no hyper-parameter tuning. SWAG also seem to act similarly to Bayesian model averages produced by the empirical distribution of SGD, suggesting that the Gaussian approximation is well founded. There are many exciting directions to extend this work: particularly in incorporating more interesting covariance matrix and Hessian approximations.

Acknowledgements

We would like to thank Ruqi Zhang and Jacob Gardner for helpful discussions, particularly in regards to storage issues of the empirical covariance matrix. WM is supported by an NSF Graduate Research Fellowship under Grant No. DGE-165044.

References

[1] S. Asmussen and P. W. Glynn. *Stochastic simulation: algorithms and analysis*. Number 57 in Stochastic modelling and applied probability. Springer, New York, 2007. OCLC: ocn123113652.

[2] B. Athiwaratkun, M. Finzi, P. Izmailov, and A. G. Wilson. Improving Consistency-Based Semi-Supervised Learning with Weight Averaging. arXiv: 1806.05594, June 2018.

[3] X. Chen, J. D. Lee, X. T. Tong, and Y. Zhang. Statistical Inference for Model Parameters in Stochastic Gradient Descent. arXiv: 1610.08637, Oct. 2016.

[4] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On Calibration of Modern Neural Networks. In *ICML*, June 2017.

[5] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson. Averaging Weights Leads to Wider Optima and Better Generalization. In *To appear in UAI*, 2018, Mar. 2018. arXiv: 1803.05407.

[6] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In *NIPS*, 2017.

[7] D. J. C. MacKay. A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation*, 4(3):448–472, May 1992.

[8] D. J. C. MacKay. *Information theory, inference, and learning algorithms*. Cambridge University Press, Cambridge, UK ; New York, 2003.

[9] S. Mandt and M. D. Hoffman. Stochastic Gradient Descent as Approximate Bayesian Inference. *JMLR*, 18:1–35, 2017.

[10] J. Martens and R. Grosse. Optimizing Neural Networks with Kronecker-factored Approximate Curvature. *arXiv:1503.05671*, Mar. 2015. arXiv: 1503.05671.

[11] Z. Nado, J. Snoek, B. Xu, R. Grosse, D. Duvenaud, and J. Martens. Stochastic Gradient Langevin Dynamics That Exploit Neural Network Structure. In *ICLR Workshop Track*, 2018.

[12] M. P. Naeini, G. F. Cooper, and M. Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *AAAI*, pages 2901–2907, 2015.

[13] B. T. Polyak and A. B. Juditsky. Acceleration of Stochastic Approximation by Averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, July 1992.

[14] A. Rakhlin, O. Shamir, and K. Sridharan. Making Gradient Descent Optimal for Strongly Convex Stochastic Optimization. In *ICML*, Sept. 2011.

[15] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar. Do CIFAR-10 Classifiers Generalize to CIFAR-10? arXiv: 1806.00451, June 2018.

[16] H. Ritter, A. Botev, and D. Barber. A Scalable Laplace Approximation for Neural Networks. In *ICLR*, 2018.

[17] D. Ruppert. Efficient Estimators from a Slowly Convergent Robbins-Munro Process. Technical Report 781, Cornell University, School of Operations Report and Industrial Engineering, 1988.

[18] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv:1409.1556*, Sept. 2014. arXiv: 1409.1556.

[19] A. W. v. d. Vaart. *Asymptotic Statistics*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, Cambridge, 1998.

A Supplementary Tables and Figures

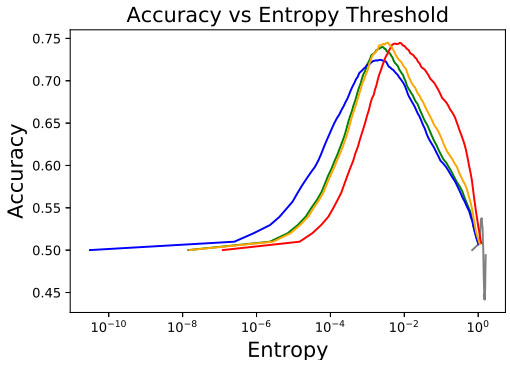


Figure A.1: Accuracy of the classifier that is attempting to predict out-of-sample images by thresholding the entropy. Blue is SWA, red is SWAG, orange is SWAG-LR, green is diagonal Laplace, while grey is SWAG-Hessian.

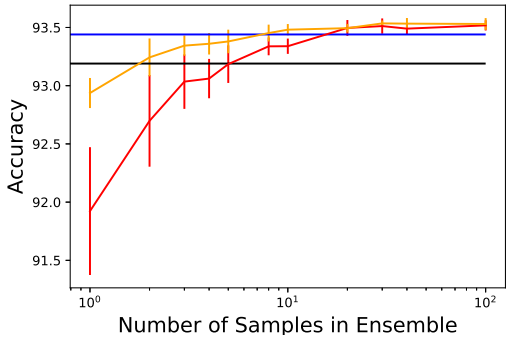


Figure A.2: Accuracy by number of models in ensemble for VGG16 trained on CIFAR-10 dataset. Blue line is SWA point estimate, while black line is SGD point estimate. Red is SWAG and orange is SWAG-LR.

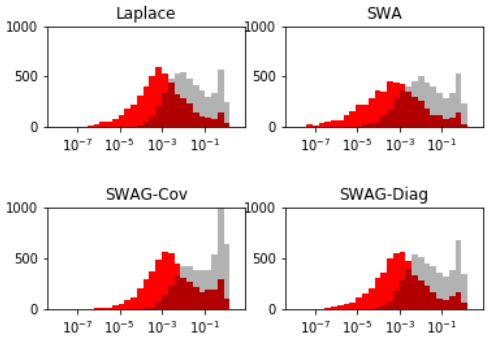


Figure A.3: Distribution of the entropies for (diagonal) Laplace, SWA, and SWAG based approximations for CIFAR10 test set, with VGG16 trained on 5 classes of CIFAR10. Red is within distribution, while grey is out-of-distribution. X axis here is entropy score, with a maximum possible entropy of $-\log 0.2 = 1.61$, and the y axis is the count of entropies at a test point for a given bin.

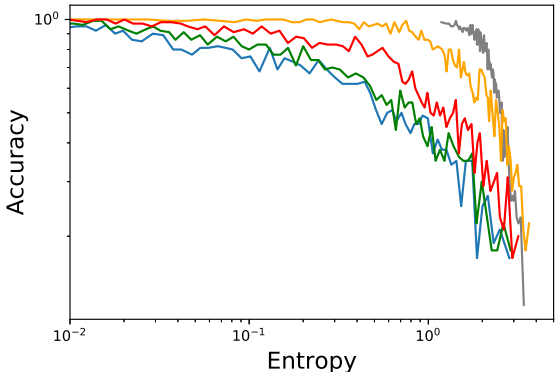


Figure A.4: Accuracy by entropy of models on CIFAR 100. Blue is SWA, red is SWAG, orange is SWAG-LR, green is diagonal Laplace, while grey is SWAG-Hessian.

B Experimental Details

B.1 SWA Averaging Details

Unless otherwise specified, we followed the same procedure for SWA as in [5], using VGG16 [18] models. We ran models for 300 epochs using SGD with momentum with $\epsilon_t = 0.01$ and $\eta = 0.9$. We additionally used a decaying learning rate schedule prior to the beginning of SWA averaging, with a constant learning rate of $\epsilon = 0.01$ afterwards. For SWA, we save the state at the final batch in the epoch at every epoch, beginning with epoch 161.

B.2 Toy Distribution

To form the Laplace approximation, we performed a grid search to find the maximum, and then evaluated the second derivative at that point to compute the standard deviation.

To form the SWAG approximation, we ran SGD for 100 steps, computing the mean and standard deviation of the last 80 steps, using this as the posterior approximation. Here, there is no noise, and so we had to choose the learning rate via cross-validation. We did this by running SGD 10 times with random initialization $x_0 \sim N(0, 3^2)$, and computing the mean squared error between the log-likelihood of the Gaussian distribution formed by the

Table A.1: Accuracy comparisons with VGG architecture for CIFAR-10 and CIFAR-100 ensembling experiments, including CIFAR-10.1 dataset. CIFAR-10.1 models were trained on CIFAR-10, as in [15]. All numbers are averaged over five runs, with 10 replications for each model also coming for the sampling based models.

Dataset (epochs)	SGD, point	SGD, empirical	SWA	SWAG, 30 samples	SWAG-LR, 30 samples
CIFAR-10 (300)	93.19 ± 0.22	93.64 ± 0.14	93.44 ± 0.09	93.53 ± 0.14	93.57 ± 0.15
CIFAR-10.1	84.93 ± 0.32	85.78 ± 0.20	86.14 ± 0.59	86.18 ± 0.57	86.24 ± 0.67
CIFAR-100 (300)	73.29 ± 0.38	74.74 ± 0.26	74.04 ± 0.25	74.44 ± 0.31	74.57 ± 0.39

Table A.2: Area under the curves computed on the CIFAR 5 tasks using entropies as uncertainty estimates.

Method	AUC
Laplace	80.41
SWA	79.26
SWAG	80.84
SWAG-LR	80.86
SWAG-Hessian	47.22

Table A.3: Expected calibration error on 1 CIFAR100 model. Lower is better.

Method	ECE
Laplace	0.7604
SWA	0.7650
SWAG	0.3794
SWAG-LR	0.6001
SWAG-Hessian	0.7093

SGD iterates and the true distribution's log-likelihood.

Finally, to form the SWAG-Hessian approximation, we used the SWA mean and computed its second derivative, and then calculated the standard deviation.

B.3 Toy Regression Problem

We draw 20 samples of $x \sim U(-4, 4)$ and $y \sim x^3 + N(0, 3^2)$, using a Gaussian distribution for the likelihood. We use a two-hidden layer MLP with ReLU activations and 7 hidden units to directly compare to [16], with weight decay of $1e-4$ creating a Gaussian prior with that standard deviation. We used the standard learning rate schedule, but with $\epsilon_t = 0.001$ and $\epsilon = 0.01$.

To estimate the diagonal Laplace approximation, we followed a grid search on a validation set over the scaling factor of the Hessian as in [16], but did not also scale τ, instead using the weight decay factor that we trained the network with. Otherwise, we would be creating both a data-dependent prior and scaling the prior incorrectly as compared to what is used. To do otherwise would essentially be modifying the prior to fit the data distribution post facto, violating the assumptions of Bayes' Theorem.

B.4 Bayesian Model Averaging with DNNs

Comparisons with SGD are with independently trained models that do not use the SWA learning rate (instead decaying in the same manner to zero), and are supposed to represent best practice with this architecture. Here, reported standard deviations come from both 5 independent runs of the model and $K = 30$ independent samples from the approximate posterior (either empirical or Gaussian) when possible. Here, empirical denotes 30 samples from the empirical distribution formed by SGD iterates, and so the comparison is to test if the Gaussian used by SWAG is a good proxy for this distribution.

In both cases, Bayesian predictions on the testing data, y^*, using the approximate posterior, q, can now be made using Bayesian model averaging:

$$p(y^*|y) = \mathbb{E}_{p(\theta|y)}(p(y^*|\theta)) \approx \frac{1}{K}\sum_{i=1}^{K} p(y^*|\theta_i), \quad \theta_i \sim q(\theta|Y). \tag{4}$$

As we use weight decay, this corresponds to a version of a Gaussian prior on the weights, implying a proper prior.

B.5 Out-of-Sample Uncertainty

To out of sample uncertainty, we use a modified version of the experiment in [6]. We train VGG models on five classes from the CIFAR-10 dataset, but now test on the full 10 class dataset. We hypothesize that predictions on images from the five out-of-sample classes should be more uncertain than images from the five in-sample classes. Thus, we consider the entropy of the predictions, and use a threshold to classifier test images beneath the threshold as in-class, and test images above the threshold as out-of-class. We then compute the AUC based on the accuracy of this threshold.

As to the poor performance of the SWAG-Hessian method

on this task, we found paradoxically, that out-of-sample images were nearly indistinguishable in terms of entropy, and if anything, had a slightly lower distribution. We attribute this to the Hessian providing too much regularization, and thus the predictions have too much variance associated with them.